PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR
FACULTAD DE INGENIERÍA
ESCUELA DE SISTEMAS Y COMPUTACIÓN



ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE GESTIÓN DEL CUADERNO ELECTRÓNICO DEL ESTUDIANTE

NOMBRES:

SALGADO PROAÑO JOSÉ ANTONIO

GRANJA ORAMAS MARÍA ISABEL

"Trabajo previo a la obtención del Título de Ingeniero/a de Sistemas y Computación"

Quito, marzo 2020

Dedicatoria

Este Trabajo de Titulación está dedicado a:

Mis padres Mario y Adela, por ser un pilar fundamental en mi vida, por ser mis modelos a seguir dándome un buen ejemplo para mi desarrollo tanto de manera humana como profesional, por ser mis maestros de vida, por todos los valores y principios que me inculcaron, por siempre apoyarme y ayudarme a cumplir mis sueños, y principalmente, por todo su amor.

Mis hermanos Mario Agustín y María Victoria, porque siempre me brindan todo su amor y cariño, por ayudarme a crecer como persona, por todos los consejos que me han dado y por todas las aventuras que hemos vivido juntos.

A mi enamorado y compañero de este Trabajo de Titulación, por ser una persona increíble, el cual me ha enseñado y me ha apoyado bastante durante todo este trayecto.

María Isabel Granja Oramas

Dedicatoria

Dedico este Trabajo de Titulación:

A mi madre Mayra, por siempre estar apoyándome desde el principio de mi vida y por darme un ejemplo de fortaleza inquebrantable.

A mi padre René, por siempre ser para mí un gran ejemplo de perseverancia y por inculcarme muchos valores en mi vida.

De forma general a mis padres, por mostrarme que nada fácil vale la pena y que si se toma decisiones adecuadas en tu vida todo es posible. Gracias por guiarme por el camino hacia la superación y, sobre todo, nunca olviden cuanto los amo y los respeto.

A mis abuelos, Armando, Bertha, Carmen y Nelson, por enseñarme que nada es imposible si realmente pones todo de ti para lograrlo y por mostrarme que las mejores armas contra cualquier adversidad siempre son el conocimiento, la experiencia y las buenas intenciones.

Al resto de mi familia, por acogerme y aconsejarme cuando lo he necesitado.

A mis amigos cercanos, por apoyarme, por ayudarme a creer más en mí y por motivarme a seguir adelante sin importar lo dificultad del camino.

A mis respetados profesores, por permitirme construir con ladrillos de conocimientos los cimientos de mi vida profesional, por potenciar mis talentos y por ayudarme a enrumbar mis sueños en base a mis afinidades académicas.

A mi enamorada Isabel, por siempre estar a mi lado en las buenas y en las malas, y por permitirme soñar y crecer junto a ella. Sé que juntos alcanzaremos todo lo que nos propongamos. Todo esto es posible gracias a ustedes.

José Antonio Salgado Proaño

Agradecimientos

Estoy agradecida primeramente con Dios, debido a que durante todo el transcurso de mi vida he tenido la dicha de estar rodeada de gente buena, que me han apoyado y me ha dado una mano de una u otra forma.

Agradezco a mi familia, la cual es digna de admirar en todos y cada uno de los aspectos. Me han enseñado numerosísimas cosas en base a su experiencia y conocimiento, siempre están prestos a ayudarme, además de que siempre me entregan bastante amor, cariño y afecto.

Agradezco a mi Colegio Cardenal Spellman Femenino de Quito, ya que en el pude crear recuerdos muy memorables y hacer grandes amistades que perdurarán por siempre. Conocí a mi mejor amiga Valeria V.

Agradezco a la Pontificia Universidad Católica del Ecuador, ya que pude conocer una gran cantidad de buenos amigos y personas de gran corazón, tanto profesores como compañeros de aula. Durante esta etapa pude conocer a un ser extraordinario llamado José S. quien tengo el gusto de que sea mi enamorado y compañero de profesión, una persona muy dedicada e inteligente, lleno de valores que me enseña y me ayuda a crecer día a día. Además, conocí a grandes amigos y compañeros como Francisco U., Alexei R., Cesar A., Israel M., Daniel V., Cristopher C. y Alexis C.; quienes son personas muy valiosas puesto que siempre estuvieron pendientes de nosotros, nos ayudaron y crecieron de forma profesional y personal a nuestro lado.

A mi directora del presente Trabajo de Titulación, Suyana A. y a cada uno de los revisores Oswaldo E. y Juan Francisco C. por ayudarnos durante la elaboración del Trabajo de Titulación y por guiarnos, brindándonos los mejores consejos.

A mis profesores, Suyana A., Oswaldo E., Eddy S., Miguel O., Alfredo C., Beatriz C., Francisco R., Edison M., Fabián de la C., Xavier C., Charles E., Jorge A., Damián N., Katherine A., Pablo T., Xavier C., entre otros; los cuales me ayudaron para mi formación académica y también a crecer como persona, enseñándome buenos valores. Les agradezco infinitamente por todo su conocimiento transmitido, por todas sus historias y sus experiencias compartidas que me permitieron aprender bastante y, principalmente por su valiosa amistad.

Agradezco a Miguel O. y al equipo del LTIC, por habernos brindado su apoyo y por prestarnos la ayuda necesaria para la elaboración del sistema.

A parte, agradezco bastante a Mayra P., René S., Carmen V., Armando P., Berta L., y Nelson S. por siempre estar pendientes de nosotros, por su cariño y por su ayuda en todo momento.

María Isabel Granja Oramas

Agradecimientos

Desde que era pequeño siempre creí en el potencial de los sueños y vaya que no me equivoqué. Construir, crear e imaginar son facultades que siempre me han caracterizado y por ello crecí recibiendo palabras de aliento de mis seres queridos que me decían que debo enrumbar dicha creatividad en una carrera que me permita alcanzar mis metas para lograr ser un gran profesional. Jamás olvidaré como me encantaba armar ciudades con Legos y las tantas aventuras que vivía al jugar mi viejo Nintendo 64 junto a mis padres. De esto aprendí que por más que se derrumbaban mis castillos o por más que mis vidas se agotaban en un videojuego, siempre habría una segunda oportunidad que me permitiría aprender de mis errores y me ayudaría a levantarme cada vez más fuerte. Mis padres siempre me decían que nada fácil vale la pena y que al caer siempre se aprende algo nuevo que te ayudará a ser mejor; pero que la humildad, la paciencia y los valores humanos siempre están por delante de cada éxito.

El tiempo fue pasando y de la nada me encontraba en el primer establecimiento educativo que he asistido en mi vida, llamado Escuela. Al principio tenía bastante miedo de empezar algo nuevo y sobre todo por tener que separarme de mis padres, sin embargo, supongo que es normal tener miedo a lo desconocido hasta acostumbrarse. Aquí, recuerdo que mis padres me dijeron que tenía que ser fuerte y que con mucha perseverancia el estudio me abriría varias puertas en mi vida. En ese tiempo no había nada más que hubiera deseado que jugar y pasar tiempo de calidad con mis padres. No obstante, tenía tanta curiosidad de saber a qué se referían mis abuelos cuando hablaban de que el conocimiento y la

experiencia son la mejor arma ante cualquier situación difícil, que decidí seguir adelante y adentrarme en lo desconocido.

Sin embargo, el significado de “Conocimiento” se hizo un poco más entendible para mí cuando entré a mi segundo establecimiento educativo, llamado Colegio. Aquí conocí a mi mejor amigo de vida llamado Sebastian F., que siempre me ha dado una mano cuando más lo he necesitado y sobre todo que ha sido mi acolite en varias travesuras y aventuras. Se que siempre podré contar con él y él conmigo. Conjuntamente, fui capaz de conocer a varios profesores (de los cuales mencionaré algunos de sus nombres: Francisco G., Jorge C., Darwin, entre otros) que me permitieron adquirir conocimientos específicos sobre ciencias exactas, ciencias humanas, ciencias sociales, el idioma inglés y que a su vez me permitió reafirmar e incrementar mi interés nato por la “Tecnología”, lo cual a futuro me llevaría a decidir mi carrera universitaria de Ingeniería de Sistemas y Computación en la PUCE.

Posteriormente pasó nuevamente el tiempo y llegó el momento, así que decidí aplicar a la carrera de Ingeniería de Sistemas y Computación en la PUCE, en ese momento no se si fue algo que ya estaba escrito como parte de mi destino por Dios o un simple deseo obstinado de conseguir lo que deseaba por más difícil que fuera. El problema fue que solamente apliqué a la PUCE y a su carrera de Ingeniería de Sistemas; es decir, solo tenía una sola alternativa. Por lo tanto, sentía mucho miedo y angustia, pero con el apoyo de todos mis amigos del colegio y de mi familia, además de mi deseo de conseguirlo, estudié y me preparé bastante para lograrlo.

Un día finalmente recibí una respuesta satisfactoria que cambiaría mi vida completamente y era que al fin lo conseguí, entre en la PUCE. Este factor fue relevante en mi vida ya que me permitió superar mi nostalgia de tener que abandonar a mi familia del colegio (Profesores y Amigos) y crecer como todo un profesional para alcanzar mis sueños.

Al principio, fue muy difícil como era de esperarse. Nunca imaginé que el cambio de colegio a la universidad iba a ser tan complicado, pero afortunadamente y con mi firme deseo de alcanzar mis sueños y conseguir mis metas de vida, lo fui consiguiendo de poco a poco. Conjuntamente, al seguir avanzando mis estudios Universitarios, conocí a una persona muy especial para mí en mi vida, aquella chica con mirada dulce, valores de oro, sentimientos puros y con muchos deseos de superación y sueños similares a los míos. En ese momento supe que estaba enamorado de la chica que ahora forma parte de mi vida como mi enamorada, llamada Isabel G.. Juntos hemos conseguido muchas cosas asombrosas en el transcurso de la carrera y hemos vivido aventuras geniales y difíciles. Pero sobre todo hemos reído y llorado sin darnos por vencidos, apoyándonos el uno con el otro sin importar las adversidades.

Definitivamente puedo decir que en la Universidad he aprendido muchas cosas nuevas, he reafirmado mi pasión por la Tecnología, he encontrado un nuevo hogar en el que me siento a gusto adquiriendo conocimiento de temas que me apasionan y he conocido a personas muy importantes que ahora forman parte de mi vida además de mi enamorada. Una de estas personas es mi segundo amigo de vida Francisco U., que me ha ayudado en muchos momentos difíciles y que me ha aconsejado sobre cualquier problema como yo a él, pero sobre todo que nos hemos apoyado mutuamente para ser cada vez mejores. Se que

siempre podré contar con él a nivel profesional y como un amigo sincero. Por otra parte, conseguí también otros amigos (de los cuales mencionaré algunos de sus nombres: Daniel V., Roberto G., Israel M., Alexis C., Gabriel Y., Cristopher C., Alexei R., Cesar A., Mario G., Mario A., Victoria G., Adela O., entre otros) que me han permitido conocerme mejor como persona, ponerme a prueba y mejorar de forma profesional y humana. Simultáneamente, conocí a varios profesores que me han apoyado en mi formación, que me han permitido potenciar mis talentos y que me han guiado para mejorar mis conocimientos técnicos y no técnico (de los cuales mencionaré algunos de sus nombres: Suyana A., Alfredo C., Fabian DLC., Abel C. (QDEP), Xavier C., Francisco R., Miguel O., Oswaldo E., Andrés J., Eddy S., Edisson M., Jorge A., Charles E., Damián N., Beatriz C., Pablo T., Ana L., Kathy A., entre otros).

Y bien, ahora que voy a culminar la etapa de mi vida Universitaria y a empezar una etapa completamente nueva como un Profesional, estoy seguro de que voy a recordar todas las experiencias, a atesorar todos los buenos momentos y recuerdos especiales, a mejorar como persona en todo aspecto, para ser uno de los mejores profesionales que la patria pueda tener, y a luchar por alcanzar mis nuevas metas futuras para poder cumplir mis sueños; y a futuro ser un padre ejemplar que pueda dar una vida feliz a sus hijos y esposa. Llenándolos de valores como lo hicieron mis padres y mis abuelos, pero sobre todo transmitiendo mis experiencias de vida y conocimientos para que aspiren grande y puedan alcanzar sus sueños.

José Antonio Salgado Proaño

Prefacio

En la actualidad, la adquisición de conocimiento se ha convertido en una necesidad base para todas las personas; esto se debe al avance exponencial de la ciencia y tecnología que obligan a todos a superarse cada día más. Además, la gran cantidad de información que rodea a la sociedad ha dado como resultado un nuevo concepto denominado: "Sociedad de la Información". La idea de un mundo digitalizado, lleno de nuevas concepciones y tecnologías que requieren una actualización constante de sapiencias por parte de un individuo.

Por lo mencionado, se necesita urgentemente un sistema que permita adquirir conocimiento de forma significativa para los Estudiantes y/o futuros profesionales que realmente quieran dejar su huella en el mundo. Es por esto por lo que, se construirá un sistema capaz de facilitar la adquisición de conocimiento.

El sistema permitirá organizar de mejor manera la información del Estudiante (anotaciones) y del docente (syllabus con Contenidos), debido a que los datos almacenados serán cargados directamente en una plataforma digital. Dicha plataforma permitirá acceder a toda la información orientada en

un solo lugar, en base a unidades, temas y Contenidos de una Asignatura de forma interactiva; evitando el uso de herramientas físicas como cuadernos que limiten su uso por su capacidad (hojas) y por su volatilidad (pérdida por desuso).

El sistema permitirá aprender de manera ordenada sin necesidad de estar buscando en múltiples lugares y/o fuentes; ya que simplemente bastará con utilizar el celular, tablet y/o computador que facilite la accesibilidad a dicho Contenido. Por otra parte,

al Profesor le permitirá organizar los syllabus de las Asignaturas que dicta en cada período, con sus respectivos temas y Contenidos (imágenes, textos, videos, audios y/o notas relevantes) que potencien la creatividad del docente y a la vez que le permitan guiar al Estudiante.

El sistema, permitirá aprovechar la potencialidad de herramientas tecnológicas para mejorar la productividad e interacción de las capacidades adquisitivas de información por parte del Estudiante. Y que a su vez ayudará a los docentes a mejorar la relación con sus alumnos, mejorando algunos aspectos tales como: la mejora de gestión académica y la de competencias personales. Permitiendo así, que las instituciones puedan aprovechar los resultados en exámenes y pruebas (por parte de sus alumnos y docentes) para monitorear el trabajo realizado y alinearlo con sus planes estratégicos.

Además, contará con una funcionalidad para que el Estudiante pueda practicar para pruebas como el “Ser Bachiller” y el de “Fin de Carrera”, o en sí para que pueda repasar y reforzar los conocimientos que desee cuando lo requiera. Esto, facilitará la adquisición de información por parte de los Estudiantes, mediante el uso de tecnologías digitales; usando sus propias percepciones conceptuales que forman parte del conocimiento teórico-práctico postulado por un agente del saber. Es decir, permitir que además de poder revisar su información, le permita tomar anotaciones de cada idea para aprender mejor.

Tabla de Contenidos

Capítulo 1 Fundamento Teórico....1
Metodología de Investigación....1
Diseño Universal....5
Metodología Orientada a Objetos....7
Diagramas de Casos de Uso....10
Diagrama de Actividades....11
Diagrama de Clases....12
Diagrama de Paquetes....12
Diagrama de Secuencia....13
Herramientas de Desarrollo....14
Lenguaje de Programación.....15
Definición de Variables....15
Asignación de Valor a una Variable. 15
Tipos de Variables....16
Objetos. 17
Comentarios. 17
Comentarios de una Línea.....18
Comentarios de Varias Líneas 18
Funciones. 18
Framework. 18
Librerías. 22
JavaScript. 23
HTML. 24
CSS....24
Motor de Base de Datos. 24
Modelo de Cascada Modificado (Sinergy) 25
Análisis....26
Diseño 26
Construcción (Implementación)....27
Pruebas 28
Documentación 28
Implantación....28
Capítulo 2 Análisis y Diseño....30
Fase de Análisis 30
Diagrama Caso de Uso General 30
Diagrama de Actividades 35
F0: Gestionar Inicio de Sesión 36
F15: Gestionar Anotación 37
F19: Gestionar Asignación de Profesores por Asignatura 39
F20: Gestionar Asignación de Estudiantes por Asignatura 40
Diagrama Caso de Uso Siguiente Nivel....41
F0: Gestionar Inicio de Sesión 42

F1: Gestionar Provincia....43
F2: Gestionar Ciudad....43
F3: Gestionar Universidad....45
F4: Gestionar Sede....46
F5: Gestionar Periodo....47
F6: Gestionar Facultad....48
F7: Gestionar Escuela....49
F8: Gestionar Carrera....50
F9: Gestionar Estudiante....51
F10: Gestionar Profesor....52
F11: Gestionar Asignatura....53
F12: Gestionar Unidad de Estudio....54
F13: Gestionar Tema de Estudio....55
F14: Gestionar Contenido....56
F15: Gestionar Anotación....57
F16: Gestionar Evaluación....58
F17: Gestionar Glosario....59
F18: Gestionar Taller....60
F19: Gestionar Asignación de Profesores por Asignatura....61
F20: Gestionar Asignación de Estudiantes por Asignatura....62
F21: Gestionar Asignación de Facultades por Sede....63
F22: Gestionar Estudiante Excel....64
Fase de Diseño....64
Diagramas de Clases....64
Diagrama de Clases General....65
Diagrama de Clases Detallado....67
Diagrama de Paquetes....69
Diagrama de Paquetes por Dependencia....69
Diagrama de Paquetes por Asociación....105
Modelo Entidad-Relación....106
Modelo Conceptual....107
Modelo Físico....110
Modelo Lógico....114
Diagramas de Secuencia....117
F0: Gestionar Inicio de Sesión....117
F1: Gestionar Provincia....120
F2: Gestionar Ciudad....125
F3: Gestionar Universidad....130
F4: Gestionar Sede....136
F5: Gestionar Periodo....141
F6: Gestionar Facultad....146
F7: Gestionar Escuela....151
F8: Gestionar Carrera....156
F9: Gestionar Estudiante....161

F10: Gestionar Profesor 166
F11: Gestionar Asignatura 172
F12: Gestionar Unidad de Estudio 178
F13: Gestionar Tema de Estudio 183
F14: Gestionar Contenido 189
F15: Gestionar Anotación 195
F16: Gestionar Evaluación 199
F17: Gestionar Glosario 205
F18: Gestionar Taller 211
F19: Gestionar Asignación de Profesores por Asignatura 216
F20: Gestionar Asignación de Estudiantes por Asignatura 222
F21: Gestionar Asignación de Facultades por Sede 227
F22: Gestionar Estudiante Excel 231
Capítulo 3 Implementación y Pruebas 232
Implementación 232
Código 232
F3: Módulo Universidades 232
F9: Módulo Gestionar Estudiante 247
F10: Módulo Gestionar Profesor 269
F11: Módulo Gestionar Asignatura 285
F14: Módulo Gestionar Contenido 303
F19: Módulo Gestionar Asignación de Estudiantes por Asignatura 326
Pruebas 347
Pruebas Funcionales 347
F1: Gestionar Provincia 347
F2: Gestionar Ciudad 348
F3: Gestionar Universidad 348
F4: Gestionar Sede 349
F5: Gestionar Periodo 350
F6: Gestionar Facultad 350
F7: Gestionar Escuela 351
F8: Gestionar Carrera 352
F9: Gestionar Estudiante 352
F10: Gestionar Profesor 353
F11: Gestionar Asignatura 353
F12: Gestionar Unidad de Estudio 354
F13: Gestionar Tema de Estudio 355
F14: Gestionar Contenido 355
F15: Gestionar Anotación 356
F16: Gestionar Evaluación 357
F17: Gestionar Glosario 357
F18: Gestionar Taller 358
F19: Gestionar Asignación de Asignaturas por Profesor 358
F20: Gestionar Asignación de Asignaturas por Estudiante 359

F21: Gestionar Asignación de Facultades por Sede....360
F22: Gestionar Estudiante Excel....360
Capítulo 4 Conclusiones y Recomendaciones....362
Conclusiones....362
Recomendaciones....365
Bibliografía....368

Lista de tablas

Tabla 1. Vistas y Diagramas UML....9

Lista de figuras

Figura 1. Ejemplo de Diagrama de Casos de Uso....10
Figura 2. Ejemplo de Diagrama de Actividades....11
Figura 3. Ejemplo de Diagrama de Clases....12
Figura 4. Ejemplo de Diagrama de Paquetes....13
Figura 5. Ejemplo de Diagrama de Secuencia....14
Figura 6. Lista de Comandos Artisan....20
Figura 7. Estructura de Funcionamiento Laravel....22
Figura 8. Sinergy....29
Figura 9. Diagrama de Caso de Uso General....32
Figura 10. Continuación de Diagrama de Caso de Uso General....34
Figura 11. Diagrama de Actividades F0: Gestionar Inicio de Sesión....36
Figura 12. Diagrama de Actividades F15: Gestionar Anotación....38
Figura 13. Diagrama de Actividades F19: Gestionar Asignación de Profesores por Asignatura....39
Figura 14. Diagrama de Actividades F20: Gestionar Asignación de Estudiantes por Asignatura....40
Figura 15. Diagrama C.U. Funcionalidad 0: Gestionar Inicio de Sesión....42
Figura 16. Diagrama C.U. Funcionalidad 1: Gestionar Provincia....43
Figura 17. Diagrama C.U. Funcionalidad 2: Gestionar Ciudad....44
Figura 18. Diagrama C.U. Funcionalidad 3: Gestionar Universidad....45
Figura 19. Diagrama C.U. Funcionalidad 4: Gestionar Sede....46
Figura 20. Diagrama C.U. Funcionalidad 5: Gestionar Periodo....47
Figura 21. Diagrama C.U. Funcionalidad 6: Gestionar Facultad....48
Figura 22. Diagrama C.U. Funcionalidad 7: Gestionar Escuela....49
Figura 23. Diagrama C.U. Funcionalidad 8: Gestionar Carrera....50
Figura 24. Diagrama C.U. Funcionalidad 9: Gestionar Estudiante....51

Figura 25. Diagrama C.U. Funcionalidad 10: Gestionar Profesor 52
Figura 26. Diagrama C.U. Funcionalidad 11: Gestionar Asignatura 53
Figura 27. Diagrama C.U. Funcionalidad 12: Gestionar Unidad de Estudio 54
Figura 28. Diagrama C.U. Funcionalidad 13: Gestionar Tema de Estudio 55
Figura 29. Diagrama C.U. Funcionalidad 14: Gestionar Contenido 56
Figura 30. Diagrama C.U. Funcionalidad 15: Gestionar Anotación 57
Figura 31. Diagrama C.U. Funcionalidad 16: Gestionar Evaluación 58
Figura 32. Diagrama C.U. Funcionalidad 17: Gestionar Glosario 59
Figura 33. Diagrama C.U. Funcionalidad 18: Gestionar Taller 60
Figura 34. Diagrama C.U. Funcionalidad 19: Gestionar Asignación de 61
Figura 35. Diagrama C.U. Funcionalidad 20: Gestionar Asignación de 62
Figura 36. Diagrama C.U. Funcionalidad 21: Gestionar Asignación de 63
Figura 37. Diagrama C.U. Funcionalidad 22: Gestionar Estudiante Excel 64
Figura 38. Diagrama de Clases General 65
Figura 39. Diagrama de Clases Detallado 67
Figura 40. Diagrama de Paquetes por Dependencia General 1 69
Figura 41. Diagrama de Paquetes por Dependencia General 2 70
Figura 42. Diagrama de Paquetes por Dependencia General con Detalle de Subcarpetas 71
Figura 43. Diagrama de Paquetes por Dependencia Anotación Detallado 72
Figura 44. Diagrama de Paquetes por Dependencia Asignatura Detallado 73
Figura 45. Diagrama de Paquetes por Dependencia AsignaturaxEstudiante Detallado 75
Figura 46. Diagrama de Paquetes por Dependencia AsignaturaxProfesor Detallado 77
Figura 47. Diagrama de Paquetes por Dependencia Carrera Detallado 78
Figura 48. Diagrama de Paquetes por Dependencia Ciudad Detallado 80
Figura 49. Diagrama de Paquetes por Dependencia Contenido Detallado 81
Figura 50. Diagrama de Paquetes por Dependencia Escuela Detallado 83
Figura 51. Diagrama de Paquetes por Dependencia Estudiante y EstudianteExcel Detallados 84
Figura 52. Diagrama de Paquetes por Dependencia Evaluación Detallado 86
Figura 53. Diagrama de Paquetes por Dependencia Facultad Detallado 87
Figura 54. Diagrama de Paquetes por Dependencia FacultadxSede Detallado 89
Figura 55. Diagrama de Paquetes por Dependencia Glosario Detallado 90
Figura 56. Diagrama de Paquetes por Dependencia Periodo Detallado 92
Figura 57. Diagrama de Paquetes por Dependencia Profesor Detallado 93
Figura 58. Diagrama de Paquetes por Dependencia Provincia Detallado 95
Figura 59. Diagrama de Paquetes por Dependencia Sede Detallado 96
Figura 60. Diagrama de Paquetes por Dependencia Taller Detallado 98
Figura 61. Diagrama de Paquetes por Dependencia Tema Detallado 99
Figura 62. Diagrama de Paquetes por Dependencia Unidad Detallado 101
Figura 63. Diagrama de Paquetes por Dependencia Universidad Detallado 102
Figura 64. Diagrama de Paquetes por Asociación 105
Figura 65. Modelo Conceptual de la Base de Datos del Sistema de Agenda Electrónica para el Estudiante 108

Figura 66. Modelo Físico de la Base de Datos del Sistema de Agenda Electrónica para el Estudiante 111
Figura 67. Modelo Lógico de la Base de Datos del Sistema de Agenda Electrónica para el Estudiante 115
Figura 68. Diagrama Secuencia F0.1: Gestionar Inicio de Sesión 117
Figura 69. Diagrama Secuencia F0.2: Gestionar Cambiar Contraseña 118
Figura 70. Diagrama Secuencia F1.1: Ingresar Provincia 120
Figura 71. Diagrama Secuencia F1.2: Modificar Provincia 121
Figura 72. Diagrama Secuencia F1.3: Eliminar Provincia 122
Figura 73. Diagrama Secuencia F1.4: Consultar Provincia General 123
Figura 74. Diagrama Secuencia F1.4.1: Consultar Provincia por Parámetros 124
Figura 75. Diagrama Secuencia F2.1: Ingresar Ciudad 125
Figura 76. Diagrama Secuencia F2.2: Modificar Ciudad 126
Figura 77. Diagrama Secuencia F2.3: Eliminar Ciudad 127
Figura 78. Diagrama Secuencia F2.4: Consultar Ciudad General 128
Figura 79. Diagrama Secuencia F2.4.1: Consultar Ciudad por Parámetros 129
Figura 80. Diagrama Secuencia F3.1: Ingresar Universidad 130
Figura 81. Diagrama Secuencia F3.2: Modificar Universidad 132
Figura 82. Diagrama Secuencia F3.3: Eliminar Universidad 133
Figura 83. Diagrama Secuencia F3.4: Consultar Universidad General 134
Figura 84. Diagrama Secuencia F3.4.1: Consultar Universidad por Parámetros 135
Figura 85. Diagrama Secuencia F4.1: Ingresar Sede 136
Figura 86. Diagrama Secuencia F4.2: Modificar Sede 137
Figura 87. Diagrama Secuencia F4.3: Eliminar Sede 138
Figura 88. Diagrama Secuencia F4.4: Consultar Sede General 139
Figura 89. Diagrama Secuencia F4.4.1: Consultar Sede por Parámetros 140
Figura 90. Diagrama Secuencia F5.1: Ingresar Periodo 141
Figura 91. Diagrama Secuencia F5.2: Modificar Periodo 142
Figura 92. Diagrama Secuencia F5.3: Eliminar Periodo 143
Figura 93. Diagrama Secuencia F5.4: Consultar Periodo General 144
Figura 94. Diagrama Secuencia F5.4.1: Consultar Periodo por Parámetros 145
Figura 95. Diagrama Secuencia F6.1: Ingresar Facultad 146
Figura 96. Diagrama Secuencia F6.2: Modificar Facultad 147
Figura 97. Diagrama Secuencia F6.3: Eliminar Facultad 148
Figura 98. Diagrama Secuencia F6.4: Consultar Facultad General 149
Figura 99. Diagrama Secuencia F6.4.1: Consultar Facultad por Parámetros 150
Figura 100. Diagrama Secuencia F7.1: Ingresar Escuela 151
Figura 101. Diagrama Secuencia F7.2: Modificar Escuela 152
Figura 102. Diagrama Secuencia F7.3: Eliminar Escuela 153
Figura 103. Diagrama Secuencia F7.4: Consultar Escuela General 154
Figura 104. Diagrama Secuencia F7.4.1: Consultar Escuela por Parámetros 155
Figura 105. Diagrama Secuencia F8.1: Ingresar Carrera 156
Figura 106. Diagrama Secuencia F8.2: Modificar Carrera 157
Figura 107. Diagrama Secuencia F8.3: Eliminar Carrera 158

Figura 108. Diagrama Secuencia F8.4: Consultar Carrera General 159
Figura 109. Diagrama Secuencia F8.4.1: Consultar Carrera por Parámetros 160
Figura 110. Diagrama Secuencia F9.1: Ingresar Estudiante 161
Figura 111. Diagrama Secuencia F9.2: Modificar Estudiante 162
Figura 112. Diagrama Secuencia F9.3: Eliminar Estudiante 163
Figura 113. Diagrama Secuencia F9.4: Consultar Estudiante General 164
Figura 114. Diagrama Secuencia F9.4: Consultar Estudiante General 165
Figura 115. Diagrama Secuencia F10.1: Ingresar Profesor 166
Figura 116. Diagrama Secuencia F10.2: Modificar Profesor 168
Figura 117. Diagrama Secuencia F10.3: Eliminar Profesor 169
Figura 118. Diagrama Secuencia F10.4: Consultar Profesor General 170
Figura 119. Diagrama Secuencia F10.4.1: Consultar Profesor por Parámetros 171
Figura 120. Diagrama Secuencia F11.1: Ingresar Asignatura 172
Figura 121. Diagrama Secuencia F11.2: Modificar Asignatura 174
Figura 122. Diagrama Secuencia F11.3: Eliminar Asignatura 175
Figura 123. Diagrama Secuencia F11.4: Consultar Asignatura General 176
Figura 124. Diagrama Secuencia F11.4.1: Consultar Asignatura por Parámetros 177
Figura 125. Diagrama Secuencia F12.1: Ingresar Unidad de Estudio 178
Figura 126. Diagrama Secuencia F12.2: Modificar Unidad de Estudio 180
Figura 127. Diagrama Secuencia F12.3: Eliminar Unidad de Estudio 180
Figura 128. Diagrama Secuencia F12.4: Consultar Unidad de Estudio General 181
Figura 129. Diagrama Secuencia F12.4.1: Consultar Unidad de Estudio por Parámetros 182
Figura 130. Diagrama Secuencia F13.1: Ingresar Tema de Estudio 183
Figura 131. Diagrama Secuencia F13.2: Modificar Tema de Estudio 185
Figura 132. Diagrama Secuencia F13.3: Eliminar Tema de Estudio 186
Figura 133. Diagrama Secuencia F13.4: Consultar Tema de Estudio General 187
Figura 134. Diagrama Secuencia F13.4.1: Consultar Tema de Estudio por Parámetros 188
Figura 135. Diagrama Secuencia F14.1: Ingresar Contenido 189
Figura 136. Diagrama Secuencia F14.2: Modificar Contenido 191
Figura 137. Diagrama Secuencia F14.3: Eliminar Contenido 192
Figura 138. Diagrama Secuencia F14.4: Consultar Contenido General 193
Figura 139. Diagrama Secuencia F14.4.1: Consultar Contenido por Parámetros 194
Figura 140. Diagrama Secuencia F15.1: Ingresar o Modificar Anotación 195
Figura 141. Diagrama Secuencia F15.2: Consultar Anotación por Parámetros 197
Figura 142. Diagrama Secuencia F16.1: Ingresar Evaluación 199
Figura 143. Diagrama Secuencia F16.2: Modificar Evaluación 201
Figura 144. Diagrama Secuencia F16.3: Eliminar Evaluación 202
Figura 145. Diagrama Secuencia F16.4: Consultar Evaluación General 203
Figura 146. Diagrama Secuencia F16.4.1: Consultar Evaluación por Parámetros 204
Figura 147. Diagrama Secuencia F17.1: Ingresar Glosario 205
Figura 148. Diagrama Secuencia F17.2: Modificar Glosario 207
Figura 149. Diagrama Secuencia F17.3: Eliminar Glosario 208

Figura 150. Diagrama Secuencia F17.4: Consultar Glosario General 209
Figura 151. Diagrama Secuencia F17.4.1: Consultar Glosario por Parámetros 210
Figura 152. Diagrama Secuencia F18.1: Ingresar Taller 211
Figura 153. Diagrama Secuencia F18.2: Modificar Taller 213
Figura 154. Diagrama Secuencia F18.3: Eliminar Taller 213
Figura 155. Diagrama Secuencia F18.4: Consultar Taller General 214
Figura 156. Diagrama Secuencia F18.4.1: Consultar Taller por Parámetros 215
Figura 157. Diagrama Secuencia F19.1: Ingresar Asignación de Profesores por Asignatura 217
Figura 158. Diagrama Secuencia F19.2: Modificar Asignación de Profesores por Asignatura 218
Figura 159. Diagrama Secuencia F19.3: Eliminar Asignación de Profesores por Asignatura 219
Figura 160. Diagrama Secuencia F19.4: Consultar Asignación de Profesores por Asignatura General 220
Figura 161. Diagrama Secuencia F19.4.1: Consultar Asignación de Profesores por Asignatura por Parámetros 221
Figura 162. Diagrama Secuencia F20.1: Ingresar Asignación de Estudiantes por Asignatura 222
Figura 163. Diagrama Secuencia F20.2: Modificar Asignación de Estudiantes por Asignatura 224
Figura 164. Diagrama Secuencia F20.3: Eliminar Asignación de Estudiantes por Asignatura 224
Figura 165. Diagrama Secuencia F20.4: Consultar Asignación de Estudiantes por Asignatura General 225
Figura 166. Diagrama Secuencia F20.4.1: Consultar Asignación de Estudiantes por Asignatura por Parámetros 226
Figura 167. Diagrama Secuencia F21.1: Ingresar Asignación de Facultades por Sede 227
Figura 168. Diagrama Secuencia F21.2: Eliminar Asignación de Facultades por Sede 228
Figura 169. Diagrama Secuencia F21.3: Consultar Asignación de Facultades por Sede General 229
Figura 170. Diagrama Secuencia F21.3.1: Consultar Asignación de Facultades por Sede por Parámetros 230
Figura 171. Diagrama Secuencia F22.1: Ingresar Estudiante 231
Figura 172. Método Index Universidades 233
Figura 173. Método Create Universidades 233
Figura 174. Método Store Universidades 234
Figura 175. Método Destroy Universidades 235
Figura 176. Método Edit Universidades 235
Figura 177. Método Update Universidades 236
Figura 178. Universidades Model 237
Figura 179. Universidades Route Path 237
Figura 180. Universidades Blade 1 238
Figura 181. Universidades Blade 2 238

Figura 182. Universidades Blade 3 239
Figura 183. Universidades Create Blade 1 239
Figura 184. Universidades Create Blade 2 240
Figura 185. Universidades Edit Blade 1 241
Figura 186. Universidades Edit Blade 2 241
Figura 187. Universidades Blade – Eliminar Registro 242
Figura 188. Página de Universidades Principal 243
Figura 189. Página de Universidades Crear 244
Figura 190. Página de Universidades Editar 245
Figura 191. Método Index Estudiantes 248
Figura 192. Método EstudianteImport Estudiantes 1 248
Figura 193. Método EstudianteImport Estudiantes 2 249
Figura 194. Método Create Estudiantes 250
Figura 195. Método Create Estudiantes Excel 250
Figura 196. Método Store Estudiantes 251
Figura 197. Método Destroy Estudiantes 252
Figura 198. Método Edit Estudiantes 253
Figura 199. Método Update Estudiantes 253
Figura 200. Estudiantes Model 254
Figura 201. Estudiantes Formularios Route Path 255
Figura 202. Estudiantes en Excel Route Path 255
Figura 203. Estudiantes Blade 1 256
Figura 204. Estudiantes Blade 2 256
Figura 205. Estudiantes Blade 3 257
Figura 206. Estudiantes Create Blade 1 257
Figura 207. Estudiantes Create Blade 2 258
Figura 208. Estudiantes Create Excel Blade 1 259
Figura 209. Estudiantes Create Excel Blade 2 259
Figura 210. Estudiantes Edit Blade 1 260
Figura 211. Estudiantes Edit Blade 2 260
Figura 212. Estudiantes Blade – Eliminar Registro 261
Figura 213. Página de Estudiantes Principal 262
Figura 214. Página de Estudiantes Crear 263
Figura 215. Página de Estudiantes Excel Crear 264
Figura 216. Formato de Ejemplo de Documento de Excel para Agregar Estudiantes 264
Figura 217. Página de Estudiantes Editar 265
Figura 218. Comprobación Formato de Cédula 266
Figura 219. Método Index Profesores 271
Figura 220. Método Create Profesores 271
Figura 221. Método Store Profesores 272
Figura 222. Método Destroy Profesores 273
Figura 223. Método Edit Profesores 273
Figura 224. Método Update Profesores 274
Figura 225. Profesores Model 275

Figura 226. Profesores Formularios Route Path 275
Figura 227. Profesores Blade 1 276
Figura 228. Profesores Blade 2 276
Figura 229. Estudiantes Blade 3 277
Figura 230. Profesores Create Blade 1 278
Figura 231. Profesores Create Blade 2 278
Figura 232. Profesores Edit Blade 1 279
Figura 233. Profesores Edit Blade 2 279
Figura 234. Profesores Blade – Eliminar Registro 280
Figura 235. Página de Profesores Principal 281
Figura 236. Página de Profesores Crear 282
Figura 237. Página de Profesores Editar 283
Figura 238. Comprobación Formato de Cédula 284
Figura 239. Método Index Asignaturas 286
Figura 240. Método Create Asignaturas 287
Figura 241. Método Store Asignaturas 288
Figura 242. Método Destroy Asignaturas 289
Figura 243. Método Edit Asignaturas 289
Figura 244. Método Update Asignaturas 290
Figura 245. Asignaturas Model 291
Figura 246. Asignaturas Formularios Route Path 291
Figura 247. Asignaturas Blade 1 292
Figura 248. Asignaturas Blade 2 292
Figura 249. Asignaturas Blade 3 293
Figura 250. Asignaturas Create Blade 1 294
Figura 251. Asignaturas Create Blade 2 294
Figura 252. Asignaturas Edit Blade 1 295
Figura 253. Asignaturas Edit Blade 2 296
Figura 254. Asignaturas Blade – Eliminar Registro 297
Figura 255. Página de Asignaturas Principal 298
Figura 256. Página de Asignaturas Crear 299
Figura 257. Página de Asignaturas Editar 300
Figura 258. Método Index Contenido 304
Figura 259. Método Create Contenido 304
Figura 260. Método Store Contenido 307
Figura 261. Método Destroy Contenido 308
Figura 262. Método Edit Contenido 309
Figura 263. Método Update Contenido 311
Figura 264. Contenidos Model 312
Figura 265. Contenidos Formularios Route Path 312
Figura 266. Contenidos Blade 1 313
Figura 267. Contenidos Blade 2 313
Figura 268. Contenidos Blade 3 314
Figura 269. Contenidos Create Blade 1 315

Figura 270. Contenidos Create Blade 2 316
Figura 271. Contenidos Edit Blade 1 317
Figura 272. Contenidos Edit Blade 2 318
Figura 273. Contenidos Blade – Eliminar Registro 319
Figura 274. Página de Contenidos Principal 320
Figura 275. Página de Contenidos Crear.....321
Figura 276. Página de Contenidos Editar.....321
Figura 277. Método Index Asignación de Estudiantes por Asignatura 327
Figura 278. Método Create Asignación de Estudiantes por Asignatura 328
Figura 279. Método Store Asignación de Estudiantes por Asignatura 329
Figura 280. Método Destroy Asignación de Estudiantes por Asignatura.....330
Figura 281. Método Edit Asignación de Estudiantes por Asignatura.....331
Figura 282. Método Update Asignación de Estudiantes por Asignatura 333
Figura 283. Asignación de Estudiantes por Asignatura Model 333
Figura 284. Asignación de Estudiantes por Asignatura Formularios Route Path.....334
Figura 285. Asignación de Estudiantes por Asignatura Blade 1.....334
Figura 286. Asignación de Estudiantes por Asignatura Blade 2.....335
Figura 287. Asignación de Estudiantes por Asignatura Blade 3.....335
Figura 288. Asignación de Estudiantes por Asignatura Create Blade 1 337
Figura 289. Asignación de Estudiantes por Asignatura Create Blade 2 338
Figura 290. Asignación de Estudiantes por Asignatura Edit Blade 1 340
Figura 291. Asignación de Estudiantes por Asignatura Edit Blade 2 341
Figura 292. Asignación de Estudiantes por Asignatura Blade – Eliminar Registro 342
Figura 293. Página de Asignación de Estudiantes por Asignatura Principal 343
Figura 294. Página de Asignación de Estudiantes por Asignatura Crear 344
Figura 295. Página de Asignación de Estudiantes por Asignatura Editar.....345

Capítulo 1

Fundamento Teórico

Metodología de Investigación

La Metodología de Investigación que se va a utilizar es la aplicada, y consiste en un conjunto de procedimientos que permiten ordenar la información de tal forma que ayuda a resolver un problema, buscando una o varias soluciones inmediatas para el mismo.

> Según Lozada (2014): "La investigación aplicada busca la generación de conocimiento con aplicación directa a los problemas de la sociedad o el sector productivo". Por lo que, dentro del campo tecnológico, es utilizada para obtener productos libres de defectos.

A partir de lo mencionado se puede analizar en el siguiente ejemplo: En la actualidad, la información personal que se almacena dentro de los ordenadores llega a convertirse en un recurso invaluable y que normalmente contiene datos sensibles como contraseñas, fotos privadas, conversaciones delicadas, entre otras cosas. Ahora bien, qué sucede si dicha información es vulnerada por un tercero y, por ende, tanto la integridad como la seguridad de la persona se ve afectada. Es aquí, donde entra la investigación aplicada para hallar una o varias soluciones para dicho problema.

Entonces, al aplicar la tecnología se puede encontrar varias soluciones, que, en el caso del ejemplo anterior, sería utilizar un software capaz de garantizar seguridad conocido como antivirus. Sin embargo, se pueden utilizar otras formas de proteger la información, como: algoritmos de encriptación, firewalls físicos, configuración de una pasarela de red, etc.

Ahora bien y en base a lo mencionado, la tesis tiene como objetivo resolver los problemas presentes en el campo educativo, que dificultan la forma en la que los Estudiantes (bachilleres y universitarios) adquieren conocimiento. Entre estos problemas se encuentran los siguientes:

1. La desorganización característica de los Estudiantes en el momento de tomar apuntes, lo cual suele realizarse en múltiples cuadernos, hojas sueltas y/o dispositivos. Esto ocasiona que la información se encuentre en desorden y posteriormente se extravíe por la pérdida del cuaderno o dispositivo, que no se la vuelva a utilizar debido a que es inentendible y carente de contexto (es decir, no existe fundamento teórico que la respalde, solo son ideas sueltas), y/o simplemente que no se la encuentre cuando se la busca porque no se recuerda en donde se la anotó.
2. El desinterés de los docentes con respecto a la adquisición de conocimiento por parte de sus Estudiantes genera resultados negativos en su rendimiento. Se cree que la relación didáctica que tiene el docente con el Estudiante motiva a este a mejorar constantemente y, por lo tanto, a producir mejores resultados.

> En el rendimiento académico intervienen factores como el nivel intelectual, la personalidad, la motivación, las aptitudes, los intereses, los hábitos de estudio, la autoestima o la relación Profesor-alumno; cuando se produce un desfase entre el rendimiento académico y el rendimiento que se espera del alumno, se habla de rendimiento discrepante; un rendimiento académico insatisfactorio es aquel que se sitúa por debajo del rendimiento esperado. En ocasiones puede estar relacionado con los métodos didácticos. (Marti, 2003, p. 376)

3. La competencia laboral y profesional se encuentra en incremento y por lo tanto genera presión en los Estudiantes para cada vez ser mejores. Sin embargo, sin tener un fundamento teórico claro y por supuesto una experiencia alta, la preocupación aumenta.

> En esta sociedad los cambios no son inevitables, sino necesarios y se producen cada vez con mayor rapidez, reinventando todo: las fronteras, las empresas, la educación, etcétera. Por lo que se observa, la innovación va más allá que sólo producir nuevos productos y servicios, sino que está modificando la forma de producir, cambiando los procesos o métodos utilizados por diversos agentes sociales. Por ejemplo, al innovar en un producto se mejora sus características, ya sean técnicas o funcionales, esto implica cambio desde el material con el que está elaborado, hasta el proceso de manufactura utilizado para su creación. (Castro y Fernández, 2013)

Por lo tanto, el mercado laboral en la actualidad requiere capital humano con nuevas competencias desarrolladas en las instituciones de educación superior, capaces de satisfacer la demanda de innovación generada por nuevas formas de elaborar productos y servicios.

> Desde el punto de vista social, no basta que las instituciones educativas egresen profesionistas al mercado laboral con cierta regularidad, sino que proporcionen una educación eficiente, fundada en el análisis del mercado laboral donde los egresados aplicarán los conocimientos y habilidades recibidas durante su formación profesional, con el fin de proporcionarles

> oportunidades de obtener mayor remuneración, con posibilidades de transformar su entorno y su país. (Hernández, s.f.)

En base a lo mencionado, los Estudiantes optan por incrementar su rendimiento y conocimiento lo más rápido posible; buscando diversas guías de terceros que a más de ayudarlos para los exámenes "Ser Bachiller" y de "Fin de Carrera", les permitan poner a prueba sus conocimientos y así garantizar una mejor oportunidad de trabajo.

4. La desorganización y poca creatividad de visualización de los Contenidos del syllabus presentado por los Profesores (o que simplemente no se presentan) en la mayoría de los casos genera confusión y desinterés por parte de los Estudiantes. Aquí cada individuo intenta buscar una manera de adelantar y/o repasar Contenidos que no se encuentran muy claros o que son inentendibles. Sin embargo, sin una guía adecuada que detalle los temas y su interrelación, se vuelve completamente imposible.

   > El propósito del Syllabus es la formación profesional, es decir que considera como punto central el aprendizaje de los futuros profesionales. El aprendizaje propuesto en el syllabus se enmarca en el campo de estudio y objetivos de la Carrera y aprendizaje estudiantil relacionado con el conocimiento, habilidades y desempeños profesionales, valores, actitudes y emociones. (Zorrilla Gamboa & Garzón Luces, 2016)

5. La obsolescencia de los medios físicos, tales como los cuadernos, que con el pasar del tiempo se descomponen y por lo tanto empiezan a borrar los apuntes de los Estudiantes escritos dentro de estos. Además de la poca disponibilidad que poseen

> los mismos, ya que no se los puede llevar en todo momento y a todo lugar con nosotros. En comparación de los dispositivos tecnológicos que proveen un concepto de alta disponibilidad (24 horas los 7 días de la semana) y una tasa de desgaste menor (sin tomar en cuenta que se puede respaldar la información de forma digital). Según Viñas (2013): "Es fácil perder un papel o clasificarlo en el lugar equivocado. En tu ordenador los datos pueden recuperarse con mayor facilidad y además siempre tienes la opción de almacenar o tener una copia de seguridad en la nube".

En resumen, el sistema que se va a desarrollar al utilizar metodología de investigación aplicada pretende combatir los diversos problemas que existen en el ámbito de la adquisición de conocimientos por parte de los Estudiantes y pretende mejorar la metodología de enseñanza que en algunos casos es rígida por parte de los Profesores. Cabe recalcar que la metodología aplicada se la va a emplear en el momento de pasar las necesidades de los Estudiantes y Profesores en requerimientos funcionales del sistema. Esto se aplica dentro de la fase de análisis del Modelo de Cascada Modificado.

Diseño Universal

El Diseño Universal corresponde a un concepto de diseño en varias áreas del conocimiento que facilita la accesibilidad de un producto o servicio. La idea es enfocar el uso de lo que se ofrece al diferente tipo de personas con distintas características y habilidades que existen en el mundo, lo que se quiere alcanzar es un "diseño pensado para todos". Según Ronald Mace (1941-1998):

Las cosas que la mayoría de la población puede utilizar con independencia de su habilidad o discapacidad se pueden considerar universalmente utilizables. El diseño universal guía el alcance de la accesibilidad y sugiere hacer todos los elementos y espacios accesibles y utilizables por toda la gente, hasta el máximo grado posible.

Al incorporar en el diseño de objetos y espacios construidos, las características necesarias para la gente con limitaciones, podemos hacerlas más seguras y fáciles de usar para todos y, así, más comercial y rentable. El enfoque de diseño universal va más allá que los requisitos mínimos de la ley de accesibilidad.

En base a lo mencionado, se puede rescatar los 7 principios del diseño universal:

1. **Uso Equiparable:** Fundamente que el diseño es atractivo y útil para ser usado para ser usado por varias personas.
2. **Flexibilidad de Uso**: Precisa que el diseño se acomoda a un amplio rango de preferencias y habilidades de uso, por ejemplo: se puede usar tanto con la mano izquierda como con la mano derecha.
3. **Uso Simple e Intuitivo**: Describe que el diseño es fácil de entender y, por lo tanto, es fácil de usar. No se necesita asistir a capacitaciones sobre su uso.
4. **Información Fácilmente Perceptible**: Transmite un diseño que comunica de manera eficaz y con diferente material interactivo como imágenes, videos, etc., un contraste a la información que accede un Estudiante con sus sentidos.
5. **Tolerancia para el Error o Mal Uso:** El diseño reduce los riesgos y consecuencias involuntarias a los errores cometidos por el usuario y dispone de medios para mitigar los mismos.

6. **Poco Esfuerzo Físico:** El diseño puede ser usado de forma confortable para reducir el cansancio del usuario, incentivando su uso.
7. **Tamaño y Espacio Suficiente para el Acercamiento, la Manipulación y el Uso: El** diseño proporciona distintas facilidades, como el tamaño adecuado, la posición adecuada de elementos, etc., que faciliten el acceso, alcance, manipulación y uso por parte del usuario.

La idea de aplicar el Diseño Universal en la construcción del Sistema de Agenda Electrónica del Estudiante permitirá convertirlo en una herramienta básica que sea accesible por cualquier Estudiante y Profesor; sin importar cualquier discapacidad física o mental que este posea y que respete las leyes o principios especificados anteriormente.

Metodología Orientada a Objetos

La Metodología Orientada a Objetos es un proceso que permite modelar un sistema mediante la concatenación de un conjunto de objetos, los cuales tienen un mismo objetivo. El modelado permite tener una mayor comprensión de la funcionalidad del sistema, evitando confusiones al momento de desarrollar el mismo durante la fase de Implementación. Esto implica ciertas ventajas como: una mayor calidad del producto (menor número de defectos), mantenimiento fácil del sistema, además de que es una gran forma de comunicación con el cliente, entre otras cosas.

Quintanilla et al. (2015) afirma: “Una Metodología Orientada a Objetos es un proceso para producir software de una manera organizada (usando convenciones) y técnicas de notación predefinidas”.

Normalmente, se utiliza el UML o Lenguaje Unificado de Modelado, con el fin de modelar, es decir representar visualmente, especificar y documentar los diferentes módulos de un sistema Orientado a Objetos de forma gráfica. Provee especificaciones de los atributos y el comportamiento de las clases con respecto a otras.

> Un segmento importante de la comunidad de Ingeniería de Software ha adoptado el Lenguaje de Unificado de Modelado (UML, por sus siglas en inglés) como el método preferido para representar modelos del análisis y del diseño. El proceso unificado fue desarrollado para proveer una estructura para la aplicación del UML. (Pressman, 2010)

El modelar permite tener una base para posteriormente construir el sistema, permitiendo su comprensión de forma sencilla para los demás Ingenieros de Sistemas, Programadores y principalmente para el cliente. A continuación, se puede visualizar una tabla que indica todos los tipos de Diagramas UML que existen:

Tabla 1. Vistas y Diagramas UML

Área principal	*Vista*	*Diagrama*	*Conceptos principales*
estructural	vista estática	Diagrama de clases	asociación, clase, dependencia, generalización, interfaz, realización
	vista de diseño	estructura interna	conector, interfaz, interfaz obligatoria, interfaz proporcionada, parte, puerto
		Diagrama de colaboración	colaboración, conector, rol, uso de la colaboración
		Diagrama de componentes	componente, dependencia, interfaz proporcionada, interfaz obligatoria, puerto, realización, subsistema
	vista de casos de uso	Diagrama de casos de uso	actor, asociación caso de uso, extensión, generalización de casos de uso, inclusión
dinámica	vista de máquina de estado	Diagrama de máquina de estados	actividad hacer, disparador, efecto, estado, evento, región, transición, transición de finalización
	vista actividad	Diagrama de actividad	acción, actividad, control de flujo, división, excepción, flujo de datos, nodo de control, nodo objeto, pin, región de expansión, unión
	vista de interacción	Diagrama de secuencia	especificación de la ejecución, especificación del suceso, fragmento de la interacción, interacción, línea de vida, mensaje, operando de la interacción, señal
		Diagrama de comunicación	colaboración, condición de guarda, mensaje, rol, número de secuencia
física	vista de despliegue	Diagrama de despliegue	artefacto, dependencia, manifestación, nodo
gestión del modelo	vista de gestión del proyecto	Diagrama de paquetes	importar, modelo, paquete
	perfil	Diagrama de paquetes	estereotipo, perfil, restricción, valor etiquetado

Fuente: (Rumbaugh, Jacobson, & Booch, 2007)

En el presente trabajo, se modelará durante las fases de Análisis y de Diseño los siguientes tipos de Diagramas UML:

Diagramas de Casos de Uso

Los casos de uso indican qué es lo que realiza el sistema, subsistema, clase o interfaz, pero no especifica el cómo lo realiza. Se elaboran escenarios en los cuales se simula la interacción que tendrá el sistema con los usuarios en alguna situación específica.

En este tipo de Diagramas participan varios elementos como: los actores, los casos de uso que se representan por medio de verbos y sus relaciones.

El actor puede cumplir diversos roles en los escenarios, generalmente representan a los humanos o a un agente automático.

Los creadores del UML, Rumbaugh et al. (2007) afirman: "Un Caso de Uso describe una interacción con actores como una secuencia de mensajes entre el sistema y uno o más actores. El término actor incluye tanto a personas, como a otros sistemas informáticos y procesos".

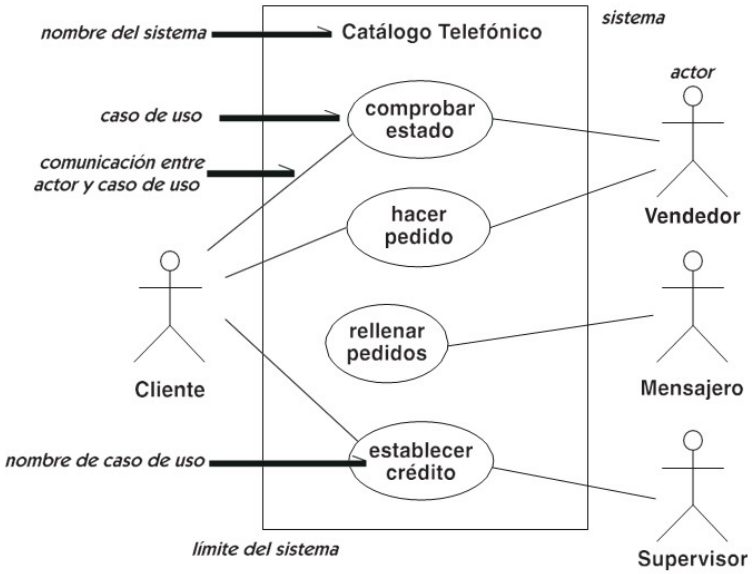


Figura 1. Ejemplo de Diagrama de Casos de Uso

Fuente: (Rumbaugh, Jacobson, & Booch, 2007)

Diagrama de Actividades

Permiten visualizar a través del tiempo el flujo entre las diferentes actividades de un proceso de forma secuencial, indicando paso a paso el orden en el que se realiza cada una.

Los elementos que posee son:

- Las Actividades: Indican la acción que se realiza.
- Los Flujos de Control: Indican el orden en el que se realizan las actividades.
- Los Carriles: Representan a los encargados de realizar determinadas actividades.

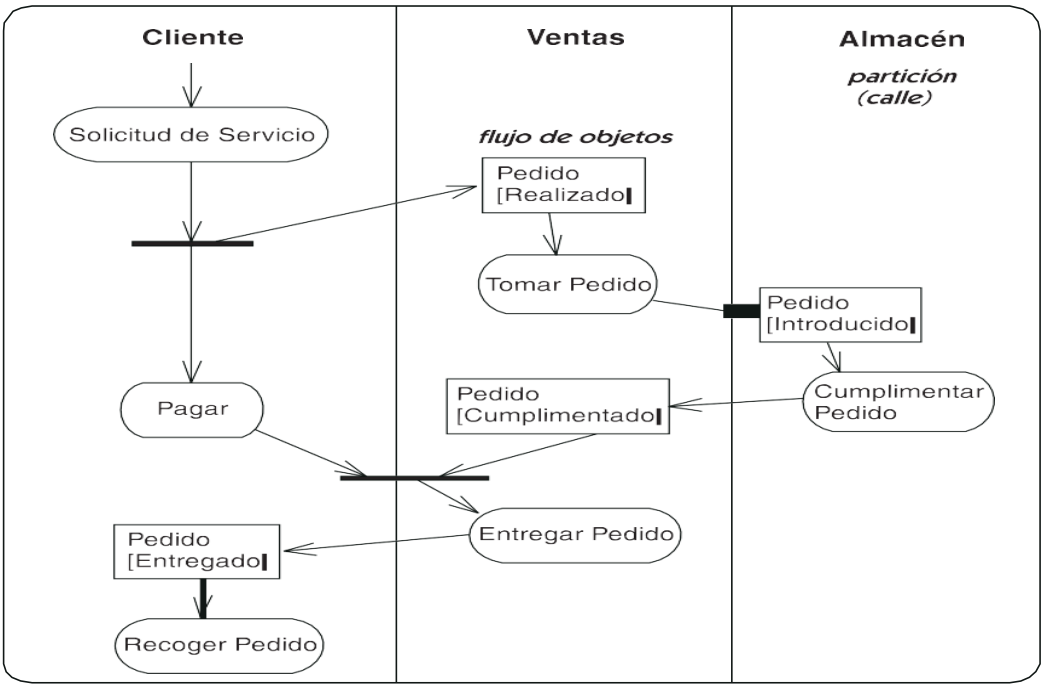


Figura 2. Ejemplo de Diagrama de Actividades

Fuente: (Rumbaugh, Jacobson, & Booch, 2007)

Diagrama de Clases

Permite visualizar todos los objetos que existen dentro del sistema. A cada clase se especifica su nombre, los atributos que posee con su tipo de dato y su valor inicial, la relación que posee con las demás clases y las operaciones que puede realizar.

Los tipos de relaciones que pueden existir entre las diferentes clases son: Asociación unidireccional o bidireccional, Agregación, Composición, Generalización y Dependencia.

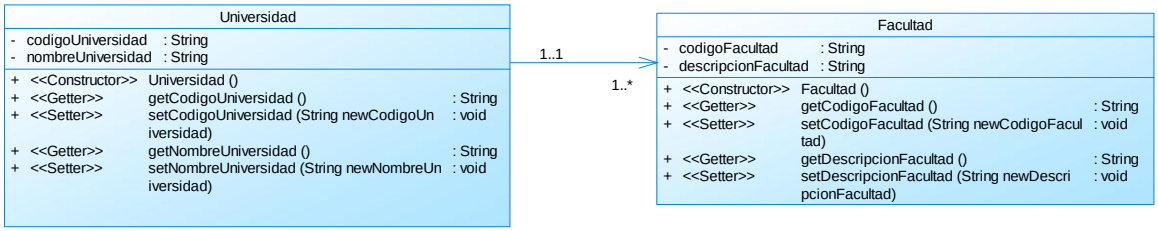


Figura 3. Ejemplo de Diagrama de Clases

Diagrama de Paquetes

Estos Diagramas se presentan con la forma de una carpeta, permiten ordenar y agrupar otros Diagramas UML en grupos. También, proporcionan la visualización del sistema de forma macro, permitiendo ver las dependencias que existen entre los paquetes y se puede colocar generalización de ser necesario.

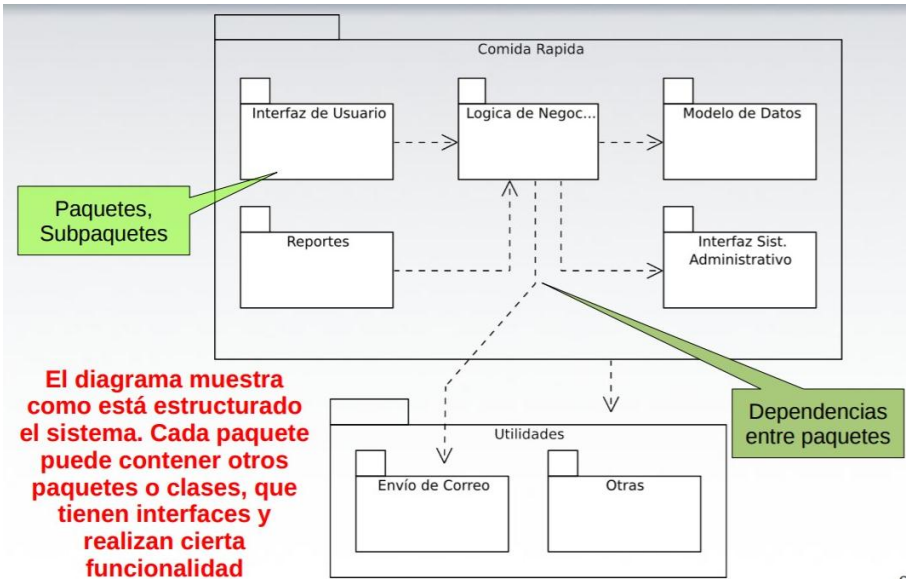


Figura 4. Ejemplo de Diagrama de Paquetes

Fuente: (Gutierrez, 2009)

Diagrama de Secuencia

Este Diagrama muestra de forma ordenada los mensajes que se envían y que se reciben a través del tiempo. Todo proceso inicia con algún mensaje enviado por un algún objeto que inicia la interacción, estos se colocan al extremo izquierdo del Diagrama. Los mensajes se representan con flechas que van de derecha a izquierda, y viceversa, Se utilizan líneas entrecortadas de forma vertical para representar la vida de los objetos y los focos de control sirven para saber la duración que tiene una acción.

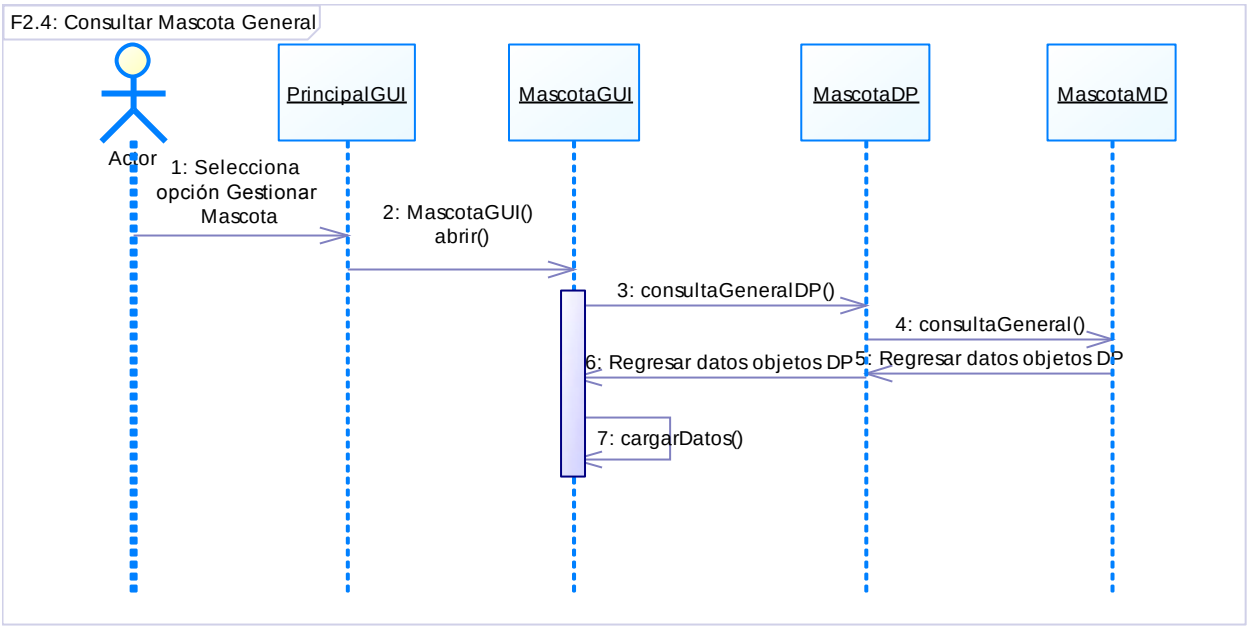


Figura 5. Ejemplo de Diagrama de Secuencia

Herramientas de Desarrollo

Las herramientas de desarrollo son un conjunto de instrumentos que dotarán de vida al sistema. Aquí, se encuentra:

1. El lenguaje de programación que se encarga de traducir la lógica del programador en instrucciones interpretables por un computador.
2. El framework de desarrollo que funciona como gestor para la organización y desarrollo del software.
3. Las librerías proveen un paquete de implementaciones que ofrecen diversas funcionalidades.
4. El motor de base de datos proporciona un almacén para los datos y los archivos del sistema.

Lenguaje de Programación.

El lenguaje que se va a utilizar para construir el sistema es el lenguaje de programación PHP, el cual actualmente es el más popular para desarrollo WEB. Este lenguaje es versátil (es decir, se adapta rápidamente a cualquier Framework de desarrollo), es libre y abierto (no tiene costo al descargar ni al utilizar, además que se puede aportar y descargar aportes de varios desarrolladores), los entornos de desarrollo son fácilmente configurables, tiene una curva de aprendizaje baja, el manejo y conexión a múltiples motores de bases de datos es sencillo, tiene una amplia comunidad de desarrolladores que constantemente crean librerías multipropósito y en el momento de desplegar un proyecto en un servidor de aplicaciones, ofrece una mayor compatibilidad con el mismo.

Es importante saber que el código PHP casi siempre se encuentra embebido dentro del código HTML por lo que hay que recordar utilizar el tag de apertura "<?php" para empezar a escribir código PHP y el tag de finalización "?>" para terminar de escribir código PHP. A continuación, se va a mencionar brevemente cómo funciona el lenguaje de programación PHP:

Definición de Variables.

Se conoce como una variable a un elemento con cierta identidad o nombre asignado por un desarrollador que contiene un valor o dato. En caso de PHP una variable siempre se asigna con el signo de dólar "$". Ejemplo: $variable.

Asignación de Valor a una Variable.

Se conoce como proceso de asignación al momento en el que una variable pasa a tener un valor almacenado dentro de ella. Para guardar el valor dentro de la variable

se precede a la variable con el signo “=” y posteriormente se coloca el dato o valor que se desea almacenar dentro. Ejemplo: $variable = 2, aquí la variable tiene un valor numérico de 2.

Tipos de Variables.

Se dice que una variable adquiere un tipo dependiendo del valor que esta contiene. Existen 5 tipos de variables principales:

Números Enteros (Integer).

Son todos aquellos números enteros que no tienen decimales. Ejemplo: $valor = 1

Números de punto Flotante (Float):

Son todos aquellos números reales que tienen decimales. Ejemplo: $valor = 1.1

Booleanos (Boolean):

Un booleano es aquel valor que indica verdad, es decir verdadero (true) o falso (false). Ejemplo: $valor = True;

Cadenas de Texto (String).

Es todo aquel conjunto de caracteres alfanuméricos que forman una o varias palabras. Sin embargo, al componer una cadena de texto, no se debe olvidar de poner comillas dobles al principio y al fin de la cadena. Ejemplo: $valor = “hola mundo”

Listas de Elementos (Arrays).

Es un conjunto de elementos (que pueden ser de diferentes tipos de variables) agrupados dentro de una variable, y que a su vez forman una lista. Existen diversas sintaxis de creación sin embargo la más utilizada es: “array”

+ “(” + “Contenido 1” + “,” + “…” + “,” + “Contenido n” + “)” + “;”. Ejemplo:

$valor = array(“hola”, 1, “mundo” , 3, “4”);

Objetos.

Un objeto es un elemento miembro de una clase que ha sido instanciado. Por lo tanto, una clase es una particularización de una entidad existente en el mundo real; la cuál al ser renombrada (instanciada) se convierte en un objeto de clase. Ejemplo:

```
<?php
    class ejemplo
    {
        $valorInicial = ””;
        public function _ _construct($valor)
        {
            $this -> $valorInicial = $valor;
        }
    }
    $valorInstanciado = new ejemplo(“hola mundo”); //$valorInstanciado es un
    objeto de la clase ejemplo
?>
```

Comentarios.

Un comentario es una nota que se puede colocar al lado de un fragmento de código. Existen dos tipos de comentarios:

Comentarios de una Línea.

Coloca el comentario en una sola línea de código usando la siguiente sintaxis: "//" + "comentario". Ejemplo: $valor = 1 // este valor es igual a 1

Comentarios de Varias Líneas

Coloca el comentario en múltiples líneas de código usando la siguiente sintaxis: "/*" + "comentario" + "*/". Ejemplo: $valor = 1 /* este valor es igual a 1

Pero puede ser cambiado*/

Funciones.

Una función es un grupo de instrucciones invocables desde un diferente contexto que se estructuran en forma de bloque; y que pueden recibir y/o devolver variables. Ejemplo:

```
function ejemplo ($valor1){//en este caso recibe una variable que se renombrará valor1
    $valorT = $valor1 + 3 ; //dentro de la variable $valorT se almacena el resultado de la operación suma con el Contenido de $valor1 y el número 3
    return $valorT; //se devuelve el resultado de la operación (Contenido de la variable $valorT)
}
```

Framework.

El framework que se va a utilizar se llama Laravel y se encarga de suministrar las herramientas necesarias para el desarrollo WEB, creando una estructura organizada por carpetas y ficheros para ordenar el código y evitar lo que se conoce como "código espagueti".

> Laravel es accesible, pero potente, y proporciona las herramientas poderosas necesarias para aplicaciones grandes y robustas. Una magnífica inversión del contenedor de control, el sistema de migración expresivo y el soporte de pruebas unitarias estrechamente integradas le brindan las herramientas que necesita para crear cualquier aplicación con la que se le asigne la tarea. (Otwell, 2019)

Debido a las propiedades que ofrece, posee una curva de aprendizaje media-alta, por lo que es imprescindible tener conocimientos previos de programación. Sin embargo, los beneficios e integración con múltiples lenguajes como PHP son impresionantes.

Además, toda su estructura se basa en dependencias, lo cual permite integrar una o varias librerías, uno o varios Frameworks y unos o varios motores de bases de datos. Esto le proporciona una alta gama de posibilidades en la construcción de interfaces y manejo de procesos con audio, video, texto, imágenes, etc. El gestor de dependencias de Laravel por defecto, se llama Composer.

Por otra parte, Laravel incluye una interfaz de comandos llamada Artisan que interpreta fácilmente las instrucciones dadas por un usuario a través de comandos. Para listar de manera rápida los comandos más básicos se usa el siguiente comando en el terminal: php artisan list. Así, se obtiene la lista de los comandos con una breve descripción de su uso; cómo se puede observar a continuación:



Figura 6. Lista de Comandos Artisan

Como se mencionó anteriormente, Laravel utiliza una estructura organizada por carpetas y ficheros para administrar las dependencias (librerías y/o Frameworks), las rutas (que permiten comunicar los componentes del proyecto entre la capa back-end con la capa front-end), los HTML (llamadas vistas), los controladores (que son equivalentes a la capa lógica del negocio del modelo vista controlador), los modelos de conexión con la base de datos, plantillas de carga de datos (llamadas semillas), plantillas de creación de tablas (llamadas migraciones) y algunos archivos de configuración de Laravel, como por ejemplo el archivo .env que almacena los datos de conexión con el Motor de Base de Datos. Pero por el momento se van a explicar las carpetas principales con su funcionamiento general:

1) app: Esta carpeta contiene el código principal de la aplicación.

2) config: Esta carpeta contiene todos los archivos de configuración de la aplicación como bases de datos, cache, correos, sesiones, servicios, etc.
3) database: Esta carpeta contiene la definición de las bases de datos y dentro de ella existen 3 subcarpetas importantes: factories, migrations y seeds (que como se mencionó anteriormente contienen plantillas configurables con carga de tablas, plantillas configurables para la simulación de usuarios con roles y permisos, y plantillas configurables con carga de datos de prueba).
4) public: Esta carpeta contiene las peticiones y solicitudes que pasan por nuestro servidor WEB. Además, aquí se encuentra el archivo index.php que inicia el proceso de ejecución del framework y el directorio en el que se alojan los archivos CSS, JavaScript, imágenes, videos, etc.
5) resources: Esta carpeta contiene tres subcarpetas llamadas Asset, Views y Lang. La carpeta Asset tiene las fuentes del proyecto, la carpeta Lang tiene los archivos PHP que guardan los estilos de las páginas y la carpeta Views contiene los archivos HTML que permiten visualizar las páginas WEB.
6) bootstrap: Esta carpeta contiene el código de carga inicial para incorporar el layout de la librería Bootstrap por defecto en nuestro proyecto.
7) storage: Esta carpeta contiene los archivos de sesión de la página WEB, el cache, las cookies, logs del sistema, etc.
8) vendor: Esta carpeta contiene todas las librerías y dependencias que posee Laravel.

No obstante, y con mayor profundidad existen además dos ficheros esenciales para el funcionamiento de Laravel :

1) .env: Este archivo con extensión .env posee la configuración de producción del servidor del sistema y el acceso a los motores de base de datos configurados por la aplicación.
2) composer.json: Este archivo con extensión .json permite realizar la instalación de Laravel utilizando el manejador de dependencias Composer y posteriormente de otros paquetes y/o dependencias que incluyan funcionalidades en Laravel.

Finalmente, y si se quiere relacionar lo previamente visto con una estructura lógica de funcionamiento se puede utilizar el siguiente esquema:

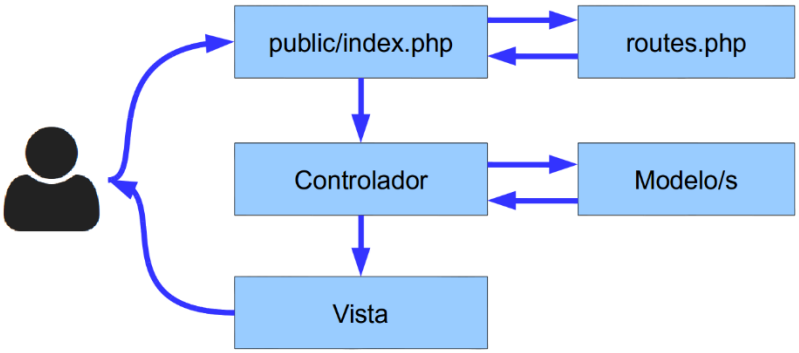


Figura 7. Estructura de Funcionamiento Laravel

Fuente: (Gallego, 2019)

Librerías.

La librería que se integrará al sistema es jQuery, ya que permite enriquecer páginas WEB creando interfaces innovadoras mediante la utilización de animaciones, efectos e interacciones por medio de AJAX (el cual responde a Asynchronous JavaScript And XML y sirve para dar mayor velocidad a la página WEB). La librería jQuery permite manipular los elementos del DOM.

> El Modelo de Objetos de Documento, en inglés, Document Object Model, (DOM), es una forma de representar los elementos de un documento estructurado (tal como una página WEB HTML o un documento XML) como objetos que tienen sus propios métodos y propiedades. El responsable del DOM es el World Wide WEB Consortium (W3C). (Hernández & Greguas, 2010)

jQuery es una librería que permite escribir con mayor facilidad código JavaScript. Por lo que para poder manejar esta librería se requiere un alto conocimiento en lo que se refiere a JavaScript, HTML y CSS.

JavaScript.

Es un lenguaje de programación que se utiliza para la creación de páginas WEB dinámicas, es decir, dentro de la página incorpora efectos especiales como animaciones, interacción con el usuario, manejo de botones, ventanas y mensajes.

Técnicamente, es un lenguaje interpretado por lo cual no es necesario compilar los programas para ejecutarlos, el navegador automáticamente revisa la sintaxis JavaScript para su ejecución o no, sin necesidad de procesos intermedios.

La sintaxis es similar a lenguajes como Java y C. Algunos aspectos básicos de JavaScript son los siguientes.

1. Se distinguen las mayúsculas de las minúsculas
2. No es necesario definir variables
3. Al crear una variable no se requiere definir el tipo de dato que almacenará

No es necesario terminar cada instrucción con punto y coma, sin embargo, puede ser conveniente colocarlo.

HTML.

Es un lenguaje de etiquetas el cual permite indicar el orden y cómo va a ser presentado el Contenido de una página WEB. Las etiquetas se encuentran rodeados de los símbolos mayor que y menor que. Por ejemplo, <body>.

Ahora se utiliza lo que se conoce como HTML5, el cual es una tecnología nueva que permite la construcción de aplicaciones y sitios WEB que combinan el uso de dispositivos móviles, computación en la nube y ambientes de red.

CSS.

En español se la conoce como una hoja de estilos. Se llama de esta manera debido a que permite definir el aspecto de los componentes que se encuentran dentro de un archivo HTML.

Normalmente, se solía manejar el diseño de las páginas WEB embebido en el archivo .html, pero a partir de la creación del archivo .css todo se encuentra ordenado de mejor manera por lo que facilita el entendimiento del código en todos los archivos.

Motor de Base de Datos.

Se utilizará PostgreSQL, debido a que el enfoque del Sistema necesita trabajar con datos que sean de tipo objeto - relacional. Por otra parte, posee una gran variedad de características que brindan a los desarrolladores una mayor facilidad para manejar los datos sin importar cuán grande estos conjuntos de datos sean. (PostgreSQL Global Development Group, 2019)

Además y a diferencia de otros motores de datos conocidos como Oracle y DB2, es gratuito y de código abierto. Además, permite la creación de tipos de datos propios y maneja de forma simple sentencias DDL, DML Y DCL.

DDL es la abreviación de Data Definition Language, traducido del inglés significa Lenguaje de Definición de Datos. Estas sentencias permiten al usuario crear bases de datos, tablas, vistas a partir del comando CREATE. Permiten alterar una tabla agregando una nueva columna o eliminando otra mediante la sentencia ALTER. Mediante DROP es posible eliminar una tabla o inclusive la base de datos por completo. (Segovia, 2017)

DML es la abreviación de Data Manipulation Language, traducido del inglés significa Lenguaje para la Manipulación de Datos. Permite la realización de operaciones CRUD (Crear, Leer, Actualizar y Borrar), mediante los comandos INSERT, SELECT, UPDATE y DELETE. (Segovia, 2017)

DCL es la abreviación de Data Control Language, traducido del inglés significa Lenguaje para el Control de Datos. Como su nombre lo indica, permite otorgar permisos utilizando el comando GRANT, así como a su vez se los puede retirar mediante el comando REVOKE. (Segovia, 2017)

Modelo de Cascada Modificado (Sinergy)

El desarrollo del sistema se encontrará regido por un proceso de ingeniería definido por fases secuenciales, que determinarán el ciclo de vida del proyecto. Para esto se utilizará el Modelo de Cascada Modificado (Synergy) , el cuál es una variante del Modelo de Cascada Normal; teniendo como principal diferencia la definición de dos permutaciones comunicativas entre las fases de análisis y diseño, entre las fases de construcción (implementación), pruebas y documentación.

Este modelo permite iterar hasta que se evidencie el cumplimiento de los requerimientos obtenidos en la fase de análisis y que el software alcance un nivel de calidad óptimo (tiempo de respuesta adecuado, concurrencia de usuarios ideal, etc).

Las fases del modelo de cascada modificado son las siguientes:

Análisis

Propósito:

- Analizar las necesidades del usuario de forma general
- Evaluar alternativas de solución al problema de manera general

Resultados:

- Especificar los requerimientos funcionales y no funcionales del sistema por desarrollar por medio de Diagramas de Caso de Uso.
- Definir los procesos de desarrollo mediante la elaboración de Diagramas de Actividades.

Diagramas por Obtener:

a. Elaborar Diagramas de Caso de Uso
   i. Diagramas de Casos de Uso General
   ii. Diagramas de Casos de Uso Detallado
   iii. Diagramas de Casos de Uso Siguiente Nivel

b. Elaborar Diagramas de Actividades
   i. Diagrama de Actividades General
   ii. Diagramas de Actividades Detallados

Diseño

Propósito:

- Especificar cómo se va a solventar de forma computacional a nivel arquitectónico y detallado.
    - Arquitectónica: plantear de forma general el software, los módulos y subsistemas

- Detallada: describir lo que realizan los módulos

Resultados:

- Especificación del diseño del sistema a realizar, descomponiéndolo en subsistemas, a través de Diagramas de Clases, Diagramas de Entidad-Relación y Diagramas de Paquetes.
- Especificación de cada módulo a manera de detalle, indicando lo que debe realizar cada uno de los módulos y en base a su implementación, mediante los Diagramas de Secuencia.

Diagramas por Desarrollar:

a. Elaborar Diagramas de Clases
   i. Diagrama de Clases General
   ii. Diagrama de Clases Detallado

b. Elaborar Diagramas Entidad-Relación
   i. Modelo Entidad-Relación Conceptual
   ii. Modelo Entidad-Relación Lógico
   iii. Modelo Entidad-Relación Físico

c. Elaborar Diagramas de Paquetes
   i. Diagramas de Paquetes por Dependencia
   ii. Diagramas de Paquetes por Asociación

d. Elaborar Diagramas de Secuencia

Construcción (Implementación)

Propósito:

- Implementar los módulos definidos en la Fase de Análisis en base a los requerimientos del usuario y probar su funcionamiento

Resultados:

- Codificar módulos definidos en la Fase de Análisis, en base a estándares de codificación WEB en lenguaje PHP (encabezados por módulos, convenciones de nombres de variables y subprogramas, máximo de líneas por componente, revisión de migraciones

incompatibles, uso de versión estable, estructura de Contenido adecuada).

- Diseñar y construir interfaces gráficas utilizando un diseño WEB Responsivo.

Pruebas

Propósito:

- Principal actividad de control de calidad, funcionamiento y respuesta del sistema

Resultados:

- Elaborar pruebas que permitan comprobar la funcionalidad y respuesta del sistema.

Documentación

Propósito:

- Describir el funcionamiento del sistema que explique su utilidad y funcionalidades al usuario y/o técnico del sistema.

Productos:

- Realizar explicaciones detalladas del Sistema en forma de Manual del Usuario (incluido en el Trabajo de Titulación), que permita describir la interacción del usuario con el software.
- Realizar explicaciones detalladas del Sistema en forma de Manual Técnico (incluido en el Trabajo de Titulación), que permita comprender el comportamiento funcional del software.

Implantación

Propósito:

- Instalar el software realizado en una computadora personal y/o empresarial para su uso en condiciones reales.

Resultado:

- Instalar el prototipo funcional realizado en una computadora personal para su uso en condiciones reales.



Figura 8. Sinergy

Fuente: Ortiz, 2019.

Capítulo 2

Análisis y Diseño

Fase de Análisis

En esta fase se tomará en cuenta todas las necesidades del usuario (Profesor/Estudiante) para construir la futura aplicación y también se identificarán los problemas en el ámbito educativo actual, para buscarles una solución.

Posteriormente, se creará modelos conceptuales de casos de uso que permitirán comprender el comportamiento externo del sistema. Así, se delimitará el alcance para el sistema que facilite la adquisición de conocimiento (significativo) por parte del Estudiante.

Finalmente, las necesidades recabadas serán convertidas en lo que se conoce como requerimientos funcionales de un sistema. Estos requerimientos son prácticamente los módulos operativos que le dan vida al mismo y en base a los cuales se puede validar el avance del proyecto.

Diagrama Caso de Uso General

Para realizar el Diagrama de Caso de Uso General se tomará en cuenta dos factores principales: la identificación de módulos básicos (clases básicas) y la identificación de los módulos avanzados (procesos). Y para nombrar los Casos de Uso se utilizará la estructura básica de nombrado que según De la Cruz, (s.f.) y Rumbaugh et al. (2007): “Los nombres de casos de uso son frases con verbos activos pequeños, éstos nombran el comportamiento encontrado en el vocabulario del sistema que se está modelando”.

A continuación, se mencionarán las principales funciones externas que contendrá el sistema representadas por casos de uso, y se dividirá en 2 imágenes para poder visualizar de mejor manera en el documento:

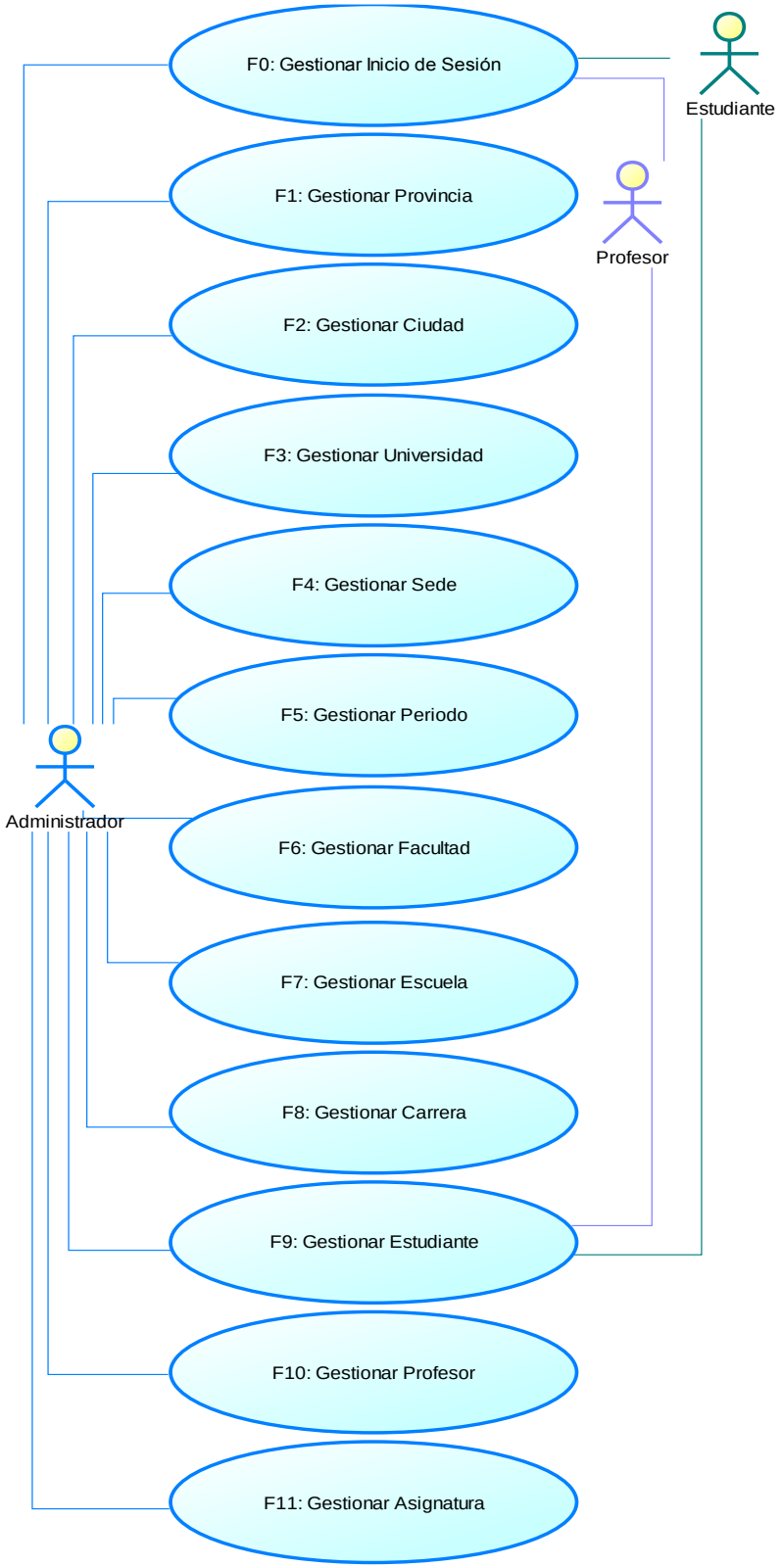


Figura 9. Diagrama de Caso de Uso General

El Diagrama presentado anteriormente contiene 1 proceso llamado Gestionar Inicio de Sesión y 10 clases básicas que permiten agregar funcionalidad a los procesos. Los 10 Casos de Uso básicos comparten una característica única que los distingue en el campo educativo y es que en base a estas se puede reconstruir el pénsum del docente con su respectivo Contenido dentro del sistema. Por otra parte, en base a dichos Contenidos el Estudiante puede tomar anotaciones en pantalla e interactuar con diversos componentes como el Glosario, Talleres, etc. Finalmente, se pueden observar las relaciones de asociación que posee cada actor (Administrador, Profesor y Estudiante) con respecto a las funcionalidades externas del sistema. Y posteriormente, se detallan con exactitud los permisos de acceso específicos que tienen cada actor con respecto a cada Caso de Uso. Sin embargo, cabe recalcar que en el caso del rol de Estudiante sus permisos de acceso a cada funcionalidad recaen principalmente en la visualización de Contenido y toma de Anotaciones.

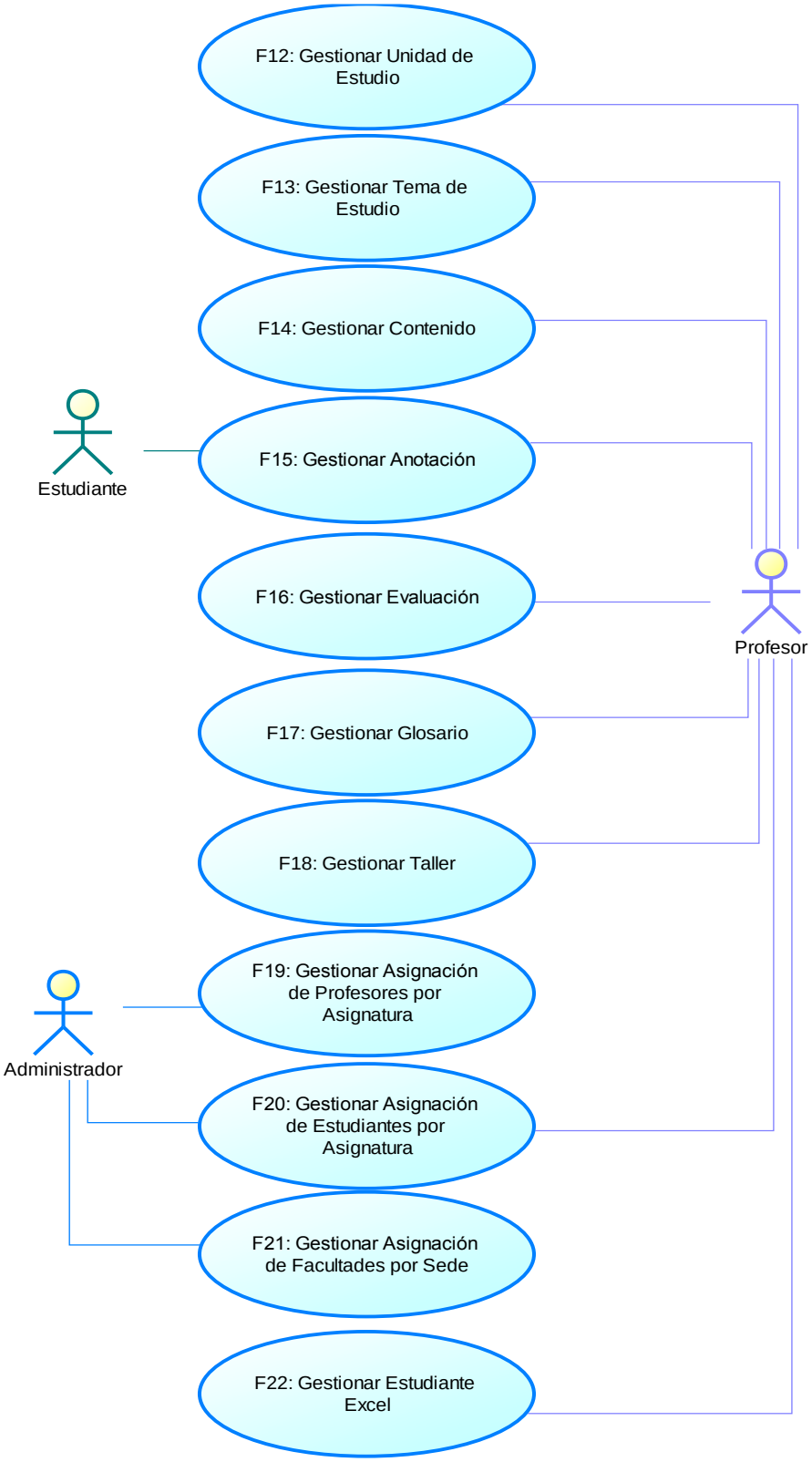


Figura 10. Continuación de Diagrama de Caso de Uso General

El Diagrama presentado anteriormente contiene 3 procesos llamados Gestionar Anotaciones, Gestionar Asignación de Estudiantes por Asignatura, Gestionar Asignación de Profesores por Asignatura y 8 clases básicas que permiten agregar funcionalidad a los procesos. Los 8 Casos de Uso básicos comparten una característica única que los distingue en el campo universitario ya que se puede reconstruir la estructura educativa, tanto como la Ciudad, Sede y Provincia en la que se encuentra la Universidad; así como las Carreras, Profesores y Estudiantes que pertenecen a la misma. Además, de que permite controlar el Periodo educativo vigente en el que se encuentra el ciclo académico. Finalmente, se pueden observar las relaciones de asociación que posee cada actor (Administrador y Profesor) con respecto a las funcionalidades externas del sistema. Y posteriormente, se detallan con exactitud los permisos de acceso específicos que tienen cada actor con respecto a cada Caso de Uso. Sin embargo, cabe recalcar que en este segmento el encargado de realizar el manejo de los datos en su mayoría corresponde al Administrador y en algunos casos al Profesor con sus propios Estudiantes.

Diagrama de Actividades

Los Diagramas de Actividades se elaboran solamente para las funcionalidades que sean procesos a los que se quiere automatizar. En este caso los procesos son las funcionalidades:

- F0: Gestionar Inicio de Sesión
- F15: Gestionar Anotación
- F19: Gestionar Asignación de Asignaturas por Profesor
- F20: Gestionar Asignación de Asignaturas por Estudiante

F0: Gestionar Inicio de Sesión

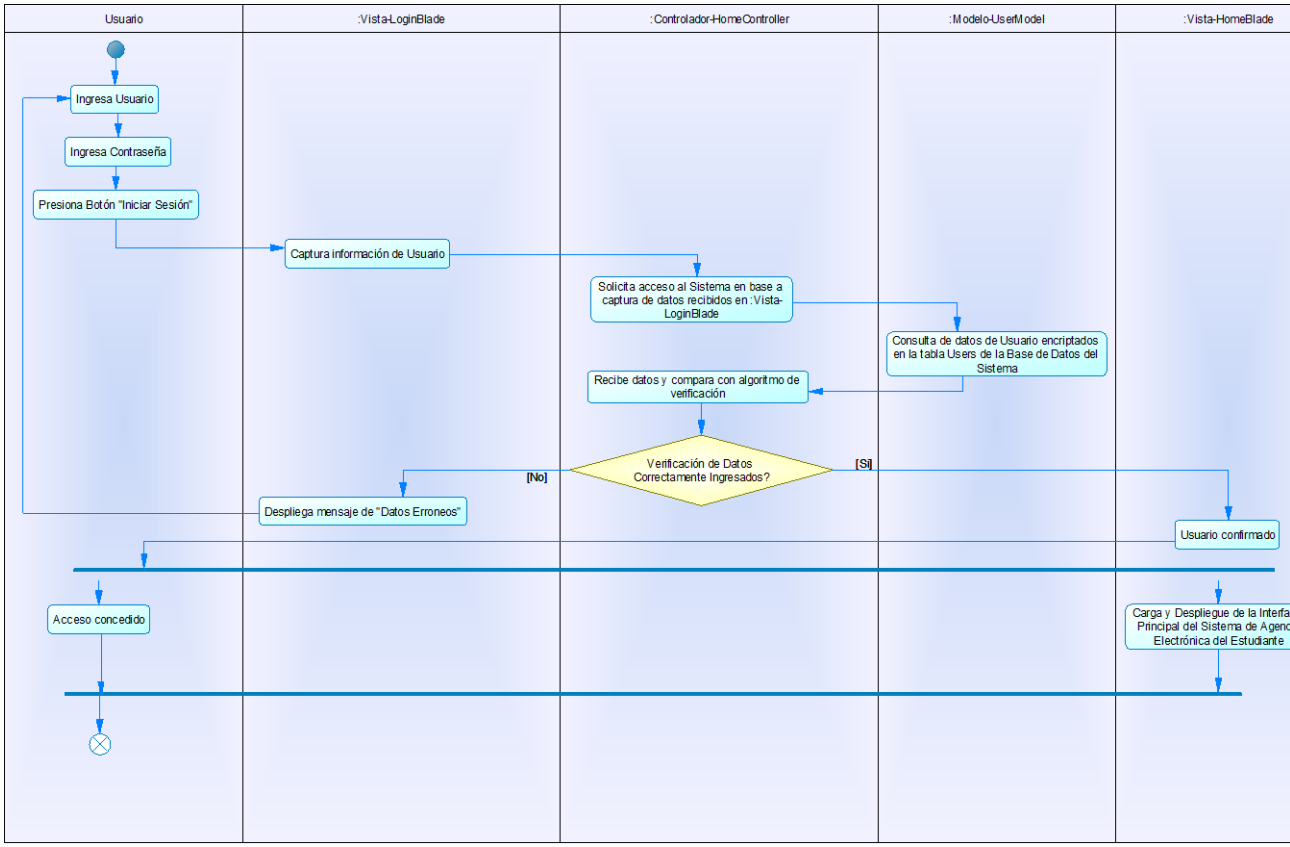


Figura 11. Diagrama de Actividades F0: Gestionar Inicio de Sesión

Pasos

1. El Usuario intenta ingresar al Sistema de Agenda Electrónica del Estudiante, por lo que ingresa Nombre de Usuario, Contraseña y presiona el botón “Iniciar Sesión” en :Vista-LoginBlade.
2. En :Vista-LoginBlade se captura la información del Usuario y se solicita en base a su :Controlador-HomeController que se verifiquen (mediante un algoritmo) que los datos ingresados sean iguales a los datos encriptados que se encuentran en la tabla de la base de datos Users; para esto se usa una llamada SQL con la ayuda del :Modelo-UserModel.
3. Si no coinciden los datos ingresados por el Usuario con los de la consulta SQL de la tabla Users, se despliega un mensaje de error que dice “Datos Erróneos” y el usuario tiene que nuevamente volver a empezar desde el Paso 1. Caso

contrario, se recibe una validación equivalente a Usuario confirmado dentro de la :Vista-HomeBlade.

4. En paralelo se cargan y despliegan los datos de la :Vista-HomeBlade al Usuario y se le concede el acceso mediante una redirección de Vista; por último y así termina el proceso.

F15: Gestionar Anotación

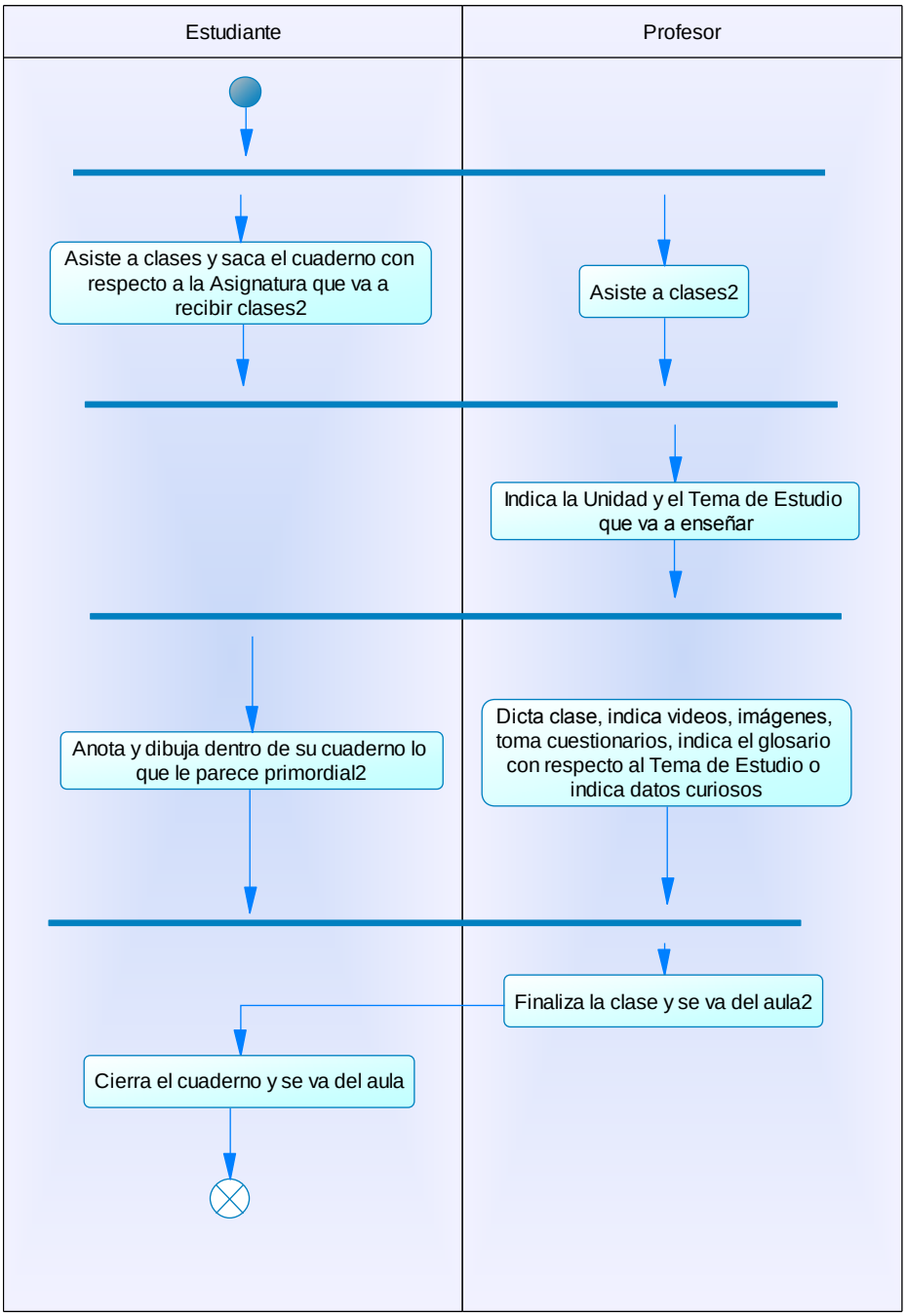


Figura 12. Diagrama de Actividades F15: Gestionar Anotación

Pasos

1. El Estudiante y el Profesor asisten a clases. El Estudiante saca su cuaderno con respecto a la Asignatura que va a recibir clases
2. El Profesor indica la Unidad y el Tema de Estudio que va a dictar en la clase
3. El Profesor dicta la clase e indica videos, imágenes, toma cuestionarios, indica el glosario con respecto al Tema de Estudio y también indica otros datos curiosos o información adicional, al mismo tiempo el Estudiante anota y dibuja dentro de su cuaderno lo que le parece primordial con respecto a lo que el Profesor dicta en clase
4. El Profesor finaliza la clase y se va del aula
5. El Estudiante cierra su cuaderno y se va del aula

F19: Gestionar Asignación de Profesores por Asignatura

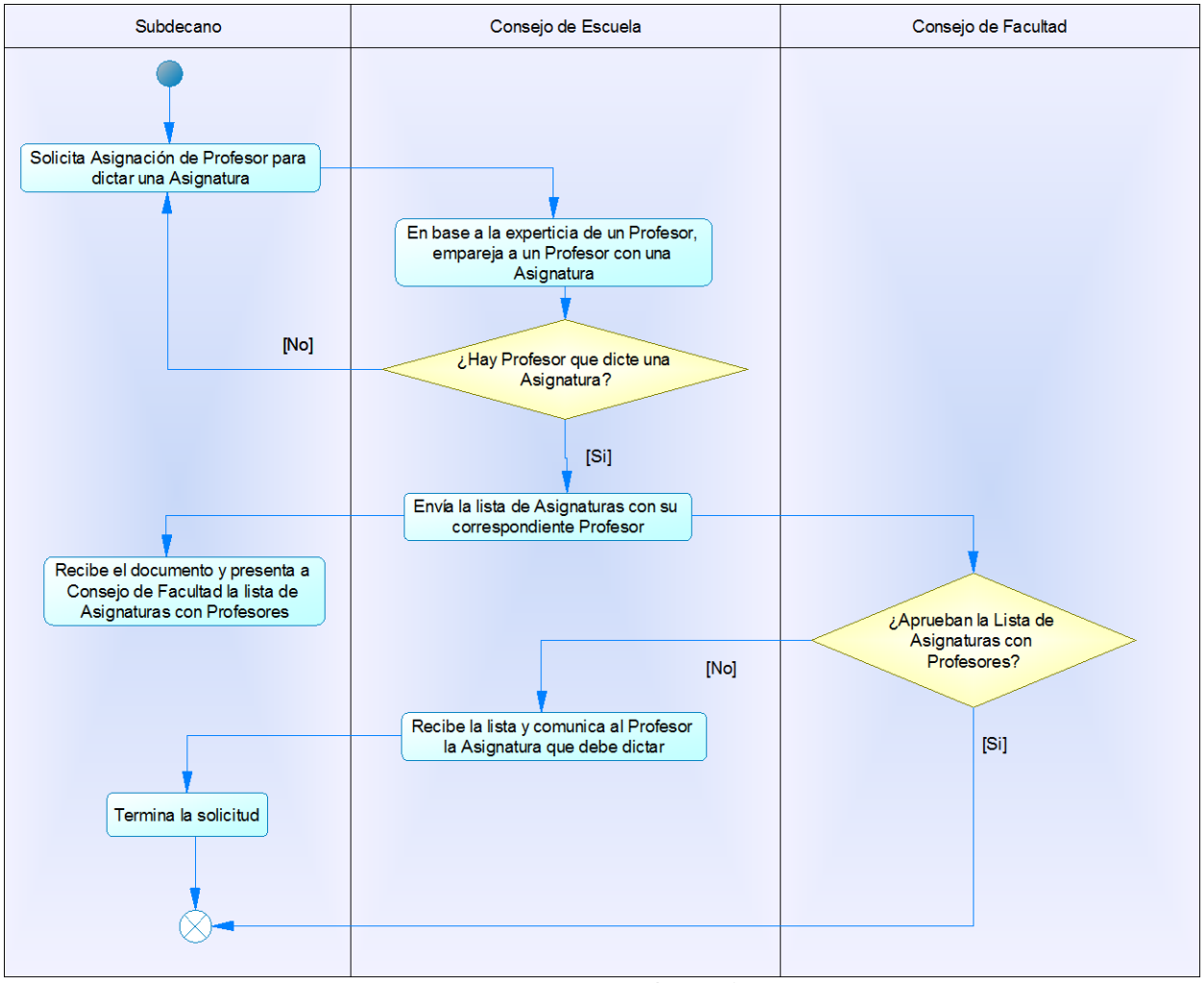


Figura 13. Diagrama de Actividades F19: Gestionar Asignación de Profesores por Asignatura

Pasos

5. El Subdecano solicita Asignación de Profesor a el Director de Escuela para que dicte una Asignatura
6. El Consejo de Escuela, empareja a un Profesor con una Asignatura en base a la experticia que el Profesor posea
7. El Consejo de Escuela verifica si hay un Profesor que pueda dictar la Asignatura, en caso de que no El Subdecano solicita Asignación de Profesor para dictar una Asignatura a otra Facultad, caso contrario, el Consejo de Escuela envía la lista de Asignaturas con su respectivo Profesor al Subdecano

8. El Subdecano recibe el documento y presenta la lista de Asignaturas con Profesores al Consejo de Facultad
9. Si el Consejo de Facultad aprueba la lista entonces el consejo de Escuela recibe la lista y comunica al Profesor la Asignatura que debe dictar y se termina la solicitud. En caso contrario, si el Consejo de Facultad no aprueba la lista, se termina el proceso

F20: Gestionar Asignación de Estudiantes por Asignatura

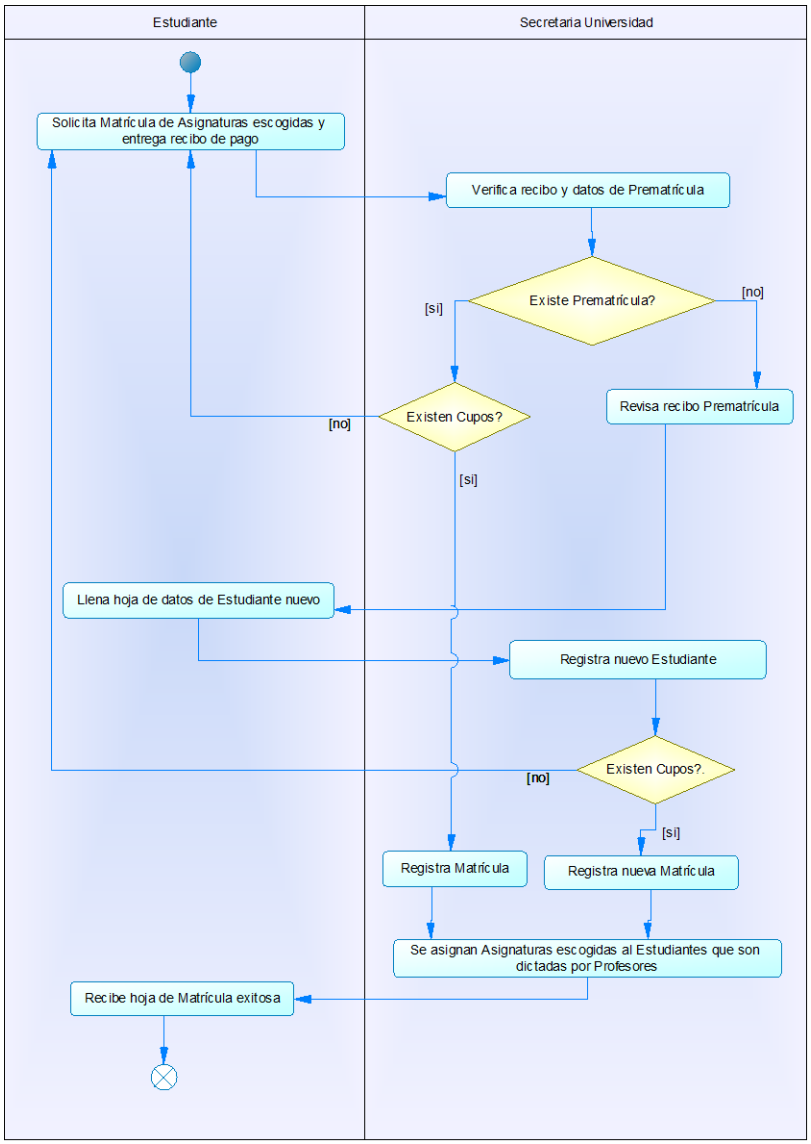


Figura 14. Diagrama de Actividades F20: Gestionar Asignación de Estudiantes por Asignatura

Pasos

1. El Estudiante solicita Matrícula de Asignaturas Escogidas y entrega recibo de pago.
2. La Secretaria Universitaria verifica y recibe datos de Prematricula y de existir, verifica si hay cupos; de no haber cupos, el Estudiante realiza nuevamente todo el proceso desde el Paso 1. Caso contrario, se registra la Matrícula en el Sistema Universitario y se asignan Asignaturas escogidas al Estudiante que son dictadas por Profesores. Posteriormente, el Estudiante recibe la hoja de Matrícula exitosa y se acaba el proceso.
3. Ahora bien, si no existe Prematricula la Secretaria recibe también el pago previo de la Prematricula y solicita los datos del Estudiante para poder registrarlo en el Sistema Universitario.
4. La Secretaria verifica si hay cupos disponibles para la Prematricula y de no existir cupos, el Estudiante tiene que realizar nuevamente todo desde el Paso 1. Caso contrario se registra una nueva Matrícula en el Sistema Universitario y se asignan Asignaturas escogidas al Estudiante que son dictadas por Profesores.
5. Finalmente, el Estudiante recibe la hoja de Matrícula exitosa y se acaba el proceso.

Diagrama Caso de Uso Siguiente Nivel

F0: Gestionar Inicio de Sesión

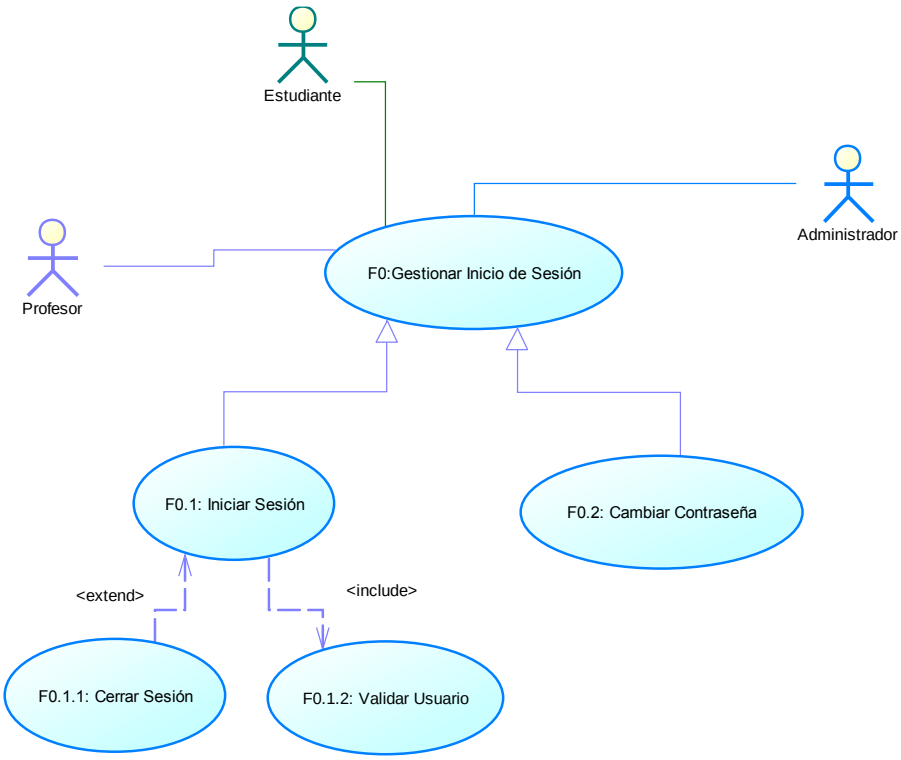


Figura 15. Diagrama C.U. Funcionalidad 0: Gestionar Inicio de Sesión

El Diagrama mostrado anteriormente permite visualizar el funcionamiento externo del proceso de inicio de sesión en el momento en el que un actor (Administrador, Profesor y Estudiante) accede al sistema. La idea principal es que el actor pueda acceder con una cuenta de correo asignada por la Universidad y con una respectiva contraseña, la cuál de preferencia es la misma que se coloca en la cuenta de Estudiante Intranet de la cuenta Universitaria. Posteriormente, los datos ingresados por el actor validados en base a la información registrada en la base de datos de este sistema (utilizando una interface de Login). Cabe recalcar, que cada uno de los Usuarios es capaz de cambiar su contraseña al Iniciar Sesión.

F1: Gestionar Provincia

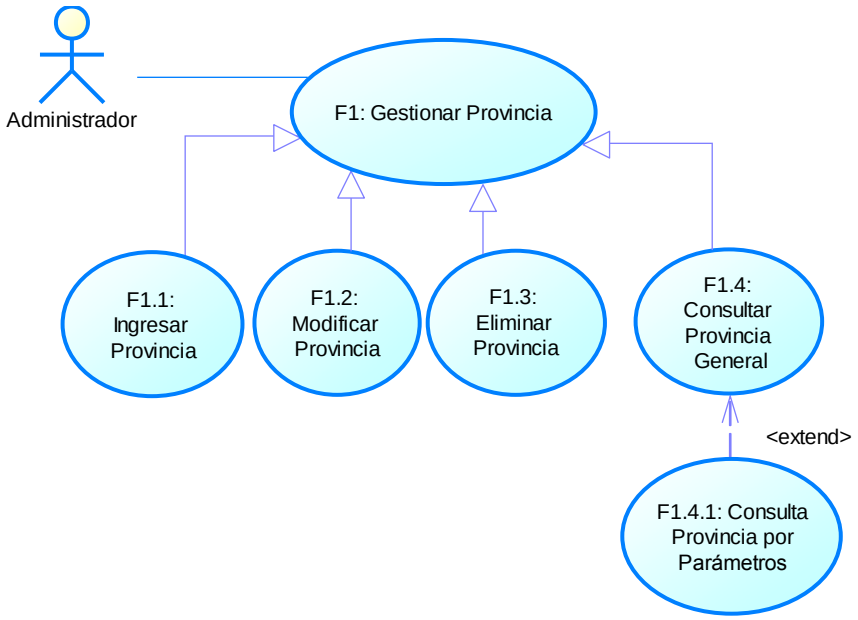


Figura 16. Diagrama C.U. Funcionalidad 1: Gestionar Provincia

El Diagrama anterior permite visualizar la estructura externa que posee la clase básica Provincia. Según lo observado, se pretende realizar una interface que maneje las 4 operaciones básicas CRUD (Create, Read, Update, Delete) con respecto a la base de datos, y que permita manejar y consultar la información de Provincias disponibles por parte de todos los actores que tengan acceso al sistema. Sin embargo, el único actor que tiene permiso para realizar la funcionalidad de Provincias, en este caso es el Administrador.

F2: Gestionar Ciudad

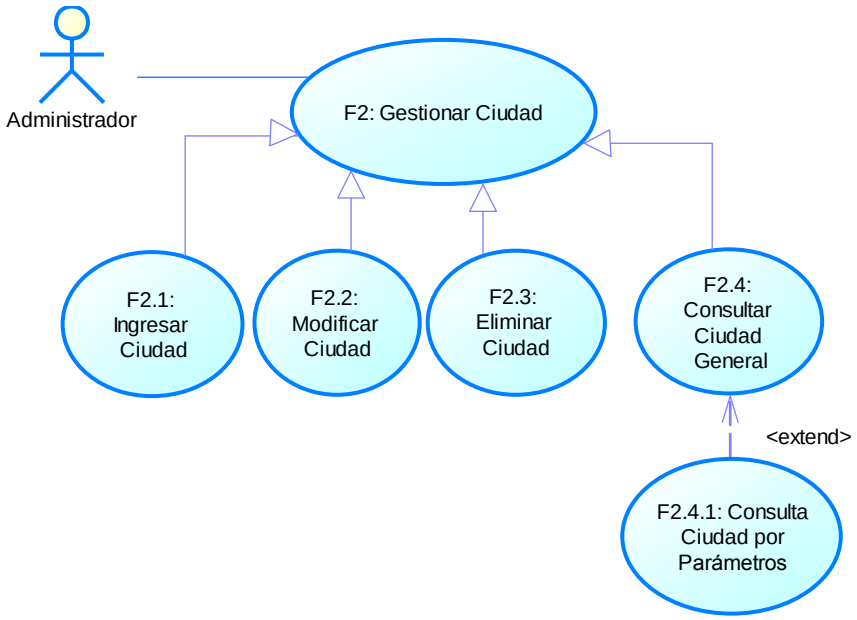


Figura 17. Diagrama C.U. Funcionalidad 2: Gestionar Ciudad

El Diagrama anterior permite visualizar la estructura externa que posee la clase básica Ciudad. Según lo observado, se pretende realizar una interface que maneje las 4 operaciones básicas CRUD (Create, Read, Update, Delete) con respecto a la base de datos, y que permita manejar y consultar la información de Ciudades disponibles por parte de todos los actores que tengan acceso al sistema. Sin embargo, el único actor que tiene permiso para realizar la funcionalidad de Ciudades, en este caso es el Administrador.

F3: Gestionar Universidad

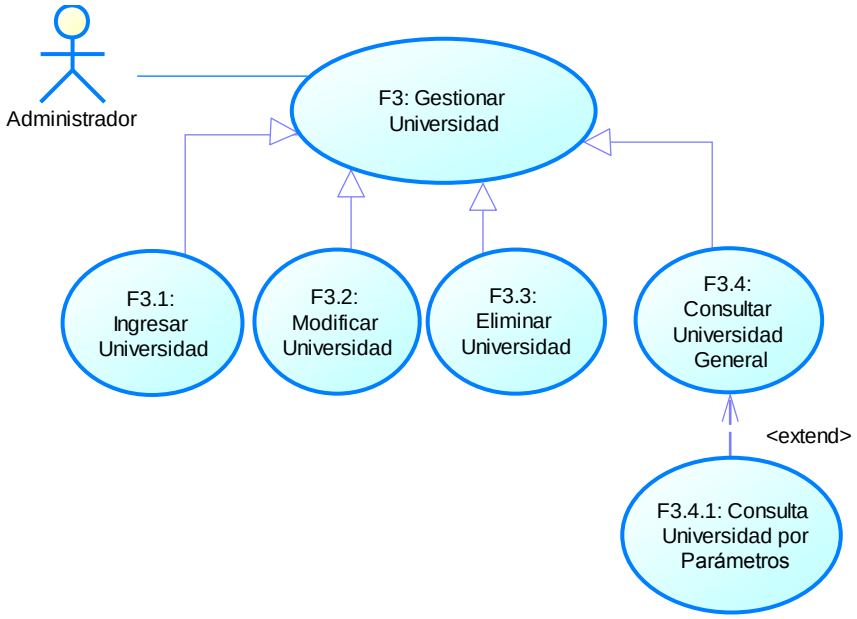


Figura 18. Diagrama C.U. Funcionalidad 3: Gestionar Universidad

El Diagrama anterior permite visualizar la estructura externa que posee la clase básica Universidad. Según lo observado, se pretende realizar una interface que maneje las 4 operaciones básicas CRUD (Create, Read, Update, Delete) con respecto a la base de datos, y que permita manejar y consultar la información de Universidades disponibles por parte de todos los actores que tengan acceso al sistema. Sin embargo, el único actor que tiene permiso para realizar la funcionalidad de Universidades, en este caso es el Administrador.

F4: Gestionar Sede

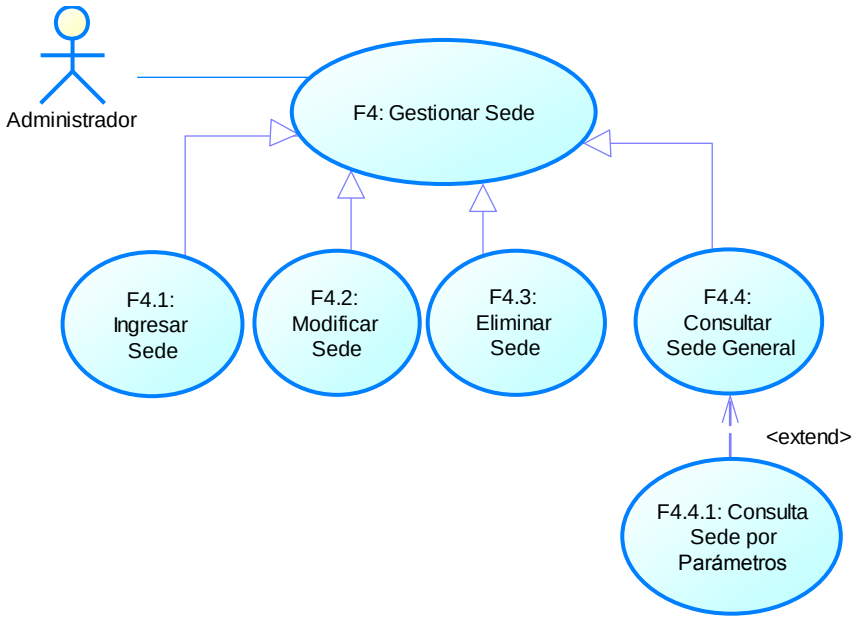


Figura 19. Diagrama C.U. Funcionalidad 4: Gestionar Sede

El Diagrama anterior permite visualizar la estructura externa que posee la clase básica Sede. Según lo observado, se pretende realizar una interface que maneje las 4 operaciones básicas CRUD (Create, Read, Update, Delete) con respecto a la base de datos, y que permita manejar y consultar la información de Sedes disponibles por parte de todos los actores que tengan acceso al sistema. Sin embargo, el único actor que tiene permiso para realizar la funcionalidad de Sedes, en este caso es el Administrador.

F5: Gestionar Periodo

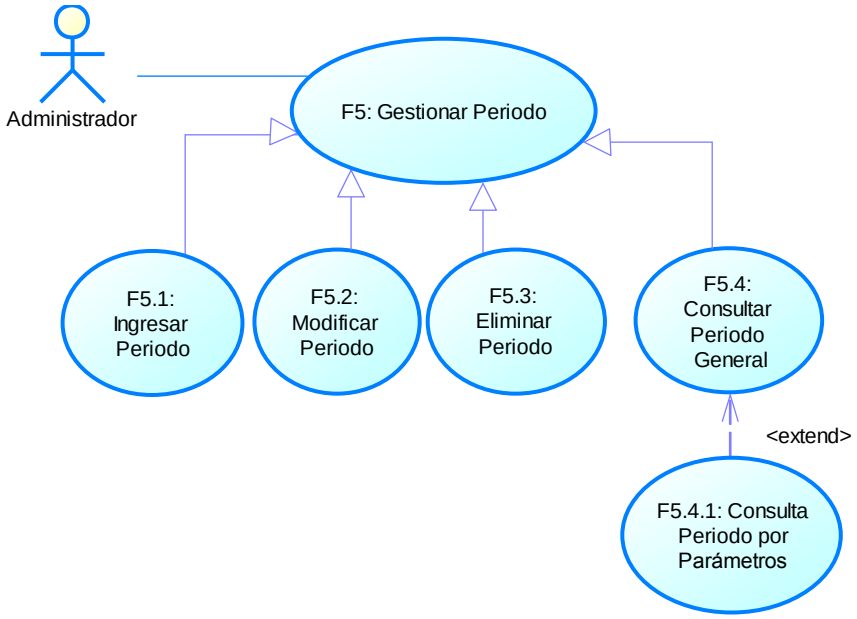


Figura 20. Diagrama C.U. Funcionalidad 5: Gestionar Periodo

El Diagrama anterior permite visualizar la estructura externa que posee la clase básica Período. Según lo observado, se pretende realizar una interface que maneje las 4 operaciones básicas CRUD (Create, Read, Update, Delete) con respecto a la base de datos, y que permita manejar y consultar la información de Períodos disponibles por parte de todos los actores que tengan acceso al sistema. Sin embargo, el único actor que tiene permiso para realizar la funcionalidad de Períodos, en este caso es el Administrador.

F6: Gestionar Facultad

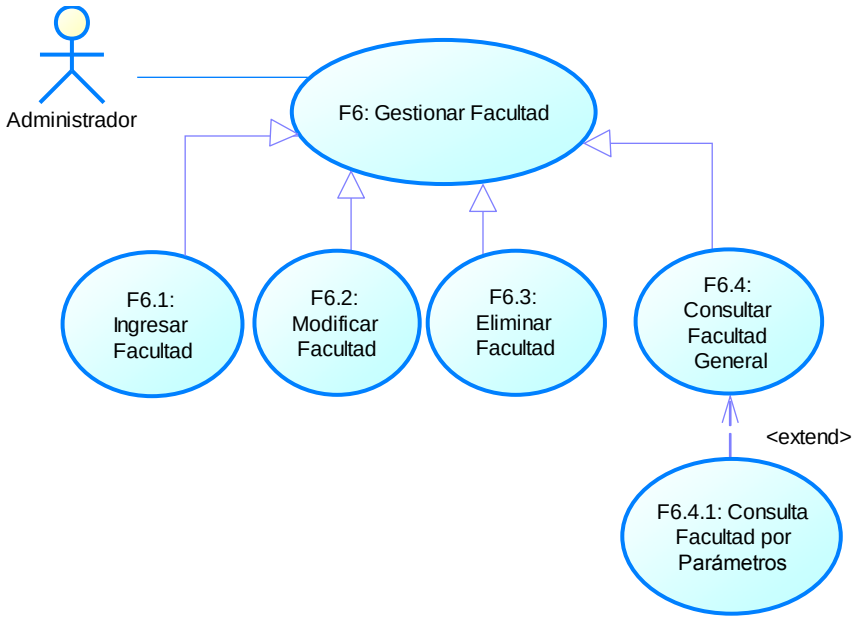


Figura 21. Diagrama C.U. Funcionalidad 6: Gestionar Facultad

El Diagrama anterior permite visualizar la estructura externa que posee la clase básica Facultad. Según lo observado, se pretende realizar una interface que maneje las 4 operaciones básicas CRUD (Create, Read, Update, Delete) con respecto a la base de datos, y que permita manejar y consultar la información de Facultades disponibles por parte de todos los actores que tengan acceso al sistema. Sin embargo, el único actor que tiene permiso para realizar la funcionalidad de Facultades, en este caso es el Administrador.

F7: Gestionar Escuela

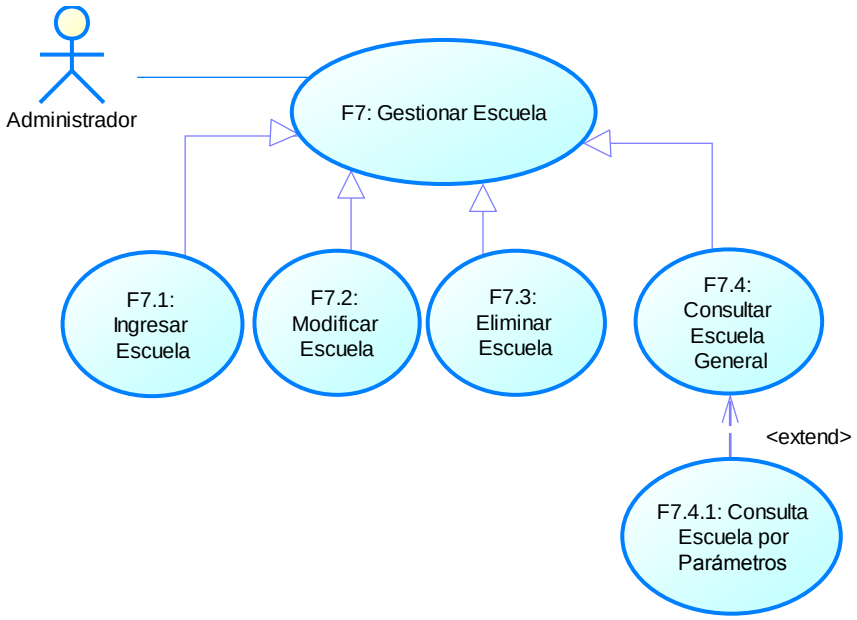


Figura 22. Diagrama C.U. Funcionalidad 7: Gestionar Escuela

El Diagrama anterior permite visualizar la estructura externa que posee la clase básica Escuela. Según lo observado, se pretende realizar una interface que maneje las 4 operaciones básicas CRUD (Create, Read, Update, Delete) con respecto a la base de datos, y que permita manejar y consultar la información de Escuelas disponibles por parte de todos los actores que tengan acceso al sistema. Sin embargo, el único actor que tiene permiso para realizar la funcionalidad de Escuelas, en este caso es el Administrador.

F8: Gestionar Carrera

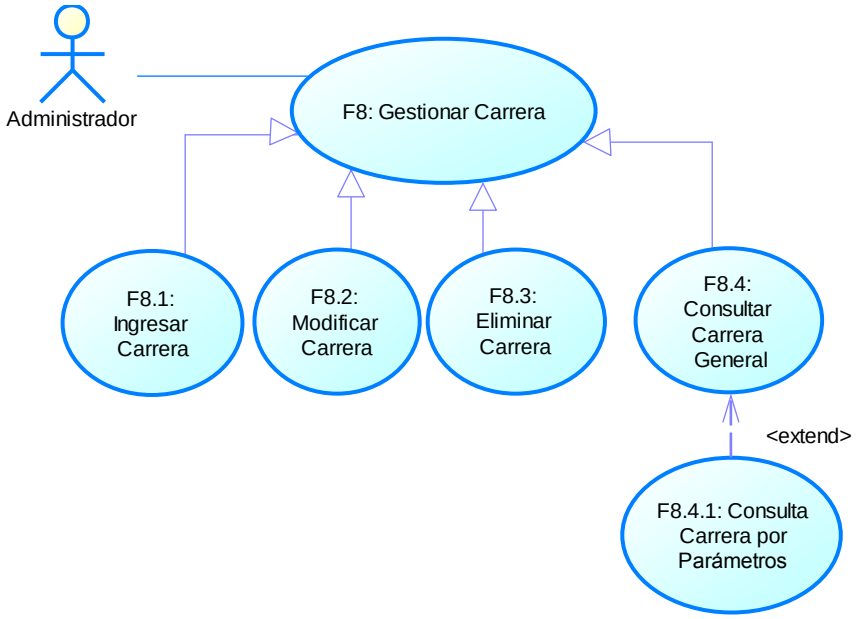


Figura 23. Diagrama C.U. Funcionalidad 8: Gestionar Carrera

El Diagrama anterior permite visualizar la estructura externa que posee la clase básica Carrera. Según lo observado, se pretende realizar una interface que maneje las 4 operaciones básicas CRUD (Create, Read, Update, Delete) con respecto a la base de datos, y que permita manejar y consultar la información de Carreras disponibles por parte de todos los actores que tengan acceso al sistema. Sin embargo, el único actor que tiene permiso para realizar la funcionalidad de Carreras, en este caso es el Administrador.

F9: Gestionar Estudiante

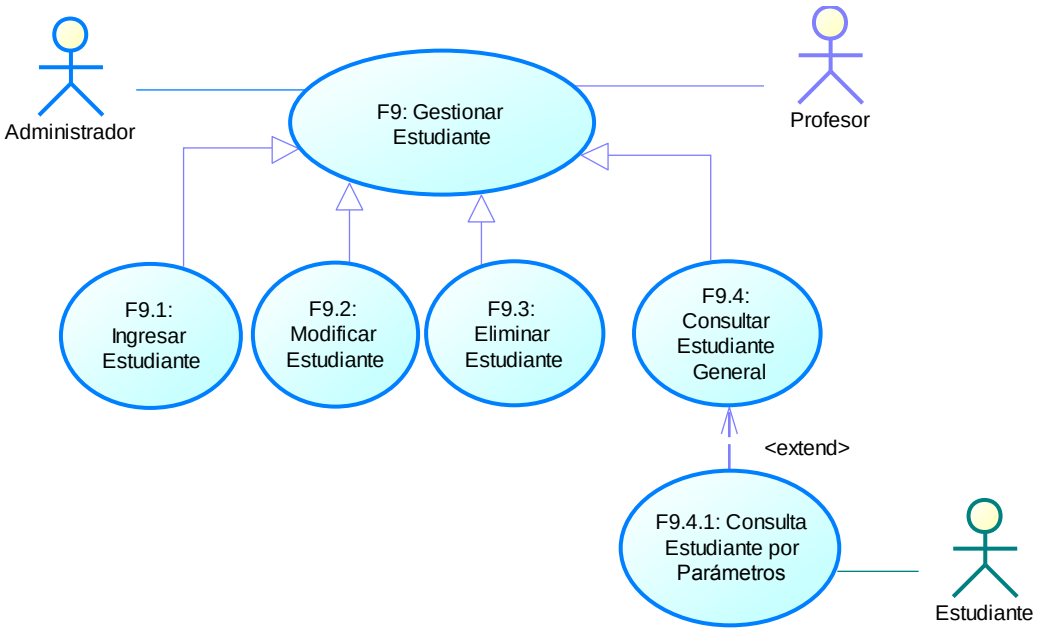


Figura 24. Diagrama C.U. Funcionalidad 9: Gestionar Estudiante

El Diagrama anterior permite visualizar la estructura externa que posee la clase básica Estudiante. Según lo observado, se pretende realizar una interface que maneje las 4 operaciones básicas CRUD (Create, Read, Update, Delete) con respecto a la base de datos. Aquí, el actor Administrador y Profesor tienen permiso total para manejar la información completa de Estudiantes (es decir todo el CRUD); mientras que el actor Estudiante simplemente puede Consultar por Parámetros sus datos con respecto a sí mismo para comprobar si se encuentran registrados correctamente.

F10: Gestionar Profesor

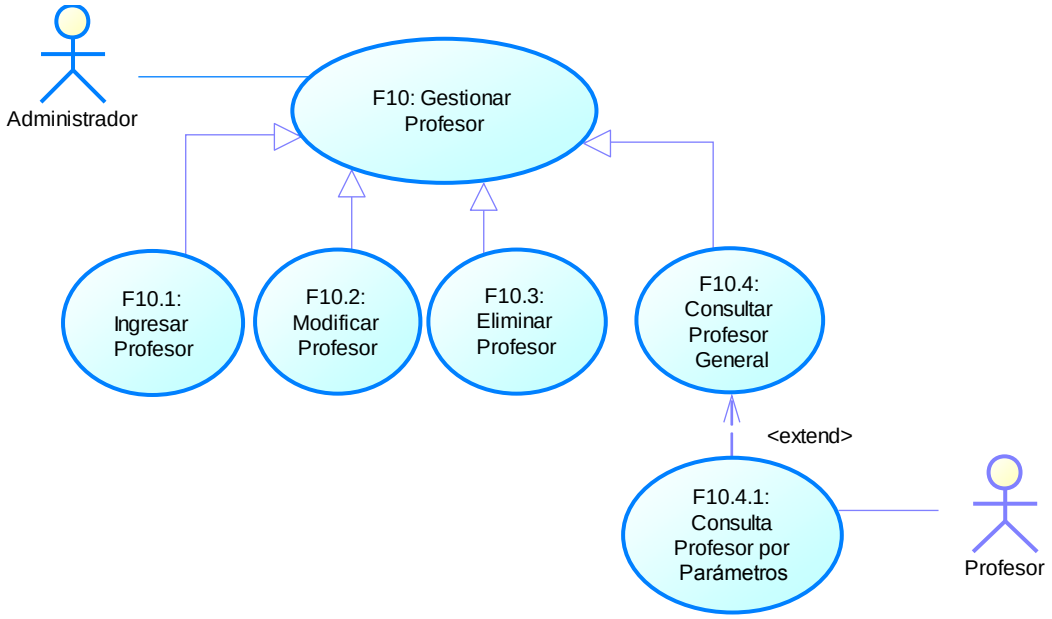


Figura 25. Diagrama C.U. Funcionalidad 10: Gestionar Profesor

El Diagrama anterior permite visualizar la estructura externa que posee la clase básica Profesor. Según lo observado, se pretende realizar una interface que maneje las 4 operaciones básicas CRUD (Create, Read, Update, Delete) con respecto a la base de datos. Aquí, el actor Administrador tiene permiso total para manejar la información completa de Profesores (es decir todo el CRUD); mientras que el actor Profesor simplemente puede Consultar por Parámetros sus datos con respecto a sí mismo para comprobar si se encuentran registrados correctamente.

F11: Gestionar Asignatura

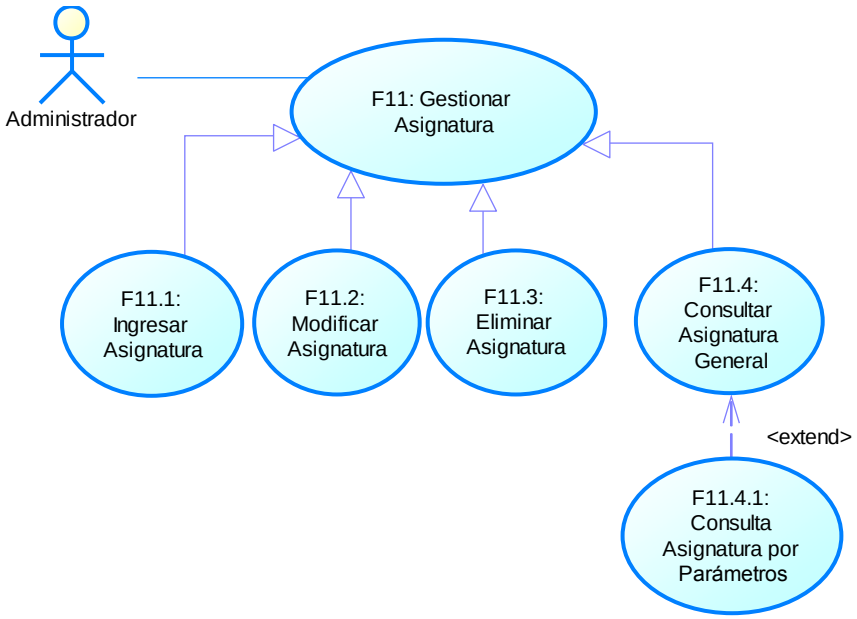


Figura 26. Diagrama C.U. Funcionalidad 11: Gestionar Asignatura

El Diagrama anterior permite visualizar la estructura externa que posee la clase básica Asignatura. Según lo observado, se pretende realizar una interface que maneje las 4 operaciones básicas CRUD (Create, Read, Update, Delete) con respecto a la base de datos. Aquí, el actor Administrador tiene permiso total para manejar la información completa de Asignaturas (es decir todo el CRUD).

F12: Gestionar Unidad de Estudio

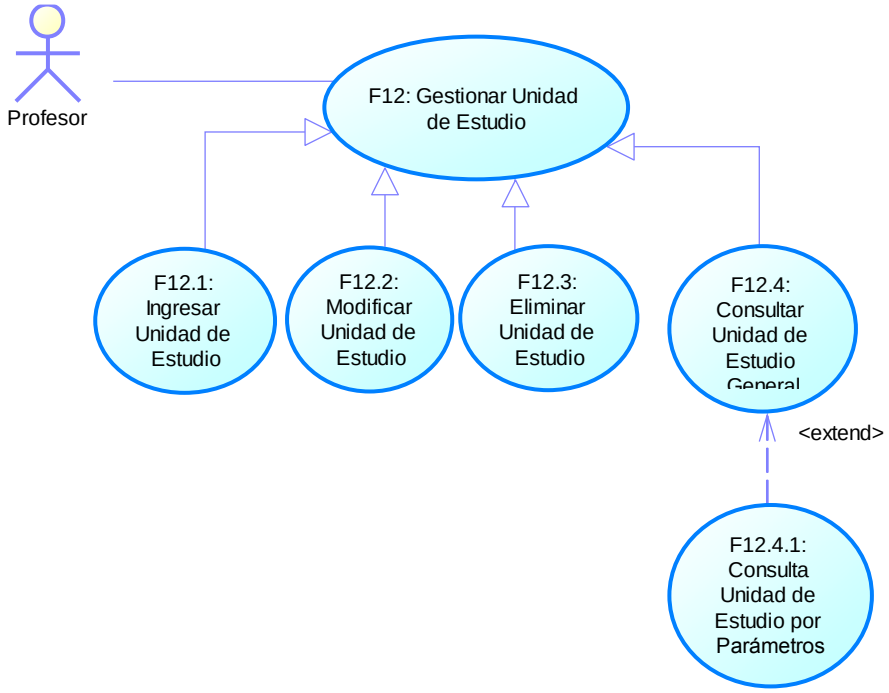


Figura 27. Diagrama C.U. Funcionalidad 12: Gestionar Unidad de Estudio

El Diagrama anterior permite visualizar la estructura externa que posee la clase básica Unidad de Estudio. Según lo observado, se pretende realizar una interface que maneje las 4 operaciones básicas CRUD (Create, Read, Update, Delete) con respecto a la base de datos. Aquí, el actor Profesor tienen permiso total para manejar la información completa de Unidades de Estudio (es decir todo el CRUD).

F13: Gestionar Tema de Estudio

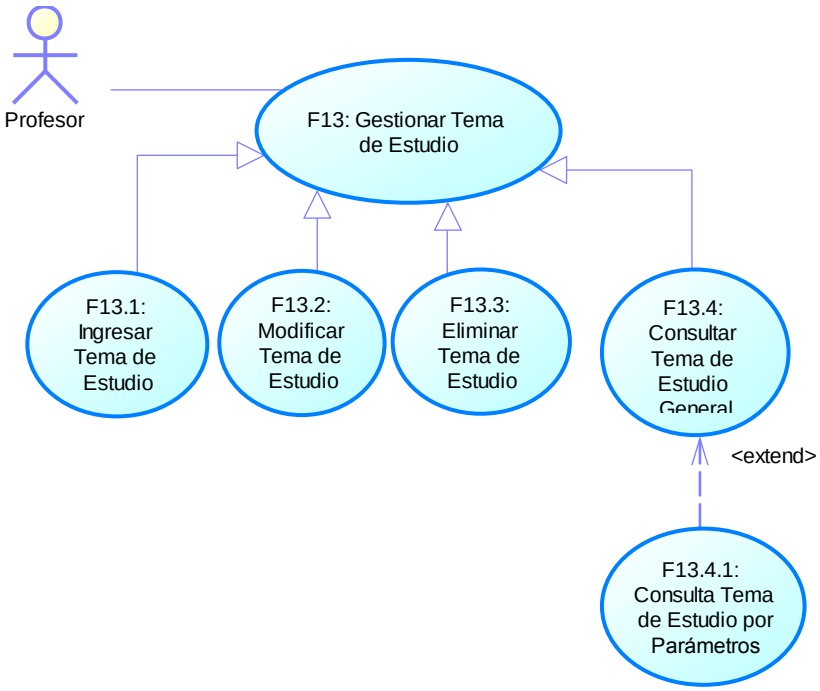


Figura 28. Diagrama C.U. Funcionalidad 13: Gestionar Tema de Estudio

El Diagrama anterior permite visualizar la estructura externa que posee la clase básica Tema de Estudio. Según lo observado, se pretende realizar una interface que maneje las 4 operaciones básicas CRUD (Create, Read, Update, Delete) con respecto a la base de datos. Aquí, el actor Profesor tienen permiso total para manejar la información completa de los Temas de Estudio (es decir todo el CRUD).

F14: Gestionar Contenido

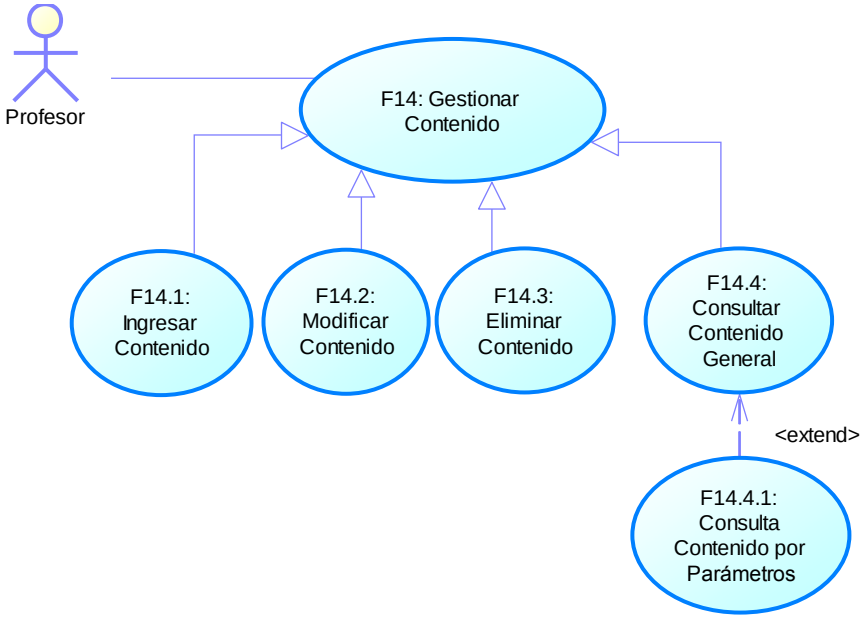


Figura 29. Diagrama C.U. Funcionalidad 14: Gestionar Contenido

El Diagrama anterior permite visualizar la estructura externa que posee la clase básica Contenido. Según lo observado, se pretende realizar una interface que maneje las 4 operaciones básicas CRUD (Create, Read, Update, Delete) con respecto a la base de datos. Aquí, el actor Profesor tienen permiso total para manejar la información completa de los Contenidos (es decir todo el CRUD).

F15: Gestionar Anotación

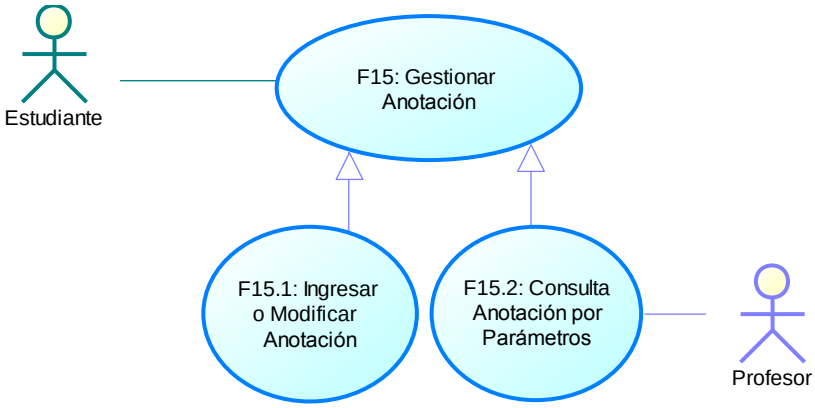


Figura 30. Diagrama C.U. Funcionalidad 15: Gestionar Anotación

El Diagrama anterior permite visualizar la estructura externa que posee la clase básica Anotación. Según lo observado, se pretende realizar una interface que maneje 3 operaciones básicas CRUD (Create, Read, Update) con respecto a la base de datos. Aquí, el actor Estudiante tiene permiso total para manejar la información completa de Anotaciones; mientras que, el actor Profesor solo puede visualizar la Consulta por parámetros de Anotaciones.

F16: Gestionar Evaluación

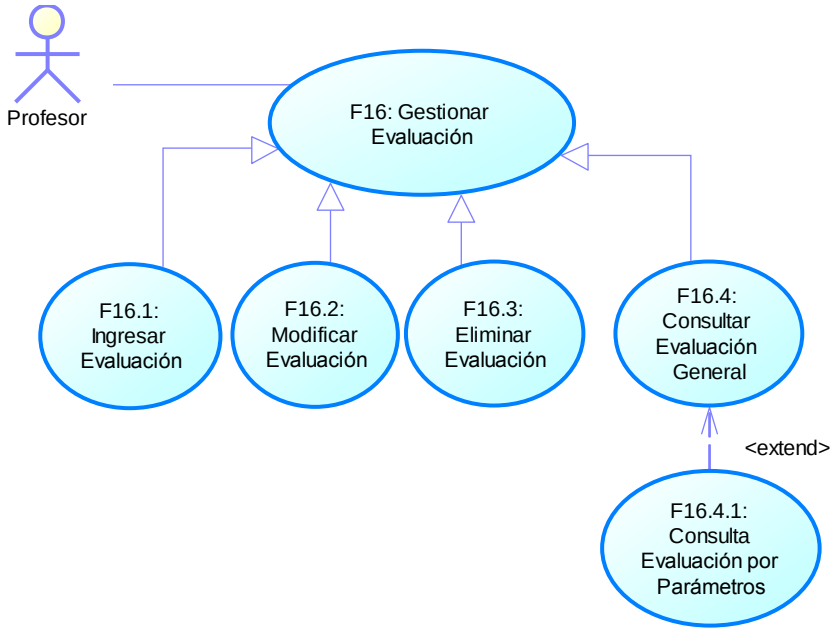


Figura 31. Diagrama C.U. Funcionalidad 16: Gestionar Evaluación

El Diagrama anterior permite visualizar la estructura externa que posee la clase básica Evaluación. Según lo observado, se pretende realizar una interface que maneje las 4 operaciones básicas CRUD (Create, Read, Update, Delete) con respecto a la base de datos. Aquí, el actor Profesor tiene permiso total para manejar la información completa de Evaluaciones (es decir todo el CRUD).

F17: Gestionar Glosario

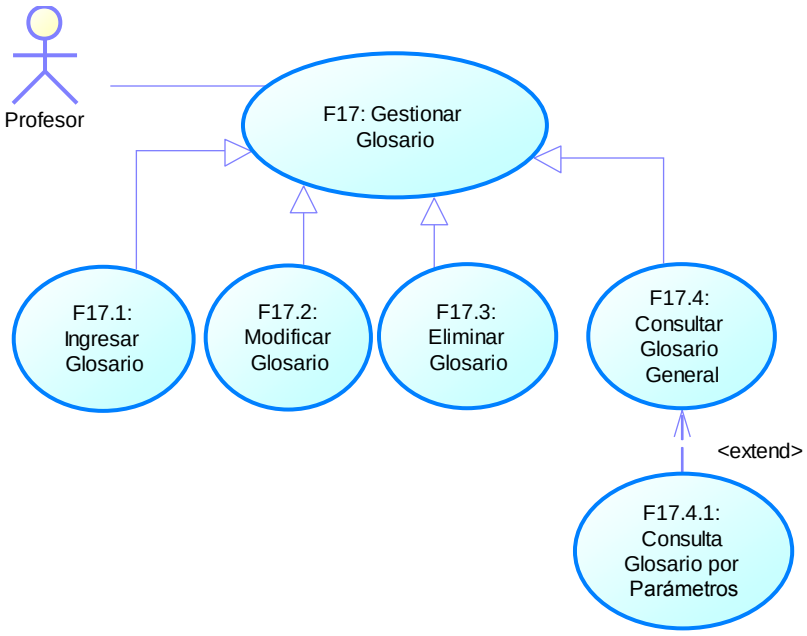


Figura 32. Diagrama C.U. Funcionalidad 17: Gestionar Glosario

El Diagrama anterior permite visualizar la estructura externa que posee la clase básica Glosario. Según lo observado, se pretende realizar una interface que maneje las 4 operaciones básicas CRUD (Create, Read, Update, Delete) con respecto a la base de datos. Aquí, el actor Profesor tiene permiso total para manejar la información completa de Glosarios (es decir todo el CRUD).

F18: Gestionar Taller

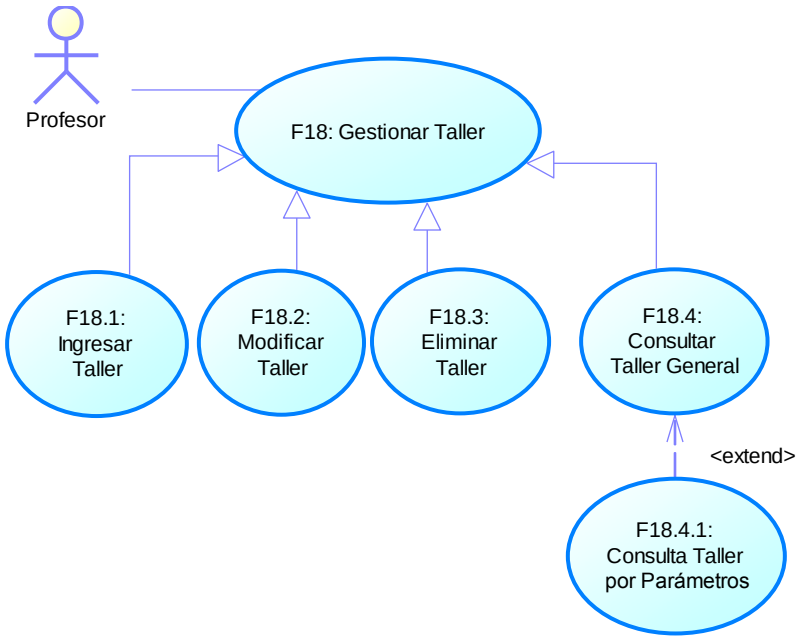


Figura 33. Diagrama C.U. Funcionalidad 18: Gestionar Taller

El Diagrama anterior permite visualizar la estructura externa que posee la clase básica Taller. Según lo observado, se pretende realizar una interface que maneje las 4 operaciones básicas CRUD (Create, Read, Update, Delete) con respecto a la base de datos. Aquí, el actor Profesor tiene permiso total para manejar la información completa de Talleres (es decir todo el CRUD).

F19: Gestionar Asignación de Profesores por Asignatura

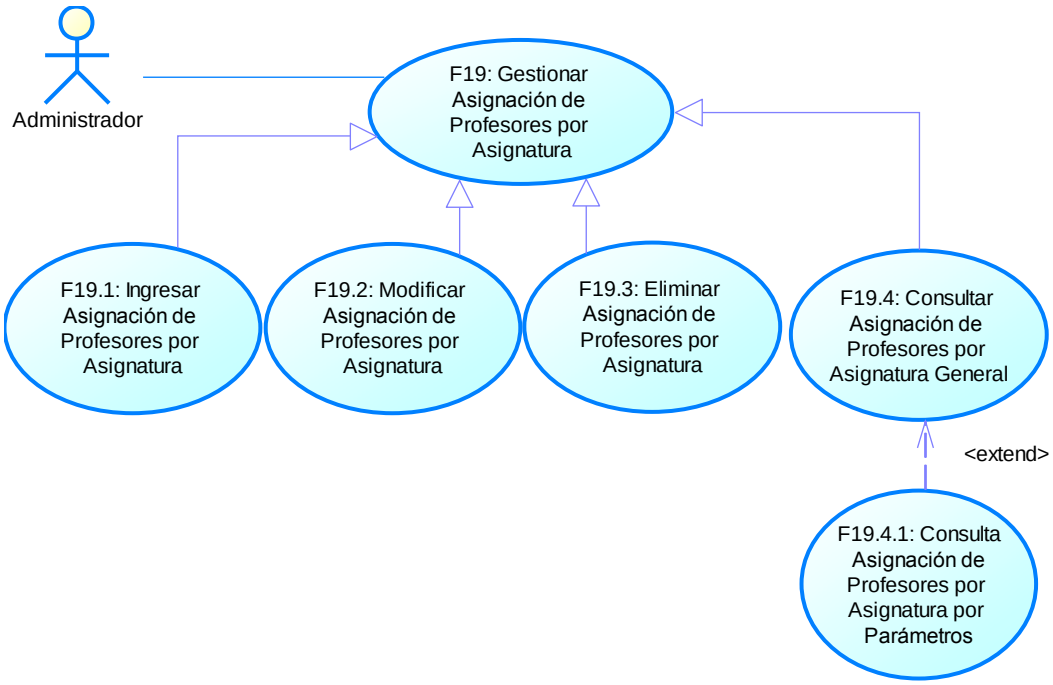


Figura 34. Diagrama C.U. Funcionalidad 19: Gestionar Asignación de Profesores por Asignatura

El Diagrama anterior permite visualizar la estructura externa que posee el proceso de Asignación de Profesores por Asignatura. Según lo observado, se pretende realizar una interface que maneje las 4 operaciones básicas CRUD (Create, Read, Update, Delete) con respecto a la base de datos. Aquí, el actor Administrador tiene permiso total para manejar la información completa del proceso (es decir todo el CRUD). El proceso tiene como propósito asignar las Asignaturas existentes en la Universidad a los Profesores que las dictan. Para que un Profesor pueda ser asignado una materia, es necesario acudir al Administrador del sistema.

F20: Gestionar Asignación de Estudiantes por Asignatura

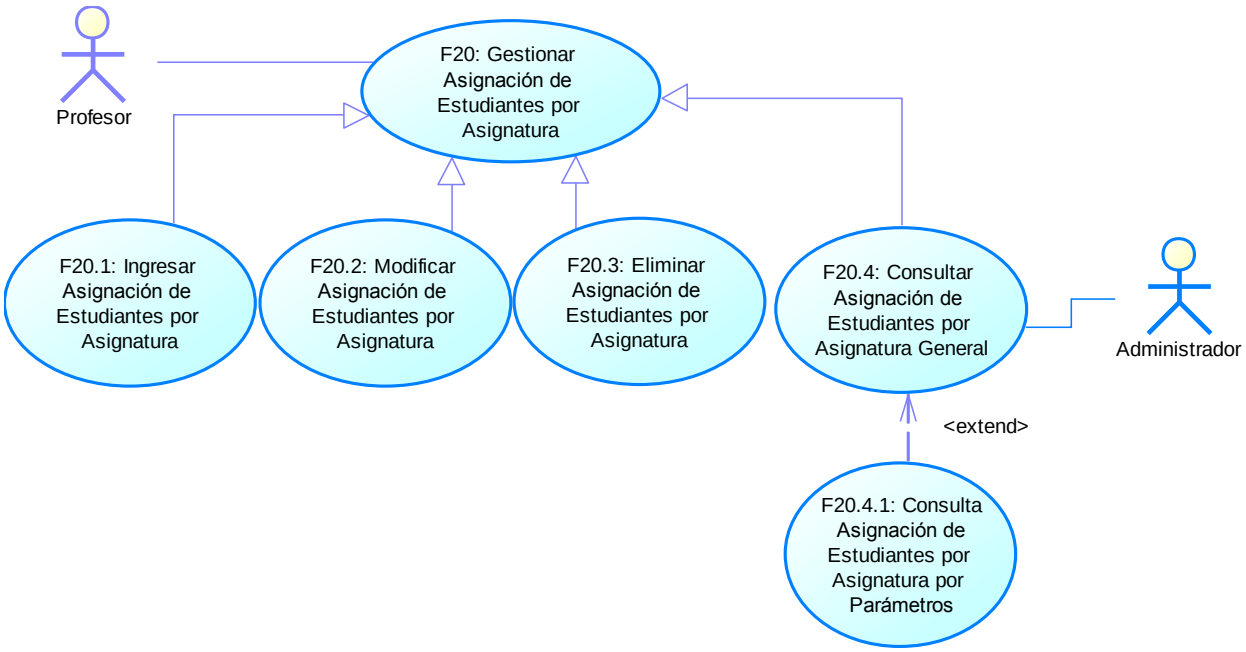


Figura 35. Diagrama C.U. Funcionalidad 20: Gestionar Asignación de Estudiantes por Asignatura

El Diagrama anterior permite visualizar la estructura externa que posee el proceso de Asignación de Estudiantes por Asignatura. Según lo observado, se pretende realizar una interface que maneje las 4 operaciones básicas CRUD (Create, Read, Update, Delete) con respecto a la base de datos. Aquí, el actor Profesor tiene permiso total para manejar la información completa del proceso (es decir todo el CRUD). El proceso tiene como propósito asignar las Asignaturas existentes en una Universidad a los Estudiantes que la reciben. Para que un Estudiante pueda ser asignado una materia, es necesario que el Profesor relacione sus Estudiantes a su materia. Sin embargo, el Administrador es capaz de consultar los Estudiantes Asignados a su Asignatura por Parámetros.

F21: Gestionar Asignación de Facultades por Sede

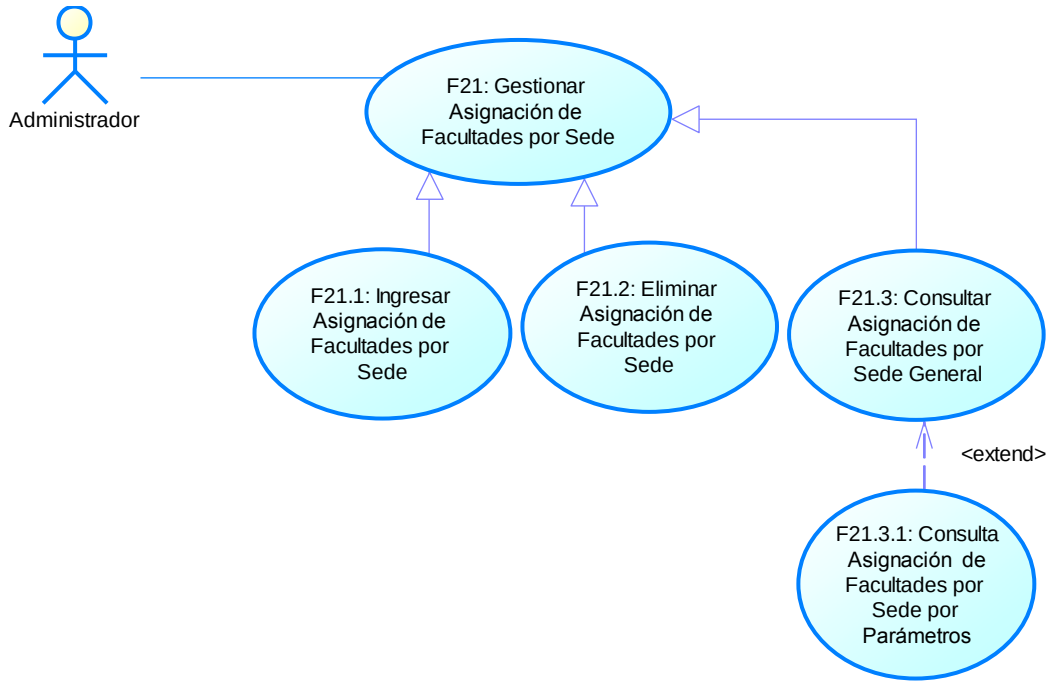


Figura 36. Diagrama C.U. Funcionalidad 21: Gestionar Asignación de Facultades por Sede

El Diagrama anterior permite visualizar la estructura externa que posee el proceso de Asignación de Facultades por Sede. Según lo observado, se pretende realizar una interface que maneje las 3 de las 4 operaciones básicas CRUD (Create, Read, Update, Delete), en este caso no requiere de la opción Update. El actor Administrador tiene permiso total para manejar la información completa del proceso. El proceso tiene como propósito asignar Facultades existentes a una Sede.

F22: Gestionar Estudiante Excel

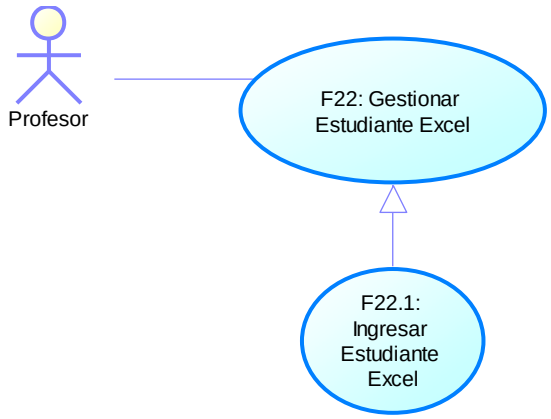


Figura 37. Diagrama C.U. Funcionalidad 22: Gestionar Estudiante Excel

El Diagrama anterior permite visualizar la estructura externa que posee la clase básica Estudiante Excel. Según lo observado, se pretende realizar una interface que permita el ingreso de múltiples Estudiantes por medio de un archivo Excel. El actor Profesor tiene permiso total para manejar la información completa de Estudiantes.

Fase de Diseño

Diagramas de Clases

Diagrama de Clases General

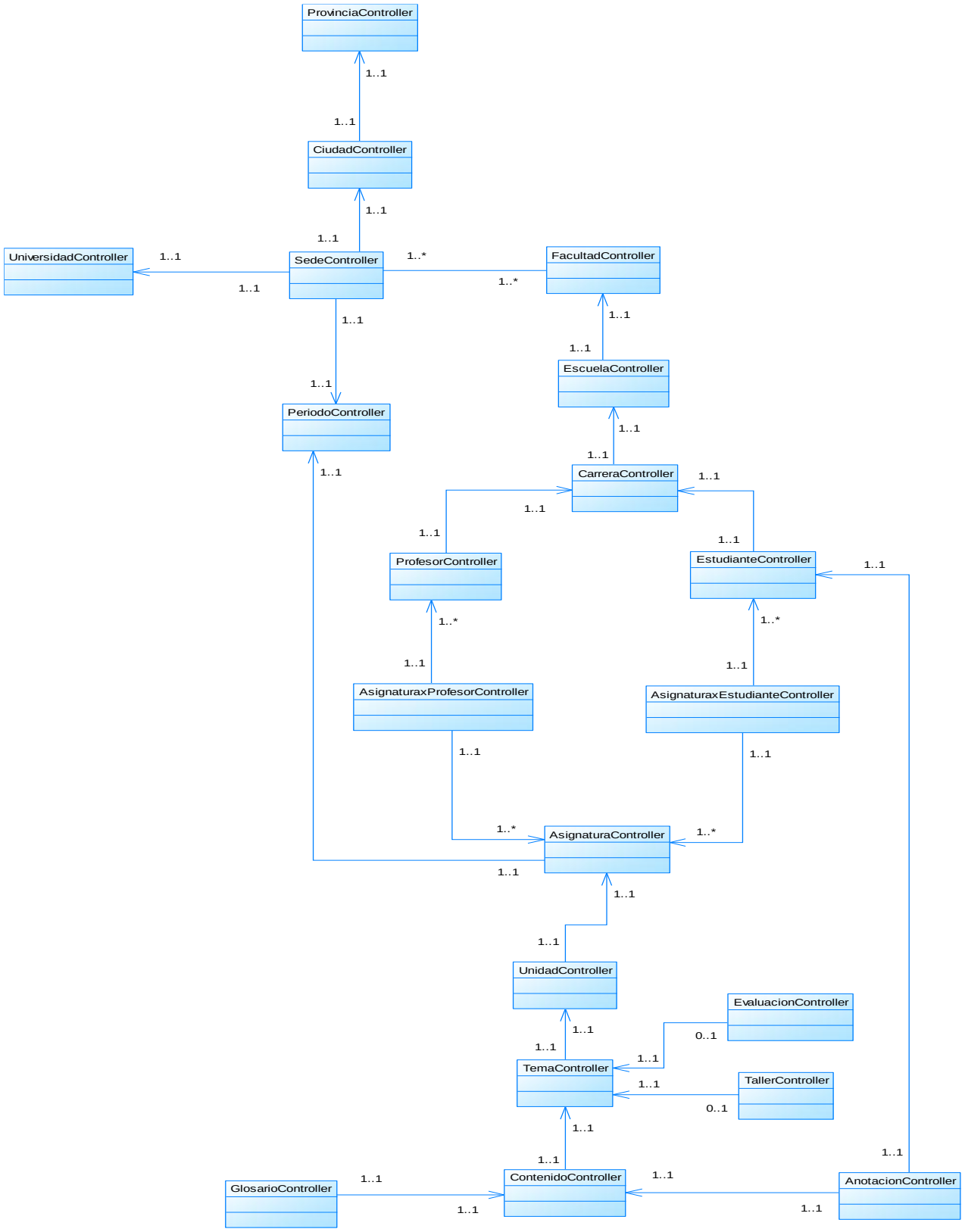


Figura 38. Diagrama de Clases General

Como se puede observar en la figura anterior, en el diagrama se encuentran detalladas todas las clases básicas y procesos que componen al sistema de forma general (es decir, no se tienen aún detallados los atributos ni métodos de cada clase). Cada clase representa un módulo funcional CRUD (Create, Read, Update y Delete) que permite gestionar los registros de la base de datos de la Agenda Electrónica del Estudiante. La idea de tener clases es trabajar con la Metodología Orientada a Objetos para manejar objetos relacionados que funcionen como entradas para que los procesos generen salidas; por ejemplo: el proceso de Asignaturas por Estudiantes está compuesto por las clases básicas Profesor, Estudiante y Asignatura. Este proceso se encarga de manejar las actividades de asignación de Estudiantes a una Respectiva Asignatura que a su vez es dictada por un Profesor. En base a este proceso se gestionan los recursos de la Agenda Electrónica y se asignan Contenidos.

Diagrama de Clases Detallado

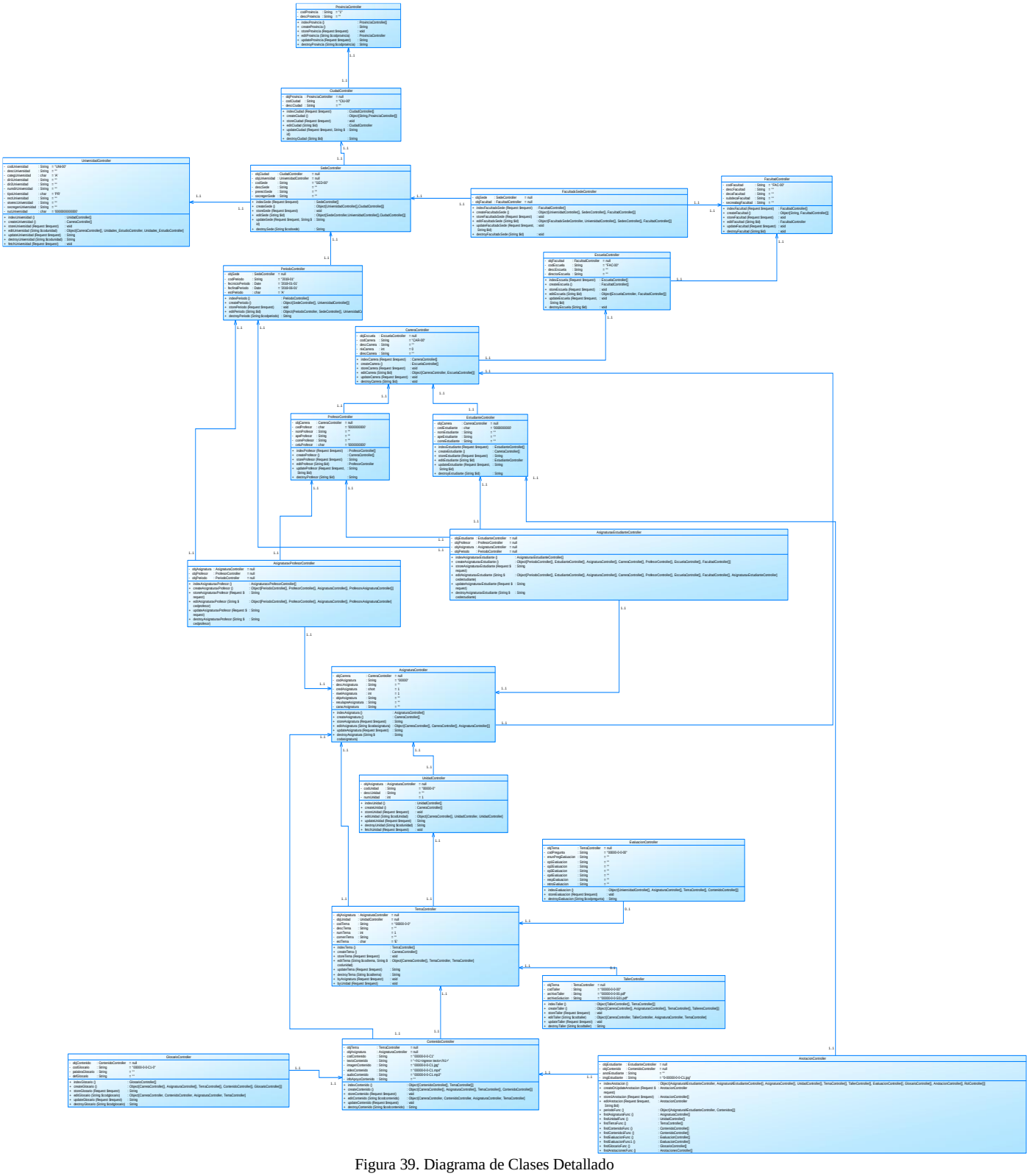


Figura 39. Diagrama de Clases Detallado

Como se puede observar en la figura anterior, en el diagrama se encuentran detalladas todas las clases básicas y procesos que componen al sistema de forma detallada (es decir, se tienen detallados los atributos y los métodos de cada clase). Cada clase representa un módulo funcional CRUD (Create, Read, Update y Delete) que permite gestionar los registros de la base de datos de la Agenda Electrónica del Estudiante; cada método que compone a las clases permite realizar las operaciones CRUD y los atributos funcionan como variables o constantes que definen a la clase y que proporcionan identidad al objeto que es definido. La idea de tener clases es trabajar con la Metodología Orientada a Objetos para manejar objetos relacionados que funcionen como entradas para que los procesos generen salidas; por ejemplo: el proceso de Asignaturas por Estudiantes está compuesto por las clases básicas Profesor, Estudiante y Asignatura. Este proceso se encarga de manejar las actividades de asignación de Estudiantes a una Respectiva Asignatura que a su vez es dictada por un Profesor. En base a este proceso se gestionan los recursos de la Agenda Electrónica y se asignan Contenidos.

Diagrama de Paquetes

Diagrama de Paquetes por Dependencia

Diagrama General

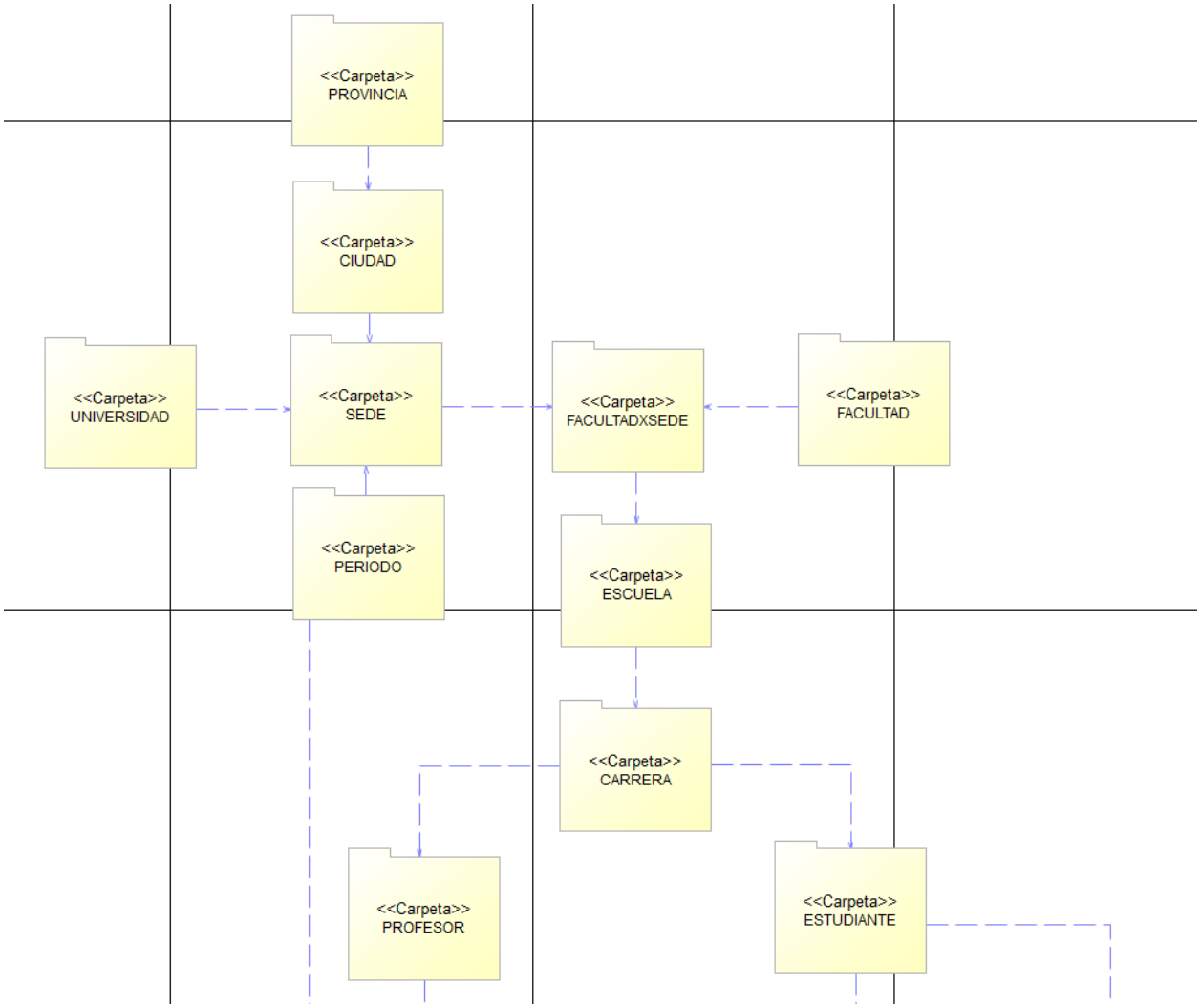


Figura 40. Diagrama de Paquetes por Dependencia General 1

En el diagrama anterior se puede observar la primera parte que compone al Diagrama de Paquetes por Dependencia General del Sistema, la idea de este Diagrama es mostrar cómo se encuentra estructurada la separación de carpetas por módulos dentro del sistema; eso sí, sin olvidar que cada una de las carpetas contiene la lógica de programación web de cada módulo separada por capas (Controladores(DP),

Vistas/Blades(GUI) y Modelos(MD)). Además de que cada carpeta se encuentra en constante comunicación mediante relaciones de asociación directa (flechas entrecortadas que demuestran la dependencia y comunicación de una carpeta con otra).

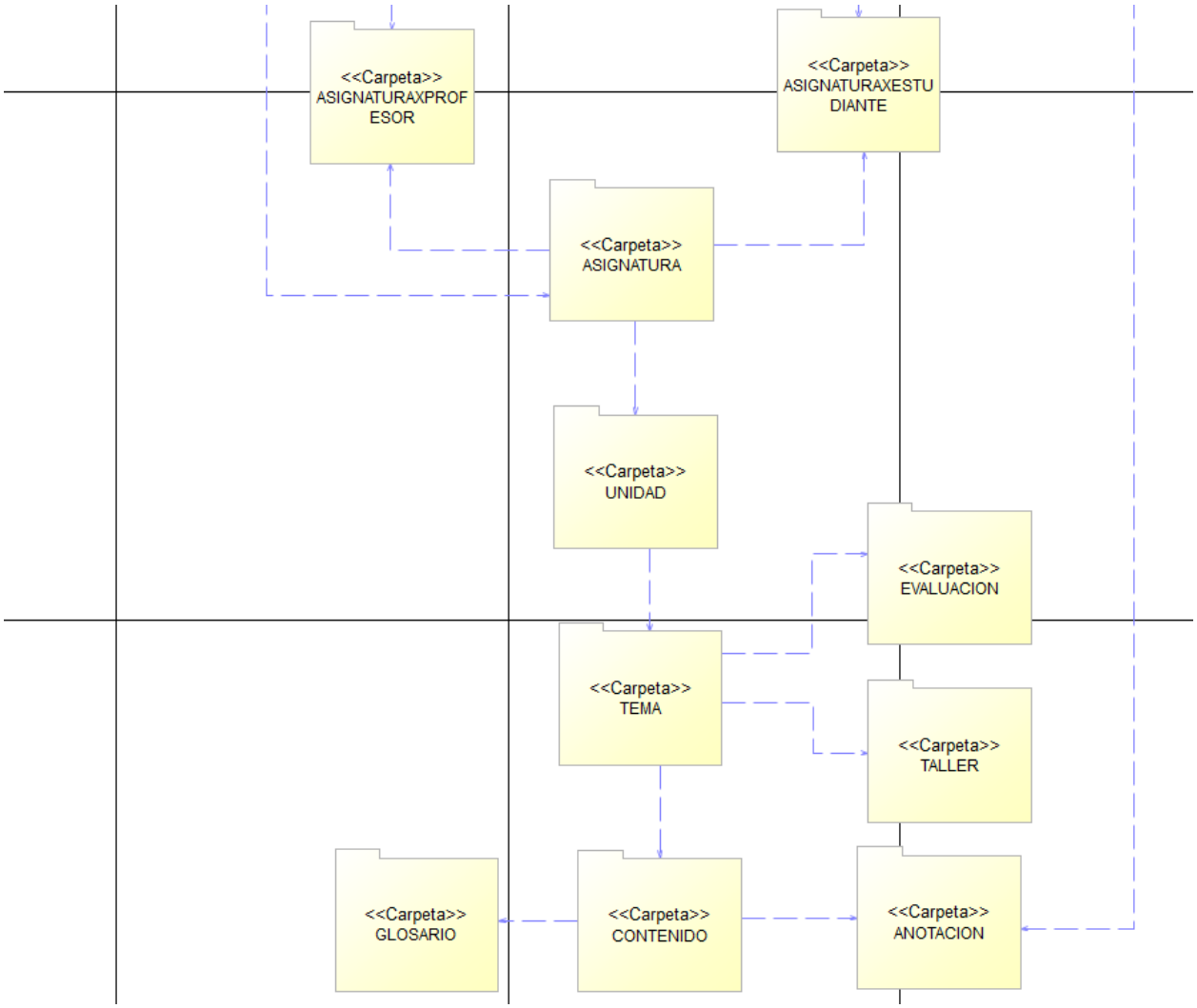


Figura 41. Diagrama de Paquetes por Dependencia General 2

En el diagrama anterior se puede observar la segunda parte que compone al Diagrama de Paquetes por Dependencia General del Sistema, la idea de este Diagrama es mostrar cómo se encuentra estructurada la separación de carpetas por módulos dentro del sistema; eso sí, sin olvidar que cada una de las carpetas contiene la lógica de programación web de cada módulo separada por capas (Controladores(DP),

Vistas/Blades(GUI) y Modelos(MD)). Además de que cada carpeta se encuentra en constante comunicación mediante relaciones de asociación directa (flechas entrecortadas que demuestran la dependencia y comunicación de una carpeta con otra).

Diagramas Detallados

Los diagramas detallados muestran el contenido de cada carpeta de los Diagramas de Paquetes por Dependencia Generales; es decir, cada módulo detallado con sus respectivas capas (Controladores(DP), Vistas/Blades(GUI) y Modelos(MD)). La estructura general se puede observar a continuación en la siguiente figura:

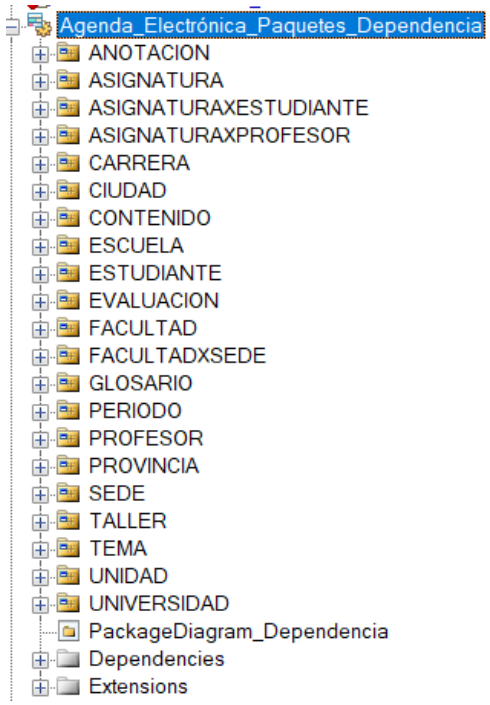


Figura 42. Diagrama de Paquetes por Dependencia General con Detalle de Subcarpetas

Cada carpeta contiene sus propias dependencias (relaciones), su diagrama de paquetes y sus extensiones que nombran al archivo (nombre de archivo).

Anotación

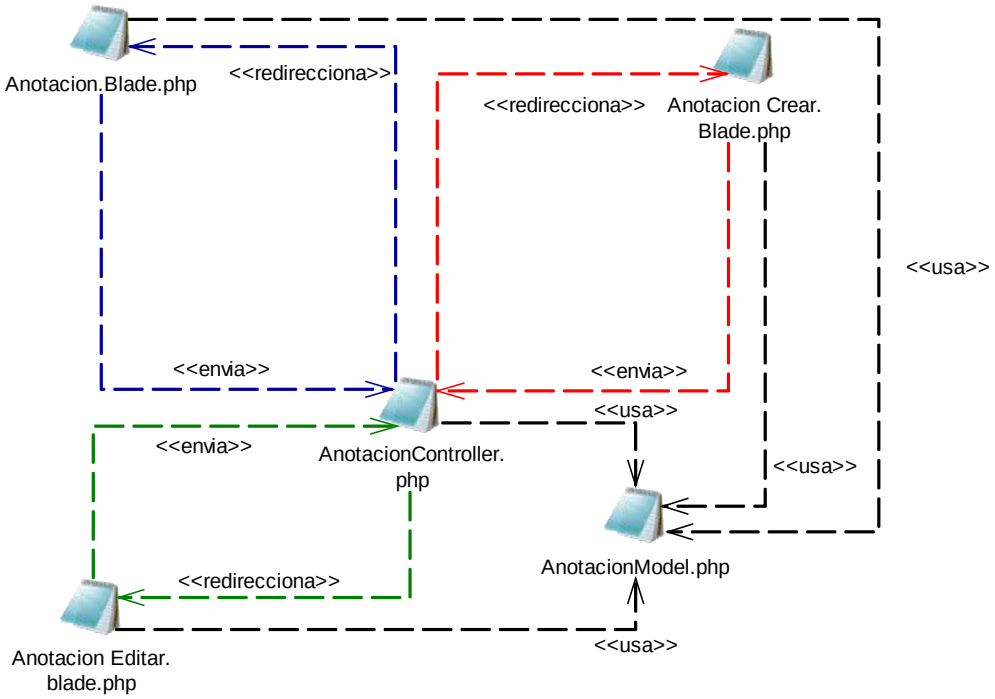


Figura 43. Diagrama de Paquetes por Dependencia Anotación Detallado

Como se puede observar en la figura anterior, el módulo Anotación tiene una estructura definida de comunicación entre capas. Aquí se muestran tres tipos de capas que interactúan entre sí: la capa GUI que tiene una extensión .Blade.php (esta es la capa de visualización que presenta y gestiona datos para interactuar con el usuario con respecto a Anotaciones en un entorno gráfico), la capa DP o Controller, tiene la extensión .php (esta es la capa de lógica del negocio/desarrollo, es decir, controla los métodos de la clase Anotaciones que son invocados por los Blade, los cuales envían datos recibidos por el usuario en las interfaces y que a su vez usan Model para comunicarse con la base de datos y gestionar los registros de la base de datos), y por último, la capa MD o Model, tiene una extensión .php (esta es la capa de manejo de datos que se comunica directamente con la base de datos del sistema y permite realizar instrucciones SQL con respecto a la tabla Anotaciones).

Cabe recalcar que las dependencias/relaciones son de suma importancia para conocer la función que cumple cada capa con respecto a otra, o desde otro punto de vista como interactúan entre ellas. Por ejemplo: el Blade llamado Anotación Crear.Blade.php es <<redireccionado>> y desplegado al usuario a través de una llamada desde la interface Anotación.Blade.php al presionar el botón nuevo registro. Aquí, el Usuario interactúa con la interface ingresando por teclado y seleccionando los datos que sirven para agregar una nueva Anotación en base a lo solicitado por las etiquetas (tags de la página web) y por detrás se <<envía>> estos datos; los cuales posteriormente serán manejados en el Controlador AnotaciónController.php con su función Store que con ayuda del modelo que <<usado>> por el controlador permita ingresar un registro de Anotación en la Base de Datos del Sistema tabla Anotaciones.

Asignatura

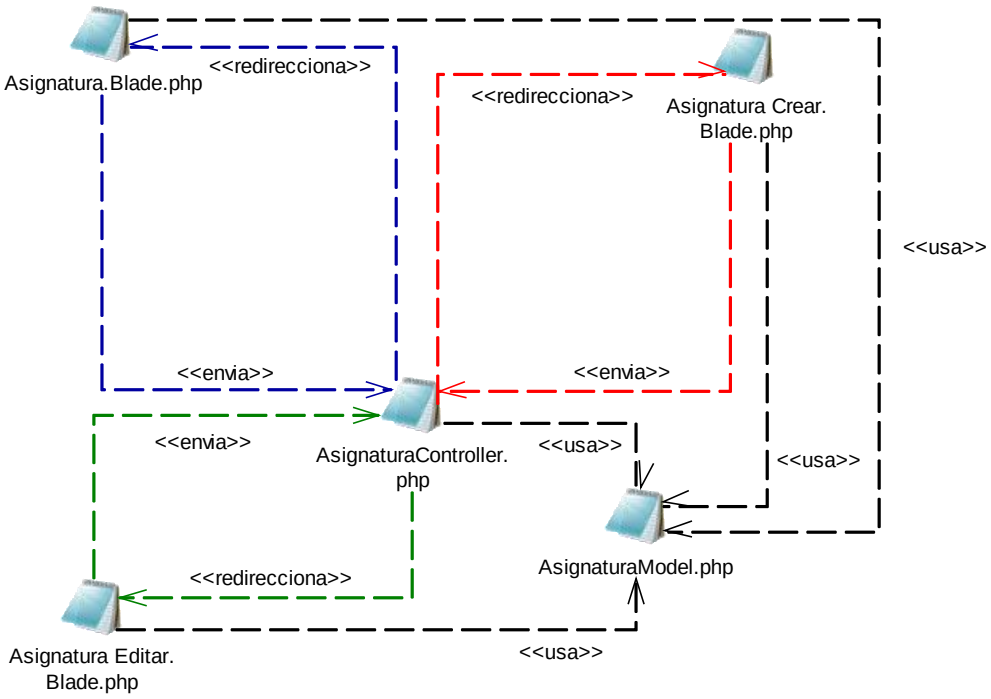


Figura 44. Diagrama de Paquetes por Dependencia Asignatura Detallado

Como se puede observar en la figura anterior, el módulo Asignatura tiene una estructura definida de comunicación entre capas. Aquí se muestran tres tipos de capas que interactúan entre sí: la capa GUI que tiene una extensión .Blade.php (esta es la capa de visualización que presenta y gestiona datos para interactuar con el usuario con respecto a Asignaturas en un entorno gráfico), la capa DP o Controller, tiene la extensión .php (esta es la capa de lógica del negocio/desarrollo, es decir, controla los métodos de la clase Asignaturas que son invocados por los Blade, los cuales envían datos recibidos por el usuario en las interfaces y que a su vez usan Model para comunicarse con la base de datos y gestionar los registros de la base de datos), y por último, la capa MD o Model, tiene una extensión .php (esta es la capa de manejo de datos que se comunica directamente con la base de datos del sistema y permite realizar instrucciones SQL con respecto a la tabla Asignaturas).

Cabe recalcar que las dependencias/relaciones son de suma importancia para conocer la función que cumple cada capa con respecto a otra, o desde otro punto de vista como interactúan entre ellas. Por ejemplo: el Blade llamado Asignatura Crear.Blade.php es <<redireccionado>> y desplegado al usuario a través de una llamada desde la interface Asignatura.Blade.php al presionar el botón nuevo registro. Aquí, el Usuario interactúa con la interface ingresando por teclado y seleccionando los datos que sirven para agregar una nueva Asignatura en base a lo solicitado por las etiquetas (tags de la página web) y por detrás se <<envía>> estos datos; los cuales posteriormente serán manejados en el Controlador AsignaturaController.php con su función Store que con ayuda del modelo que <<usado>> por el controlador permita ingresar un registro de Asignatura en la Base de Datos del Sistema tabla Asignaturas.

Asignatura por Estudiante

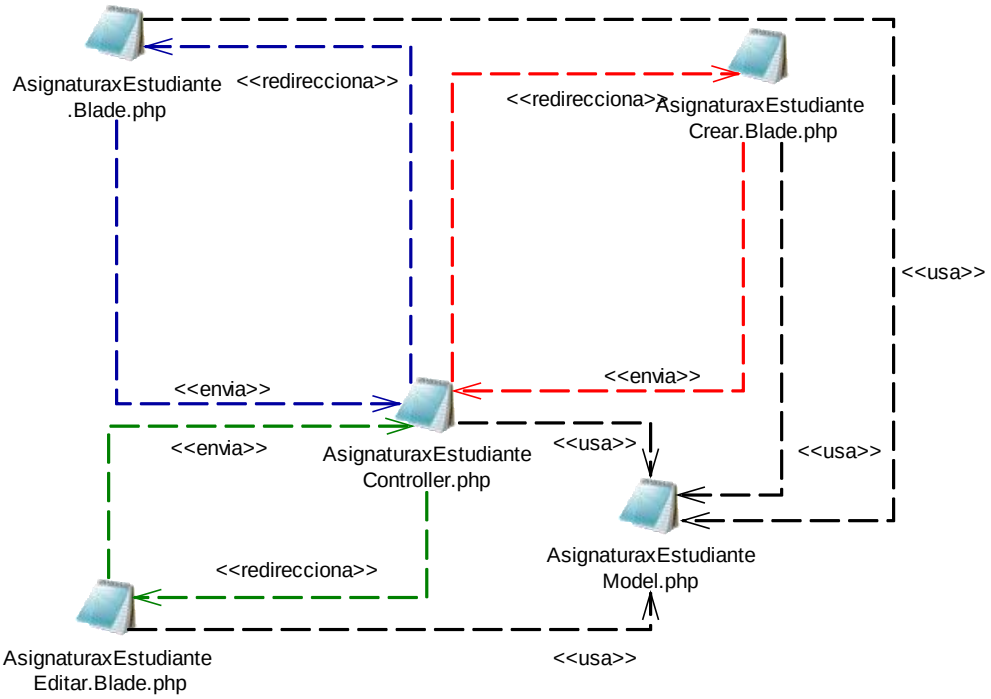


Figura 45. Diagrama de Paquetes por Dependencia AsignaturaxEstudiante Detallado

Como se puede observar en la figura anterior, el módulo AsignaturaxEstudiante tiene una estructura definida de comunicación entre capas. Aquí se muestran tres tipos de capas que interactúan entre sí: la capa GUI que tiene una extensión .Blade.php (esta es la capa de visualización que presenta y gestiona datos para interactuar con el usuario con respecto a AsignaturaxEstudiante en un entorno gráfico), la capa DP o Controller, tiene la extensión .php (esta es la capa de lógica del negocio/desarrollo, es decir, controla los métodos de la clase AsignaturaxEstudiante que son invocados por los Blade, los cuales envían datos recibidos por el usuario en las interfaces y que a su vez usan Model para comunicarse con la base de datos y gestionar los registros de la base de datos), y por último, la capa MD o Model, tiene una extensión .php (esta es la capa de manejo de datos que se

comunica directamente con la Base de Datos del Sistema y permite realizar instrucciones SQL con respecto a la tabla AsignaturasxEstudiantes).

Cabe recalcar que las dependencias/relaciones son de suma importancia para conocer la función que cumple cada capa con respecto a otra, o desde otro punto de vista como interactúan entre ellas. Por ejemplo: el Blade llamado AsignaturaxEstudiante Crear.Blade.php es <<redireccionado>> y desplegado al usuario a través de una llamada desde la interface AsignaturaxEstudiante.Blade.php al presionar el botón nuevo registro. Aquí, el Usuario interactúa con la interface ingresando por teclado y seleccionando los datos que sirven para agregar una nueva AsignaturaxEstudiante en base a lo solicitado por las etiquetas (tags de la página web) y por detrás se <<envía>> estos datos; los cuales posteriormente serán manejados en el Controlador AsignaturaxEstudianteController.php con su función Store que con ayuda del modelo que <<usado>> por el controlador permita ingresar un registro de AsignaturaxEstudiante en la Base de Datos del Sistema tabla AsignaturasxEstudiantes.

Asignatura por Profesor

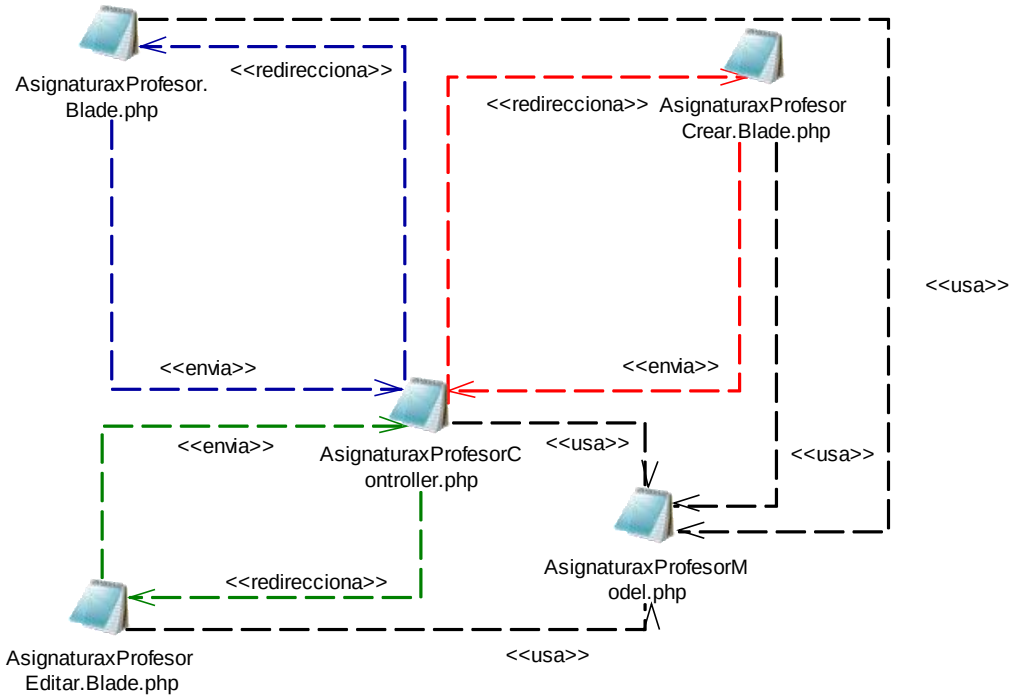


Figura 46. Diagrama de Paquetes por Dependencia AsignaturaxProfesor Detallado

Como se puede observar en la figura anterior, el módulo AsignaturaxProfesor tiene una estructura definida de comunicación entre capas. Aquí se muestran tres tipos de capas que interactúan entre sí: la capa GUI que tiene una extensión .Blade.php (esta es la capa de visualización que presenta y gestiona datos para interactuar con el usuario con respecto a AsignaturaxProfesor en un entorno gráfico), la capa DP o Controller, tiene la extensión .php (esta es la capa de lógica del negocio/desarrollo, es decir, controla los métodos de la clase AsignaturaxProfesor que son invocados por los Blade, los cuales envían datos recibidos por el usuario en las interfaces y que a su vez usan Model para comunicarse con la base de datos y gestionar los registros de la base de datos), y por último, la capa MD o Model, tiene una extensión .php (esta es la capa de manejo de datos que se comunica directamente con la base de datos del sistema y permite realizar instrucciones SQL con respecto a la tabla AsignaturasxProfesores).

Cabe recalcar que las dependencias/relaciones son de suma importancia para conocer la función que cumple cada capa con respecto a otra, o desde otro punto de vista como interactúan entre ellas. Por ejemplo: el Blade llamado AsignaturaxProfesor Crear.Blade.php es <<redireccionado>> y desplegado al usuario a través de una llamada desde la interface AsignaturaxProfesor.Blade.php al presionar el botón nuevo registro. Aquí, el Usuario interactúa con la interface ingresando por teclado y seleccionando los datos que sirven para agregar una nueva AsignaturaxProfesor en base a lo solicitado por las etiquetas (tags de la página web) y por detrás se <<envía>> estos datos; los cuales posteriormente serán manejados en el Controlador AsignaturaxProfesorController.php con su función Store que con ayuda del modelo que <<usado>> por el controlador permita ingresar un registro de AsignaturaxProfesor en la Base de Datos del Sistema tabla AsignaturasxProfesores.

Carrera

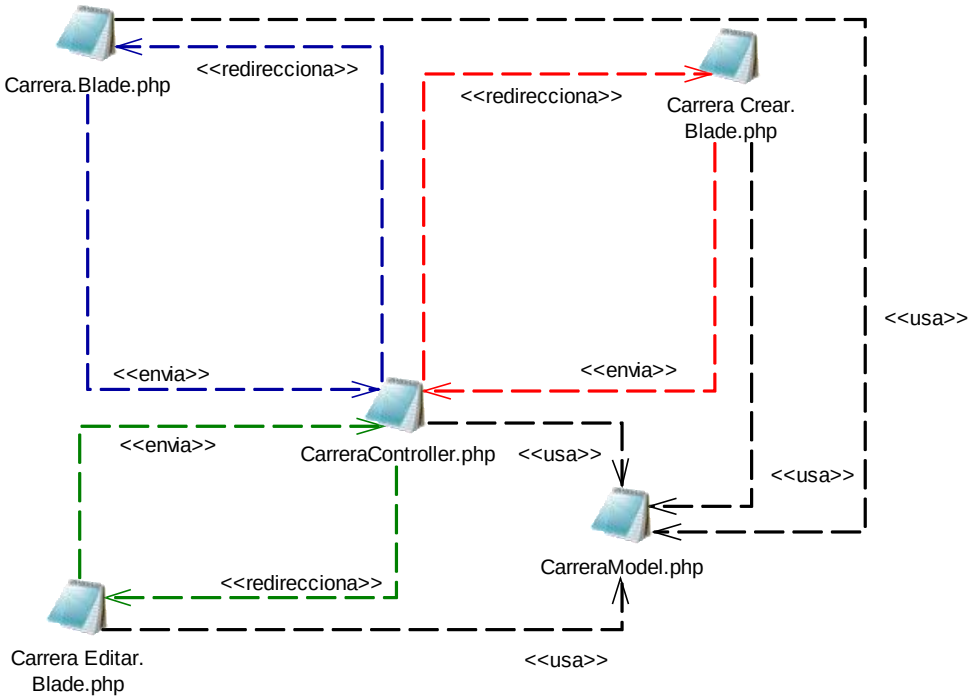


Figura 47. Diagrama de Paquetes por Dependencia Carrera Detallado

Como se puede observar en la figura anterior, el módulo Carrera tiene una estructura definida de comunicación entre capas. Aquí se muestran tres tipos de capas que interactúan entre sí: la capa GUI que tiene una extensión .Blade.php (esta es la capa de visualización que presenta y gestiona datos para interactuar con el usuario con respecto a Carrera en un entorno gráfico), la capa DP o Controller, tiene la extensión .php (esta es la capa de lógica del negocio/desarrollo, es decir, controla los métodos de la clase Carrera que son invocados por los Blade, los cuales envían datos recibidos por el usuario en las interfaces y que a su vez usan Model para comunicarse con la base de datos y gestionar los registros de la base de datos), y por último, la capa MD o Model, tiene una extensión .php (esta es la capa de manejo de datos que se comunica directamente con la base de datos del sistema y permite realizar instrucciones SQL con respecto a la tabla Carreras).

Cabe recalcar que las dependencias/relaciones son de suma importancia para conocer la función que cumple cada capa con respecto a otra, o desde otro punto de vista como interactúan entre ellas. Por ejemplo: el Blade llamado Carrera Crear.Blade.php es <<redireccionado>> y desplegado al usuario a través de una llamada desde la interface Carrera.Blade.php al presionar el botón nuevo registro. Aquí, el Usuario interactúa con la interface ingresando por teclado y seleccionando los datos que sirven para agregar una nueva Carrera en base a lo solicitado por las etiquetas (tags de la página web) y por detrás se <<envía>> estos datos; los cuales posteriormente serán manejados en el Controlador CarreraController.php con su función Store que con ayuda del modelo que <<usado>> por el controlador permita ingresar un registro de Carrera en la Base de Datos del Sistema tabla Carreras.

Ciudad

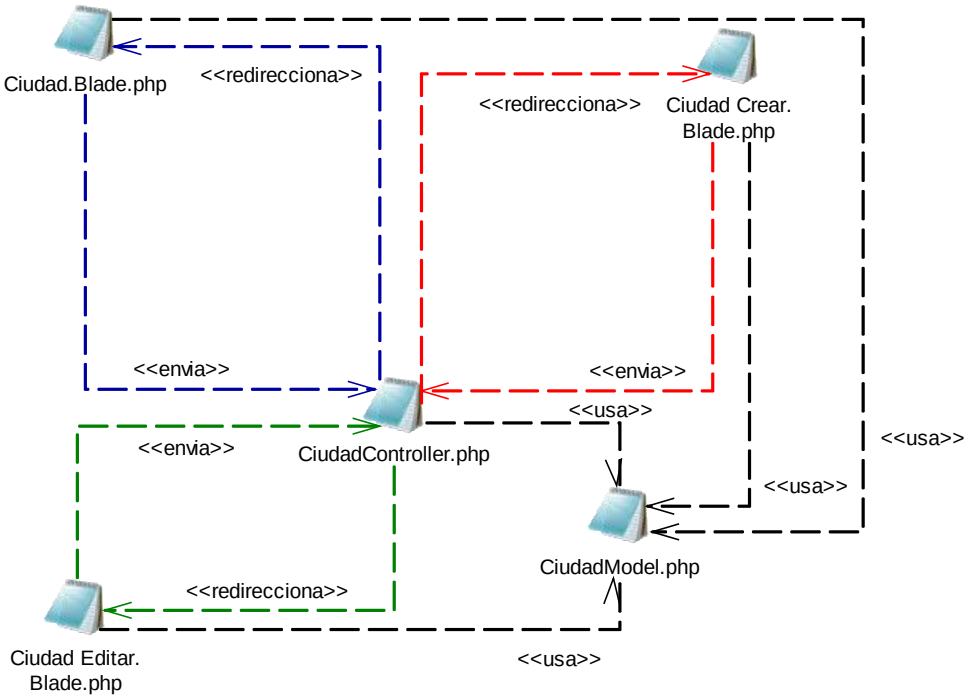


Figura 48. Diagrama de Paquetes por Dependencia Ciudad Detallado

Como se puede observar en la figura anterior, el módulo Ciudad tiene una estructura definida de comunicación entre capas. Aquí se muestran tres tipos de capas que interactúan entre sí: la capa GUI que tiene una extensión .Blade.php (esta es la capa de visualización que presenta y gestiona datos para interactuar con el usuario con respecto a Ciudad en un entorno gráfico), la capa DP o Controller, tiene la extensión .php (esta es la capa de lógica del negocio/desarrollo, es decir, controla los métodos de la clase Ciudad que son invocados por los Blade, los cuales envían datos recibidos por el usuario en las interfaces y que a su vez usan Model para comunicarse con la base de datos y gestionar los registros de la base de datos), y por último, la capa MD o Model, tiene una extensión .php (esta es la capa de manejo de datos que se comunica directamente con la base de datos del sistema y permite realizar instrucciones SQL con respecto a la tabla Ciudades).

Cabe recalcar que las dependencias/relaciones son de suma importancia para conocer la función que cumple cada capa con respecto a otra, o desde otro punto de vista como interactúan entre ellas. Por ejemplo: el Blade llamado Ciudad Crear.Blade.php es <<redireccionado>> y desplegado al usuario a través de una llamada desde la interface Ciudad.Blade.php al presionar el botón nuevo registro. Aquí, el Usuario interactúa con la interface ingresando por teclado y seleccionando los datos que sirven para agregar una nueva Ciudad en base a lo solicitado por las etiquetas (tags de la página web) y por detrás se <<envía>> estos datos; los cuales posteriormente serán manejados en el Controlador CiudadController.php con su función Store que con ayuda del modelo que <<usado>> por el controlador permita ingresar un registro de Ciudad en la Base de Datos del Sistema tabla Ciudades.

Contenido

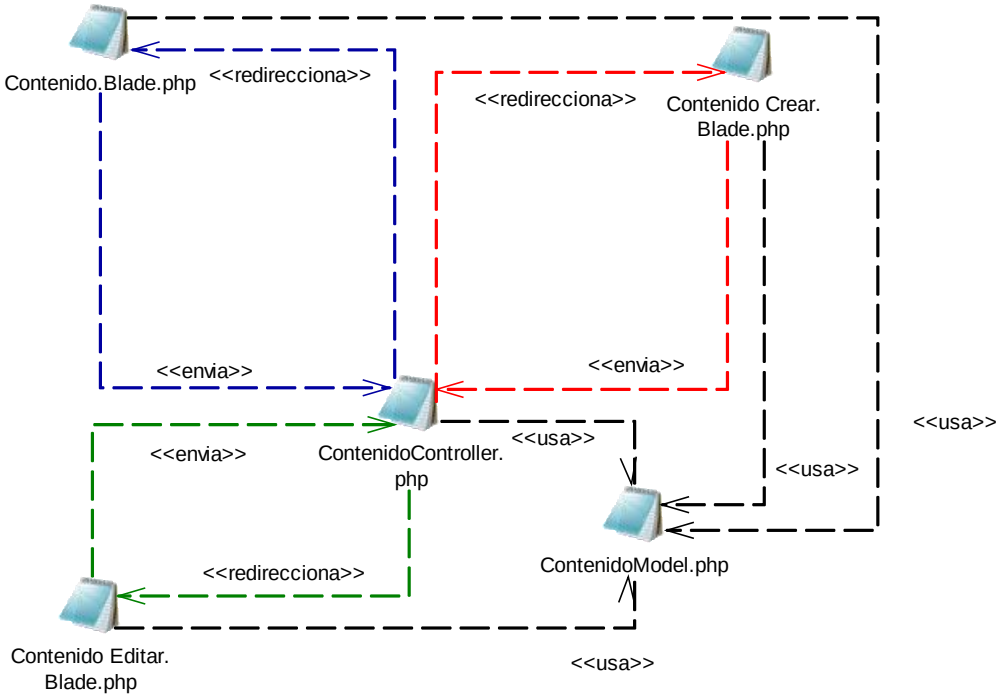


Figura 49. Diagrama de Paquetes por Dependencia Contenido Detallado

Como se puede observar en la figura anterior, el módulo Contenido tiene una estructura definida de comunicación entre capas. Aquí se muestran tres tipos de capas que interactúan entre sí: la capa GUI que tiene una extensión .Blade.php (esta es la capa de visualización que presenta y gestiona datos para interactuar con el usuario con respecto a Contenido en un entorno gráfico), la capa DP o Controller, tiene la extensión .php (esta es la capa de lógica del negocio/desarrollo, es decir, controla los métodos de la clase Contenido que son invocados por los Blade, los cuales envían datos recibidos por el usuario en las interfaces y que a su vez usan Model para comunicarse con la base de datos y gestionar los registros de la base de datos), y por último, la capa MD o Model, tiene una extensión .php (esta es la capa de manejo de datos que se comunica directamente con la base de datos del sistema y permite realizar instrucciones SQL con respecto a la tabla Contenidos).

Cabe recalcar que las dependencias/relaciones son de suma importancia para conocer la función que cumple cada capa con respecto a otra, o desde otro punto de vista como interactúan entre ellas. Por ejemplo: el Blade llamado Contenido Crear.Blade.php es <<redireccionado>> y desplegado al usuario a través de una llamada desde la interface Contenido.Blade.php al presionar el botón nuevo registro. Aquí, el Usuario interactúa con la interface ingresando por teclado y seleccionando los datos que sirven para agregar un nuevo Contenido en base a lo solicitado por las etiquetas (tags de la página web) y por detrás se <<envía>> estos datos; los cuales posteriormente serán manejados en el Controlador ContenidoController.php con su función Store que con ayuda del modelo que <<usado>> por el controlador permita ingresar un registro de Contenido en la Base de Datos del Sistema tabla Contenidos.

Escuela

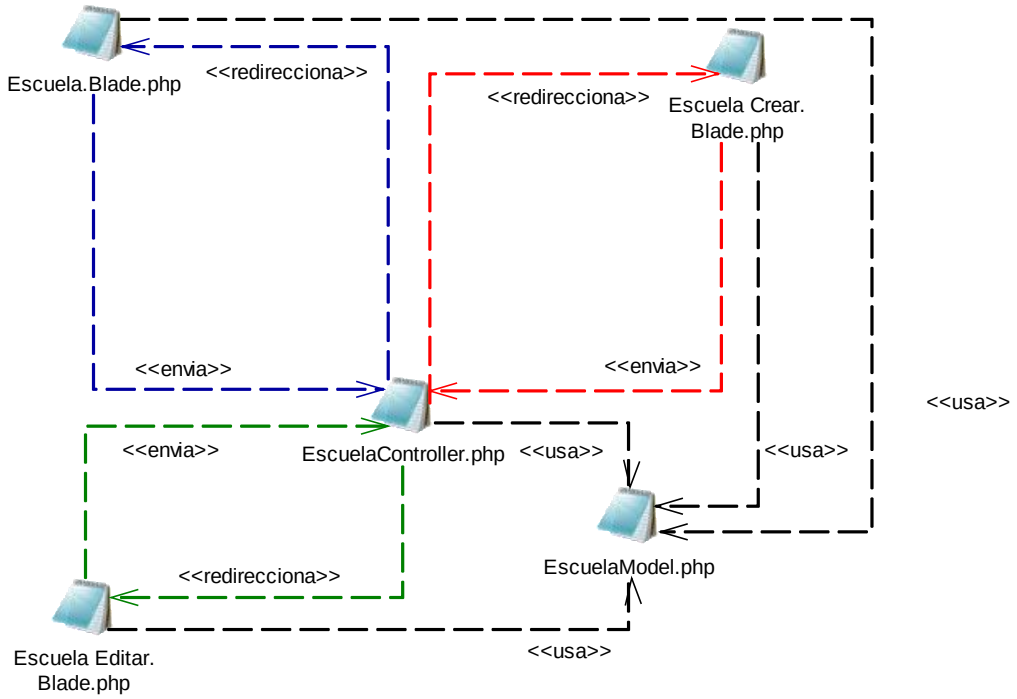


Figura 50. Diagrama de Paquetes por Dependencia Escuela Detallado

Como se puede observar en la figura anterior, el módulo Escuela tiene una estructura definida de comunicación entre capas. Aquí se muestran tres tipos de capas que interactúan entre sí: la capa GUI que tiene una extensión .Blade.php (esta es la capa de visualización que presenta y gestiona datos para interactuar con el usuario con respecto a Escuela en un entorno gráfico), la capa DP o Controller, tiene la extensión .php (esta es la capa de lógica del negocio/desarrollo, es decir, controla los métodos de la clase Escuela que son invocados por los Blade, los cuales envían datos recibidos por el Usuario en las interfaces y que a su vez usan Model para comunicarse con la base de datos y gestionar los registros de la base de datos), y por último, la capa MD o Model, tiene una extensión .php (esta es la capa de manejo de datos que se comunica directamente con la base de datos del sistema y permite realizar instrucciones SQL con respecto a la tabla Escuelas).

Cabe recalcar que las dependencias/relaciones son de suma importancia para conocer la función que cumple cada capa con respecto a otra, o desde otro punto de vista como interactúan entre ellas. Por ejemplo: el Blade llamado Escuela Crear.Blade.php es <<redireccionado>> y desplegado al Usuario a través de una llamada desde la interface Escuela.Blade.php al presionar el botón nuevo registro. Aquí, el Usuario interactúa con la interface ingresando por teclado y seleccionando los datos que sirven para agregar una nueva Escuela en base a lo solicitado por las etiquetas (tags de la página web) y por detrás se <<envía>> estos datos; los cuales posteriormente serán manejados en el Controlador ContenidoController.php con su función Store que con ayuda del modelo que <<usado>> por el controlador permita ingresar un registro de Escuela en la Base de Datos del Sistema tabla Escuelas.

Estudiante

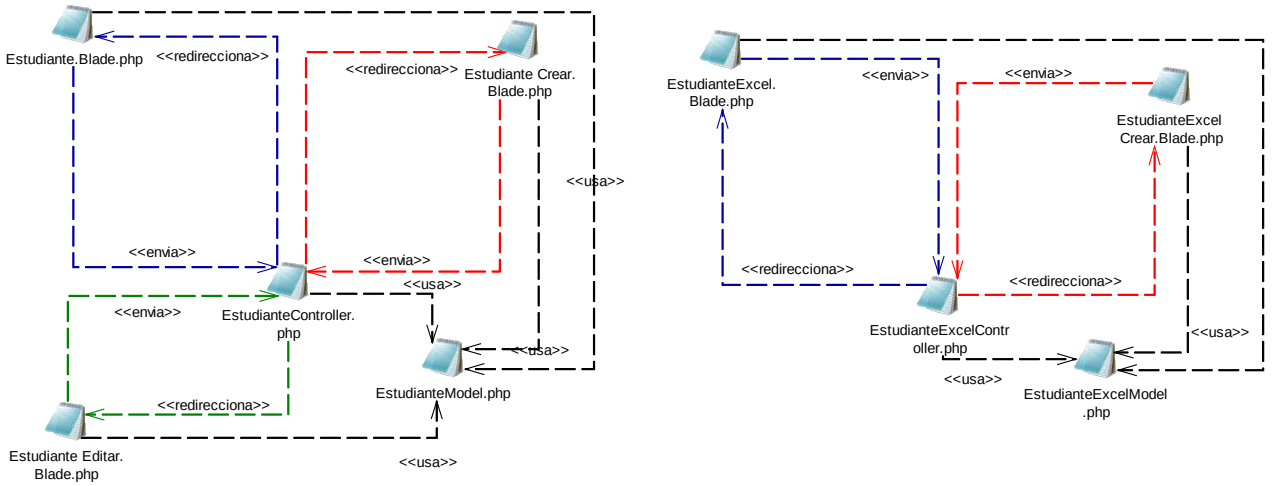


Figura 51. Diagrama de Paquetes por Dependencia Estudiante y EstudianteExcel Detallados

Como se puede observar en la figura anterior, la carpeta de Estudiante contiene dos módulos llamados Estudiante y EstudianteExcel que presentan una estructura definida de comunicación entre capas. Aquí se muestran tres tipos de capas que interactúan entre sí: la capa GUI que tiene una extensión .Blade.php (esta es la capa

de visualización que presenta y gestiona datos para interactuar con el Usuario con respecto a Estudiantes y EstudiantesExcel en un entorno gráfico), la capa DP o Controller, tiene la extensión .php (esta es la capa de lógica del negocio/desarrollo, es decir, controla los métodos de la clase Estudiantes y EstudiantesExcel que son invocados por los Blade, los cuales envían datos recibidos por el usuario en las interfaces y que a su vez usan Model para comunicarse con la base de datos y gestionar los registros de la base de datos), y por último, la capa MD o Model, tiene una extensión .php (esta es la capa de manejo de datos que se comunica directamente con la base de datos del sistema y permite realizar instrucciones SQL con respecto a la tabla Estudiantes y EstudiantesExcel).

Cabe recalcar que las dependencias/relaciones son de suma importancia para conocer la función que cumple cada capa con respecto a otra, o desde otro punto de vista como interactúan entre ellas. Por ejemplo: los Blade llamados Estudiante Crear.Blade.php y EstudianteExcel Crear.Blade.php son <<redireccionados>> y desplegados al usuario a través de una llamada desde las interfaces Estudiante.Blade.php y EstudianteExcel.Blade.php respectivamente, al presionar el botón nuevo registro. Aquí, el Usuario interactúa con las interfaces ingresando por teclado y seleccionando los datos que sirven para agregar uno o varios Estudiantes en base a lo solicitado por las etiquetas (tags de la página web) y por detrás se <<envía>> estos datos; los cuales posteriormente serán manejados por los Controladores EstudianteController.php y EstudianteExcelController.php respectivamente con su función Store, la cual con ayuda del modelo que es <<usado>> por su respectivo controlador permite ingresar uno o varios registros de Estudiantes en la Base de Datos del Sistema en la tabla Estudiantes y EstudiantesExcel respectivamente.

Evaluación

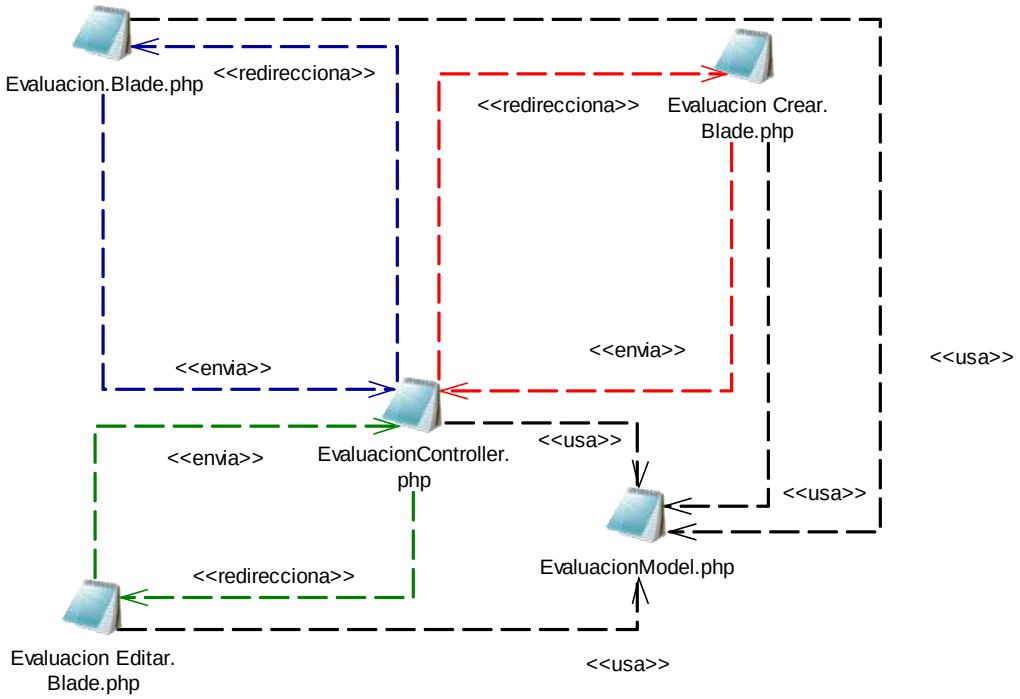


Figura 52. Diagrama de Paquetes por Dependencia Evaluación Detallado

Como se puede observar en la figura anterior, el módulo Evaluación tiene una estructura definida de comunicación entre capas. Aquí se muestran tres tipos de capas que interactúan entre sí: la capa GUI que tiene una extensión .Blade.php (esta es la capa de visualización que presenta y gestiona datos para interactuar con el usuario con respecto a Evaluación en un entorno gráfico), la capa DP o Controller, tiene la extensión .php (esta es la capa de lógica del negocio/desarrollo, es decir, controla los métodos de la clase Evaluación que son invocados por los Blade, los cuales envían datos recibidos por el Usuario en las interfaces y que a su vez usan Model para comunicarse con la base de datos y gestionar los registros de la base de datos), y por último, la capa MD o Model, tiene una extensión .php (esta es la capa de manejo de datos que se comunica directamente con la base de datos del sistema y permite realizar instrucciones SQL con respecto a la tabla Evaluaciones).

Cabe recalcar que las dependencias/relaciones son de suma importancia para conocer la función que cumple cada capa con respecto a otra, o desde otro punto de vista como interactúan entre ellas. Por ejemplo: el Blade llamado Evaluación Crear.Blade.php es <<redireccionado>> y desplegado al Usuario a través de una llamada desde la interface Evaluacion.Blade.php al presionar el botón nuevo registro. Aquí, el Usuario interactúa con la interface ingresando por teclado y seleccionando los datos que sirven para agregar una nueva Evaluación en base a lo solicitado por las etiquetas (tags de la página web) y por detrás se <<envía>> estos datos; los cuales posteriormente serán manejados en el Controlador EvaluacionController.php con su función Store que con ayuda del modelo que <<usado>> por el controlador permita ingresar un registro de Evaluación en la Base de Datos del Sistema tabla Evaluaciones.

Facultad

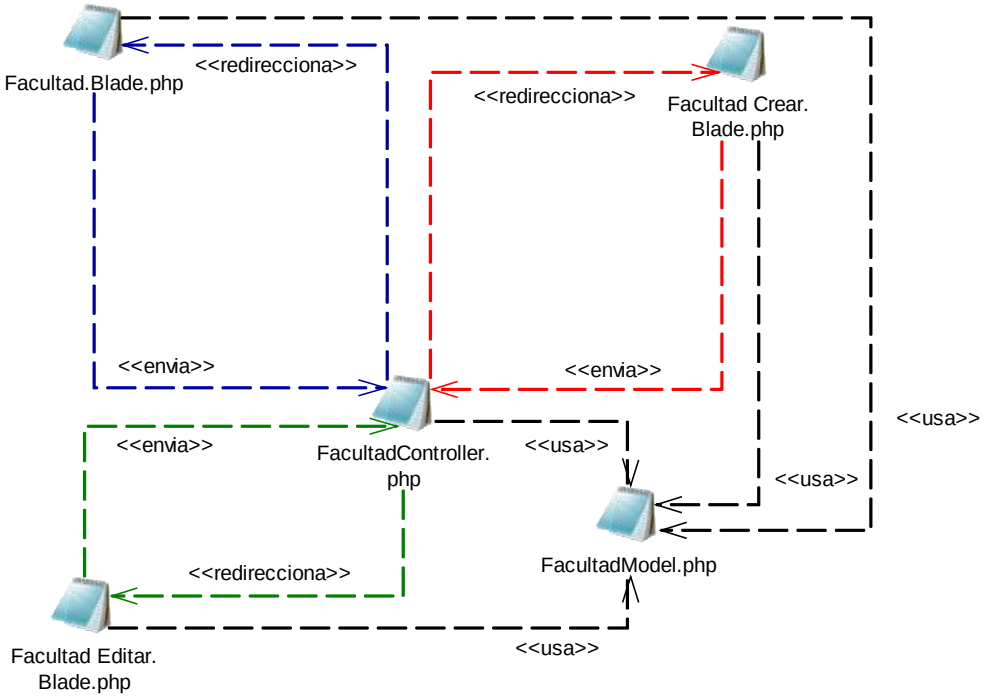


Figura 53. Diagrama de Paquetes por Dependencia Facultad Detallado

Como se puede observar en la figura anterior, el módulo Facultad tiene una estructura definida de comunicación entre capas. Aquí se muestran tres tipos de capas que interactúan entre sí: la capa GUI que tiene una extensión .Blade.php (esta es la capa de visualización que presenta y gestiona datos para interactuar con el usuario con respecto a Facultad en un entorno gráfico), la capa DP o Controller, tiene la extensión .php (esta es la capa de lógica del negocio/desarrollo, es decir, controla los métodos de la clase Facultad que son invocados por los Blade, los cuales envían datos recibidos por el Usuario en las interfaces y que a su vez usan Model para comunicarse con la base de datos y gestionar los registros de la base de datos), y por último, la capa MD o Model, tiene una extensión .php (esta es la capa de manejo de datos que se comunica directamente con la base de datos del sistema y permite realizar instrucciones SQL con respecto a la tabla Facultades).

Cabe recalcar que las dependencias/relaciones son de suma importancia para conocer la función que cumple cada capa con respecto a otra, o desde otro punto de vista como interactúan entre ellas. Por ejemplo: el Blade llamado Facultad Crear.Blade.php es <<redireccionado>> y desplegado al Usuario a través de una llamada desde la interface Facultad.Blade.php al presionar el botón nuevo registro. Aquí, el Usuario interactúa con la interface ingresando por teclado y seleccionando los datos que sirven para agregar una nueva Facultad en base a lo solicitado por las etiquetas (tags de la página web) y por detrás se <<envía>> estos datos; los cuales posteriormente serán manejados en el Controlador FacultadController.php con su función Store que con ayuda del modelo que <<usado>> por el controlador permita ingresar un registro de Facultad en la Base de Datos del Sistema tabla Facultades.

Facultad por Sede

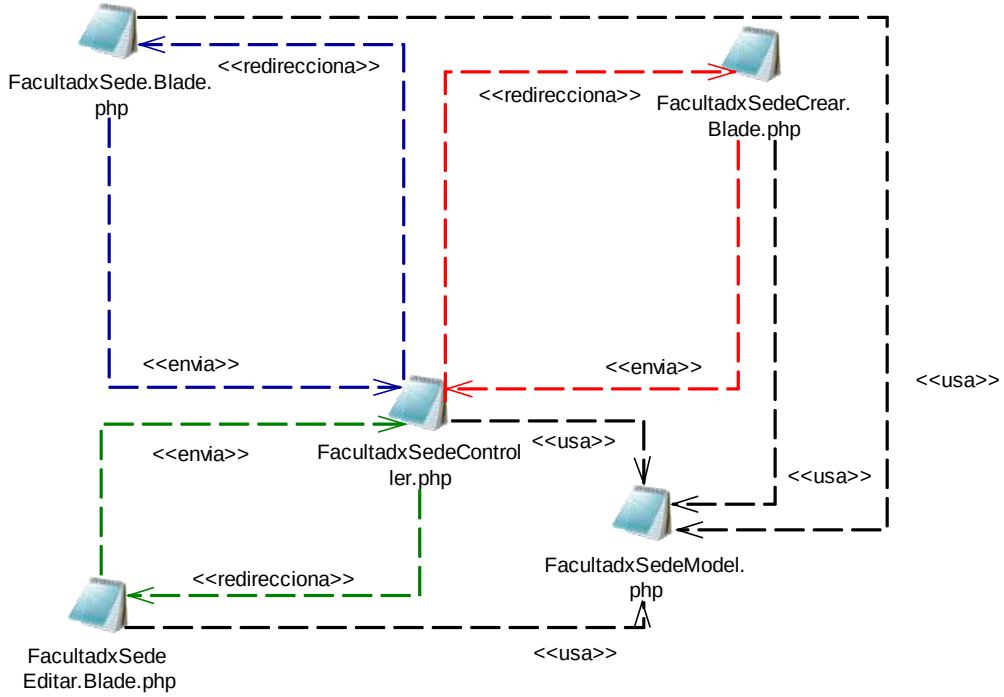


Figura 54. Diagrama de Paquetes por Dependencia FacultadxSede Detallado

Como se puede observar en la figura anterior, el módulo FacultadxSede tiene una estructura definida de comunicación entre capas. Aquí se muestran tres tipos de capas que interactúan entre sí: la capa GUI que tiene una extensión .Blade.php (esta es la capa de visualización que presenta y gestiona datos para interactuar con el usuario con respecto a FacultadxSede en un entorno gráfico), la capa DP o Controller, tiene la extensión .php (esta es la capa de lógica del negocio/desarrollo, es decir, controla los métodos de la clase FacultadxSede que son invocados por los Blade, los cuales envían datos recibidos por el Usuario en las interfaces y que a su vez usan Model para comunicarse con la base de datos y gestionar los registros de la base de datos), y por último, la capa MD o Model, tiene una extensión .php (esta es la capa de manejo de datos que se comunica directamente con la base de datos del sistema y permite realizar instrucciones SQL con respecto a la tabla FacultadesxSedes).

Cabe recalcar que las dependencias/relaciones son de suma importancia para conocer la función que cumple cada capa con respecto a otra, o desde otro punto de vista como interactúan entre ellas. Por ejemplo: el Blade llamado FacultadxSede Crear.Blade.php es <<redireccionado>> y desplegado al Usuario a través de una llamada desde la interface FacultadxSede.Blade.php al presionar el botón nuevo registro. Aquí, el Usuario interactúa con la interface ingresando por teclado y seleccionando los datos que sirven para agregar una nueva FacultadxSede en base a lo solicitado por las etiquetas (tags de la página web) y por detrás se <<envía>> estos datos; los cuales posteriormente serán manejados en el Controlador FacultadxSedeController.php con su función Store que con ayuda del modelo que <<usado>> por el controlador permita ingresar un registro de FacultadxSede en la Base de Datos del Sistema tabla FacultadesxSedes.

Glosario

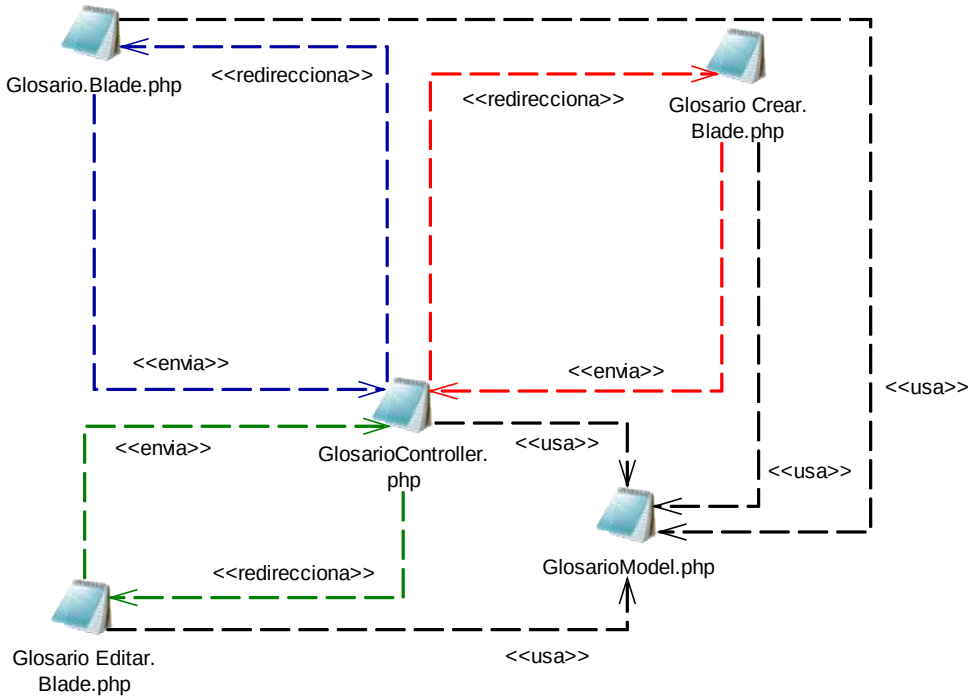


Figura 55. Diagrama de Paquetes por Dependencia Glosario Detallado

Como se puede observar en la figura anterior, el módulo Glosario tiene una estructura definida de comunicación entre capas. Aquí se muestran tres tipos de capas que interactúan entre sí: la capa GUI que tiene una extensión .Blade.php (esta es la capa de visualización que presenta y gestiona datos para interactuar con el usuario con respecto a Glosario en un entorno gráfico), la capa DP o Controller, tiene la extensión .php (esta es la capa de lógica del negocio/desarrollo, es decir, controla los métodos de la clase Glosario que son invocados por los Blade, los cuales envían datos recibidos por el Usuario en las interfaces y que a su vez usan Model para comunicarse con la base de datos y gestionar los registros de la base de datos), y por último, la capa MD o Model, tiene una extensión .php (esta es la capa de manejo de datos que se comunica directamente con la base de datos del sistema y permite realizar instrucciones SQL con respecto a la tabla Glosarios).

Cabe recalcar que las dependencias/relaciones son de suma importancia para conocer la función que cumple cada capa con respecto a otra, o desde otro punto de vista como interactúan entre ellas. Por ejemplo: el Blade llamado Glosario Crear.Blade.php es <<redireccionado>> y desplegado al Usuario a través de una llamada desde la interface Glosario.Blade.php al presionar el botón nuevo registro. Aquí, el Usuario interactúa con la interface ingresando por teclado y seleccionando los datos que sirven para agregar un nuevo Glosario en base a lo solicitado por las etiquetas (tags de la página web) y por detrás se <<envía>> estos datos; los cuales posteriormente serán manejados en el Controlador GlosarioController.php con su función Store que con ayuda del modelo que <<usado>> por el controlador permita ingresar un registro de Glosario en la Base de Datos del Sistema tabla Glosarios.

Periodo

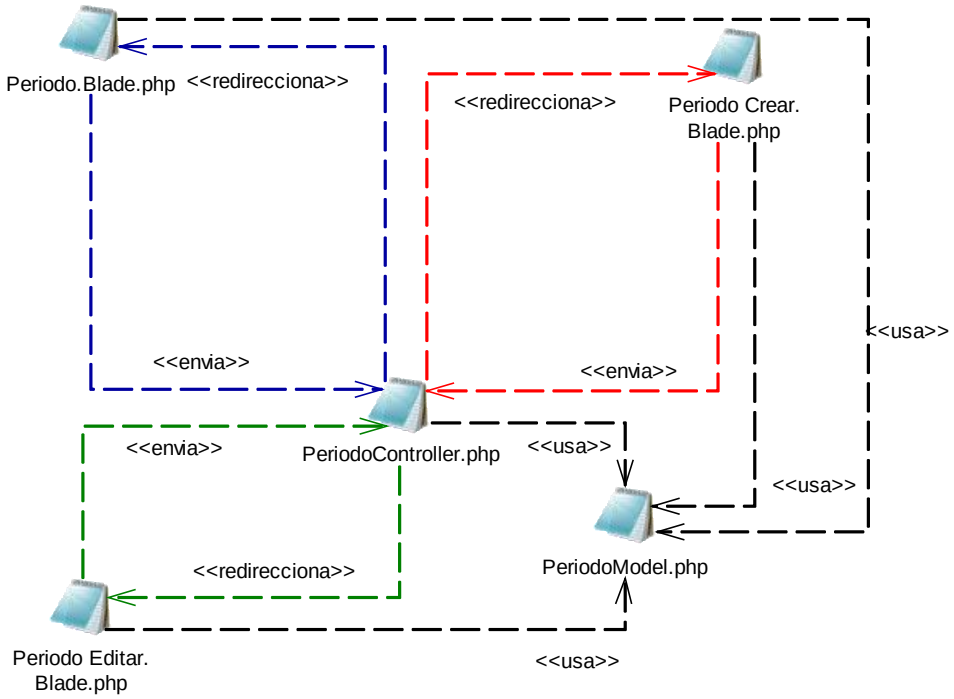


Figura 56. Diagrama de Paquetes por Dependencia Periodo Detallado

Como se puede observar en la figura anterior, el módulo Periodo tiene una estructura definida de comunicación entre capas. Aquí se muestran tres tipos de capas que interactúan entre sí: la capa GUI que tiene una extensión .Blade.php (esta es la capa de visualización que presenta y gestiona datos para interactuar con el usuario con respecto a Periodo en un entorno gráfico), la capa DP o Controller, tiene la extensión .php (esta es la capa de lógica del negocio/desarrollo, es decir, controla los métodos de la clase Periodo que son invocados por los Blade, los cuales envían datos recibidos por el Usuario en las interfaces y que a su vez usan Model para comunicarse con la base de datos y gestionar los registros de la base de datos), y por último, la capa MD o Model, tiene una extensión .php (esta es la capa de manejo de datos que se comunica directamente con la base de datos del sistema y permite realizar instrucciones SQL con respecto a la tabla Periodos).

Cabe recalcar que las dependencias/relaciones son de suma importancia para conocer la función que cumple cada capa con respecto a otra, o desde otro punto de vista como interactúan entre ellas. Por ejemplo: el Blade llamado Periodo Crear.Blade.php es <<redireccionado>> y desplegado al Usuario a través de una llamada desde la interface Periodo.Blade.php al presionar el botón nuevo registro. Aquí, el Usuario interactúa con la interface ingresando por teclado y seleccionando los datos que sirven para agregar un nuevo Periodo en base a lo solicitado por las etiquetas (tags de la página web) y por detrás se <<envía>> estos datos; los cuales posteriormente serán manejados en el Controlador PeriodoController.php con su función Store que con ayuda del modelo que <<usado>> por el controlador permita ingresar un registro de Periodo en la Base de Datos del Sistema tabla Periodos.

Profesor

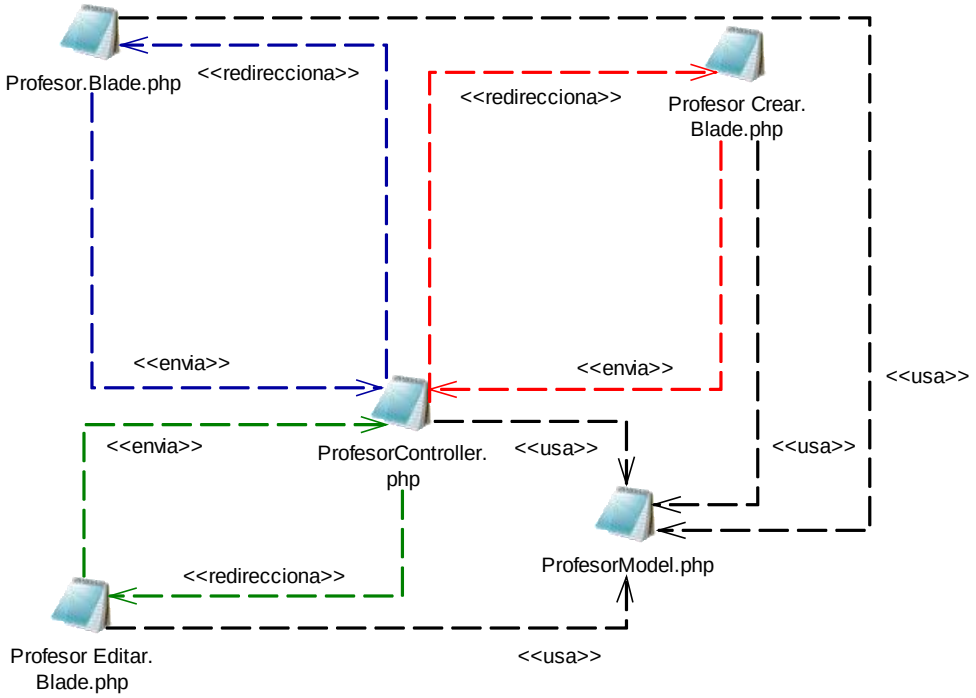


Figura 57. Diagrama de Paquetes por Dependencia Profesor Detallado

Como se puede observar en la figura anterior, el módulo Profesor tiene una estructura definida de comunicación entre capas. Aquí se muestran tres tipos de capas que interactúan entre sí: la capa GUI que tiene una extensión .Blade.php (esta es la capa de visualización que presenta y gestiona datos para interactuar con el usuario con respecto a Profesor en un entorno gráfico), la capa DP o Controller, tiene la extensión .php (esta es la capa de lógica del negocio/desarrollo, es decir, controla los métodos de la clase Profesor que son invocados por los Blade, los cuales envían datos recibidos por el Usuario en las interfaces y que a su vez usan Model para comunicarse con la base de datos y gestionar los registros de la base de datos), y por último, la capa MD o Model, tiene una extensión .php (esta es la capa de manejo de datos que se comunica directamente con la base de datos del sistema y permite realizar instrucciones SQL con respecto a la tabla Profesores).

Cabe recalcar que las dependencias/relaciones son de suma importancia para conocer la función que cumple cada capa con respecto a otra, o desde otro punto de vista como interactúan entre ellas. Por ejemplo: el Blade llamado Profesor Crear.Blade.php es <<redireccionado>> y desplegado al Usuario a través de una llamada desde la interface Profesor.Blade.php al presionar el botón nuevo registro. Aquí, el Usuario interactúa con la interface ingresando por teclado y seleccionando los datos que sirven para agregar un nuevo Profesor en base a lo solicitado por las etiquetas (tags de la página web) y por detrás se <<envía>> estos datos; los cuales posteriormente serán manejados en el Controlador ProfesorController.php con su función Store que con ayuda del modelo que <<usado>> por el controlador permita ingresar un registro de Profesor en la Base de Datos del Sistema tabla Profesores.

Provincia

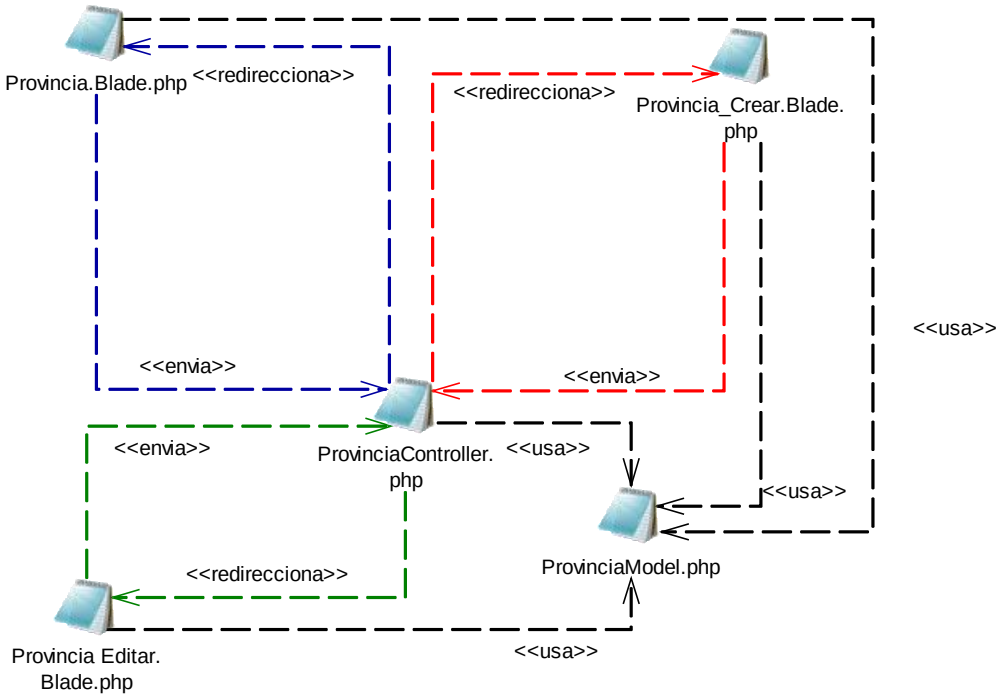


Figura 58. Diagrama de Paquetes por Dependencia Provincia Detallado

Como se puede observar en la figura anterior, el módulo Provincia tiene una estructura definida de comunicación entre capas. Aquí se muestran tres tipos de capas que interactúan entre sí: la capa GUI que tiene una extensión .Blade.php (esta es la capa de visualización que presenta y gestiona datos para interactuar con el usuario con respecto a Provincia en un entorno gráfico), la capa DP o Controller, tiene la extensión .php (esta es la capa de lógica del negocio/desarrollo, es decir, controla los métodos de la clase Provincia que son invocados por los Blade, los cuales envían datos recibidos por el Usuario en las interfaces y que a su vez usan Model para comunicarse con la base de datos y gestionar los registros de la base de datos), y por último, la capa MD o Model, tiene una extensión .php (esta es la capa de manejo de datos que se comunica directamente con la base de datos del sistema y permite realizar instrucciones SQL con respecto a la tabla Provincias).

Cabe recalcar que las dependencias/relaciones son de suma importancia para conocer la función que cumple cada capa con respecto a otra, o desde otro punto de vista como interactúan entre ellas. Por ejemplo: el Blade llamado Provincia Crear.Blade.php es <<redireccionado>> y desplegado al Usuario a través de una llamada desde la interface Provincia.Blade.php al presionar el botón nuevo registro. Aquí, el Usuario interactúa con la interface ingresando por teclado y seleccionando los datos que sirven para agregar una nueva Provincia en base a lo solicitado por las etiquetas (tags de la página web) y por detrás se <<envía>> estos datos; los cuales posteriormente serán manejados en el Controlador ProvinciaController.php con su función Store que con ayuda del modelo que <<usado>> por el controlador permita ingresar un registro de Provincia en la Base de Datos del Sistema tabla Provincias.

Sede

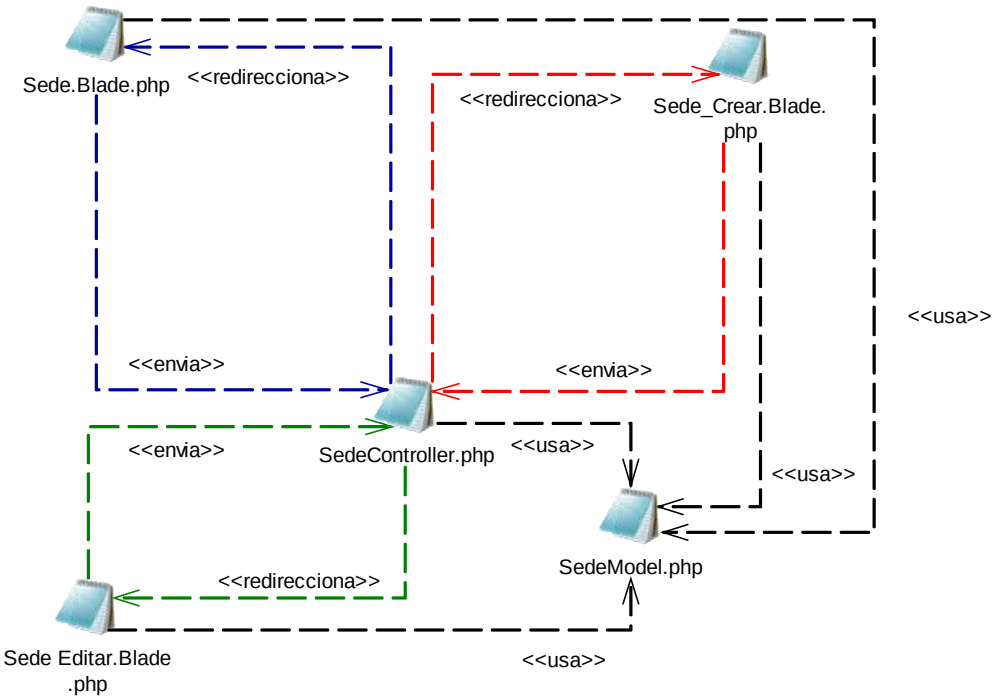


Figura 59. Diagrama de Paquetes por Dependencia Sede Detallado

Como se puede observar en la figura anterior, el módulo Sede tiene una estructura definida de comunicación entre capas. Aquí se muestran tres tipos de capas que interactúan entre sí: la capa GUI que tiene una extensión .Blade.php (esta es la capa de visualización que presenta y gestiona datos para interactuar con el usuario con respecto a Sede en un entorno gráfico), la capa DP o Controller, tiene la extensión .php (esta es la capa de lógica del negocio/desarrollo, es decir, controla los métodos de la clase Sede que son invocados por los Blade, los cuales envían datos recibidos por el Usuario en las interfaces y que a su vez usan Model para comunicarse con la base de datos y gestionar los registros de la base de datos), y por último, la capa MD o Model, tiene una extensión .php (esta es la capa de manejo de datos que se comunica directamente con la base de datos del sistema y permite realizar instrucciones SQL con respecto a la tabla Sedes).

Cabe recalcar que las dependencias/relaciones son de suma importancia para conocer la función que cumple cada capa con respecto a otra, o desde otro punto de vista como interactúan entre ellas. Por ejemplo: el Blade llamado Sede Crear.Blade.php es <<redireccionado>> y desplegado al Usuario a través de una llamada desde la interface Sede.Blade.php al presionar el botón nuevo registro. Aquí, el Usuario interactúa con la interface ingresando por teclado y seleccionando los datos que sirven para agregar una nueva Sede en base a lo solicitado por las etiquetas (tags de la página web) y por detrás se <<envía>> estos datos; los cuales posteriormente serán manejados en el Controlador SedeController.php con su función Store que con ayuda del modelo que <<usado>> por el controlador permita ingresar un registro de Sede en la Base de Datos del Sistema tabla Sedes.

Taller

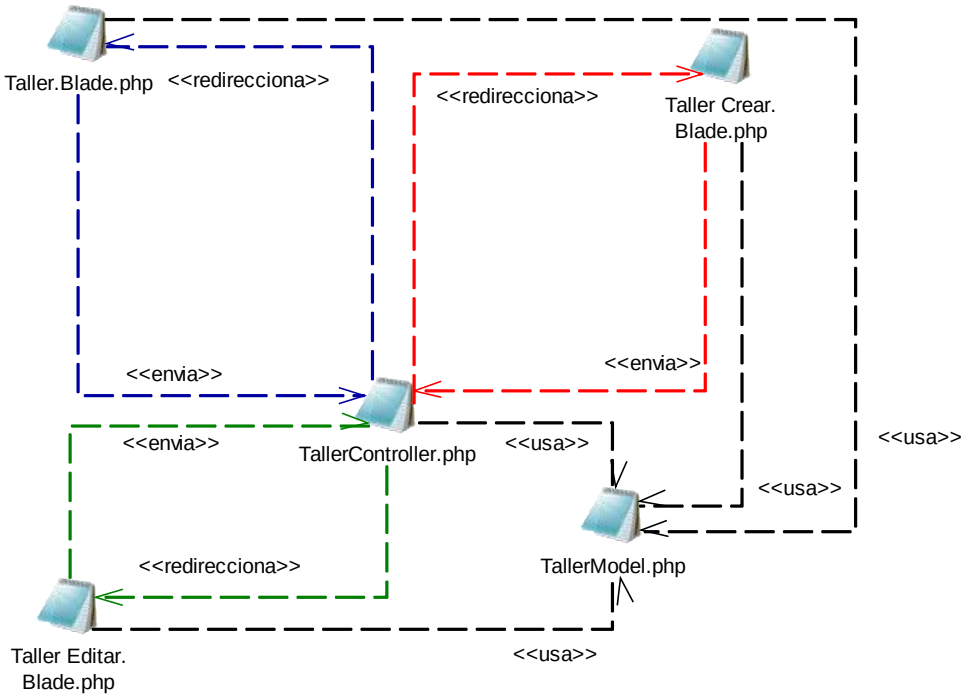


Figura 60. Diagrama de Paquetes por Dependencia Taller Detallado

Como se puede observar en la figura anterior, el módulo Taller tiene una estructura definida de comunicación entre capas. Aquí se muestran tres tipos de capas que interactúan entre sí: la capa GUI que tiene una extensión .Blade.php (esta es la capa de visualización que presenta y gestiona datos para interactuar con el usuario con respecto a Taller en un entorno gráfico), la capa DP o Controller, tiene la extensión .php (esta es la capa de lógica del negocio/desarrollo, es decir, controla los métodos de la clase Taller que son invocados por los Blade, los cuales envían datos recibidos por el Usuario en las interfaces y que a su vez usan Model para comunicarse con la base de datos y gestionar los registros de la base de datos), y por último, la capa MD o Model, tiene una extensión .php (esta es la capa de manejo de datos que se comunica directamente con la base de datos del sistema y permite realizar instrucciones SQL con respecto a la tabla Talleres).

Cabe recalcar que las dependencias/relaciones son de suma importancia para conocer la función que cumple cada capa con respecto a otra, o desde otro punto de vista como interactúan entre ellas. Por ejemplo: el Blade llamado Taller Crear.Blade.php es <<redireccionado>> y desplegado al Usuario a través de una llamada desde la interface Taller.Blade.php al presionar el botón nuevo registro. Aquí, el Usuario interactúa con la interface ingresando por teclado y seleccionando los datos que sirven para agregar un nuevo Taller en base a lo solicitado por las etiquetas (tags de la página web) y por detrás se <<envía>> estos datos; los cuales posteriormente serán manejados en el Controlador TallerController.php con su función Store que con ayuda del modelo que <<usado>> por el controlador permita ingresar un registro de Taller en la Base de Datos del Sistema tabla Talleres.

Tema

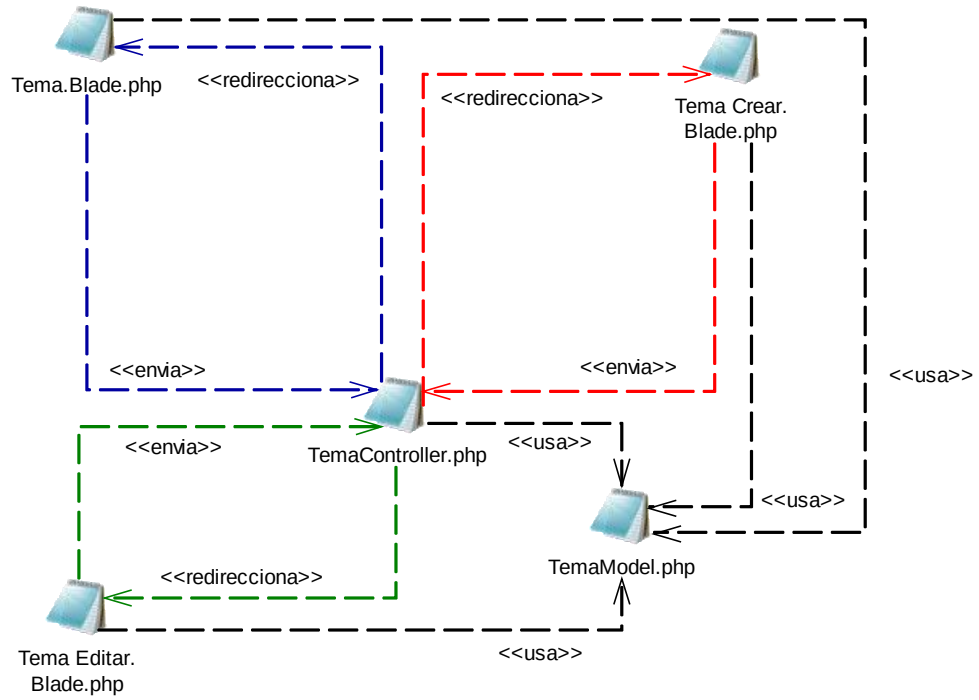


Figura 61. Diagrama de Paquetes por Dependencia Tema Detallado

Como se puede observar en la figura anterior, el módulo Tema tiene una estructura definida de comunicación entre capas. Aquí se muestran tres tipos de capas que interactúan entre sí: la capa GUI que tiene una extensión .Blade.php (esta es la capa de visualización que presenta y gestiona datos para interactuar con el usuario con respecto a Tema en un entorno gráfico), la capa DP o Controller, tiene la extensión .php (esta es la capa de lógica del negocio/desarrollo, es decir, controla los métodos de la clase Tema que son invocados por los Blade, los cuales envían datos recibidos por el Usuario en las interfaces y que a su vez usan Model para comunicarse con la base de datos y gestionar los registros de la base de datos), y por último, la capa MD o Model, tiene una extensión .php (esta es la capa de manejo de datos que se comunica directamente con la base de datos del sistema y permite realizar instrucciones SQL con respecto a la tabla Temas).

Cabe recalcar que las dependencias/relaciones son de suma importancia para conocer la función que cumple cada capa con respecto a otra, o desde otro punto de vista como interactúan entre ellas. Por ejemplo: el Blade llamado Tema Crear.Blade.php es <<redireccionado>> y desplegado al Usuario a través de una llamada desde la interface Tema.Blade.php al presionar el botón nuevo registro. Aquí, el Usuario interactúa con la interface ingresando por teclado y seleccionando los datos que sirven para agregar un nuevo Tema en base a lo solicitado por las etiquetas (tags de la página web) y por detrás se <<envía>> estos datos; los cuales posteriormente serán manejados en el Controlador TemaController.php con su función Store que con ayuda del modelo que <<usado>> por el controlador permita ingresar un registro de Tema en la Base de Datos del Sistema tabla Temas.

Unidad

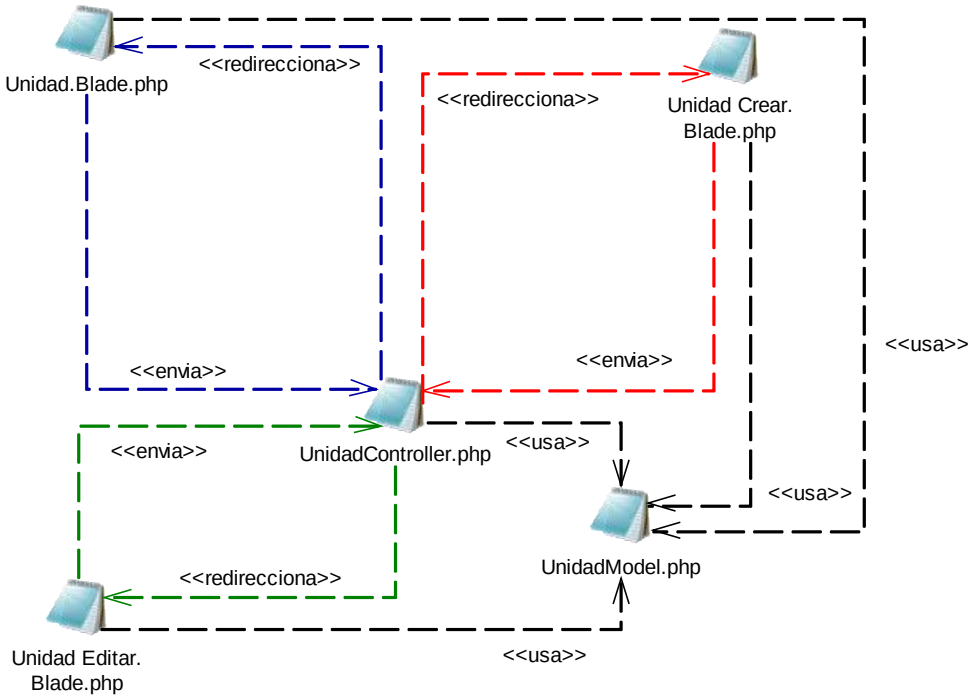


Figura 62. Diagrama de Paquetes por Dependencia Unidad Detallado

Como se puede observar en la figura anterior, el módulo Unidad tiene una estructura definida de comunicación entre capas. Aquí se muestran tres tipos de capas que interactúan entre sí: la capa GUI que tiene una extensión .Blade.php (esta es la capa de visualización que presenta y gestiona datos para interactuar con el usuario con respecto a Unidad en un entorno gráfico), la capa DP o Controller, tiene la extensión .php (esta es la capa de lógica del negocio/desarrollo, es decir, controla los métodos de la clase Unidad que son invocados por los Blade, los cuales envían datos recibidos por el Usuario en las interfaces y que a su vez usan Model para comunicarse con la base de datos y gestionar los registros de la base de datos), y por último, la capa MD o Model, tiene una extensión .php (esta es la capa de manejo de datos que se comunica directamente con la base de datos del sistema y permite realizar instrucciones SQL con respecto a la tabla Unidades).

Cabe recalcar que las dependencias/relaciones son de suma importancia para conocer la función que cumple cada capa con respecto a otra, o desde otro punto de vista como interactúan entre ellas. Por ejemplo: el Blade llamado Unidad Crear.Blade.php es <<redireccionado>> y desplegado al Usuario a través de una llamada desde la interface Unidad.Blade.php al presionar el botón nuevo registro. Aquí, el Usuario interactúa con la interface ingresando por teclado y seleccionando los datos que sirven para agregar una nueva Unidad en base a lo solicitado por las etiquetas (tags de la página web) y por detrás se <<envía>> estos datos; los cuales posteriormente serán manejados en el Controlador UnidadController.php con su función Store que con ayuda del modelo que <<usado>> por el controlador permita ingresar un registro de Unidad en la Base de Datos del Sistema tabla Unidades.

Universidad

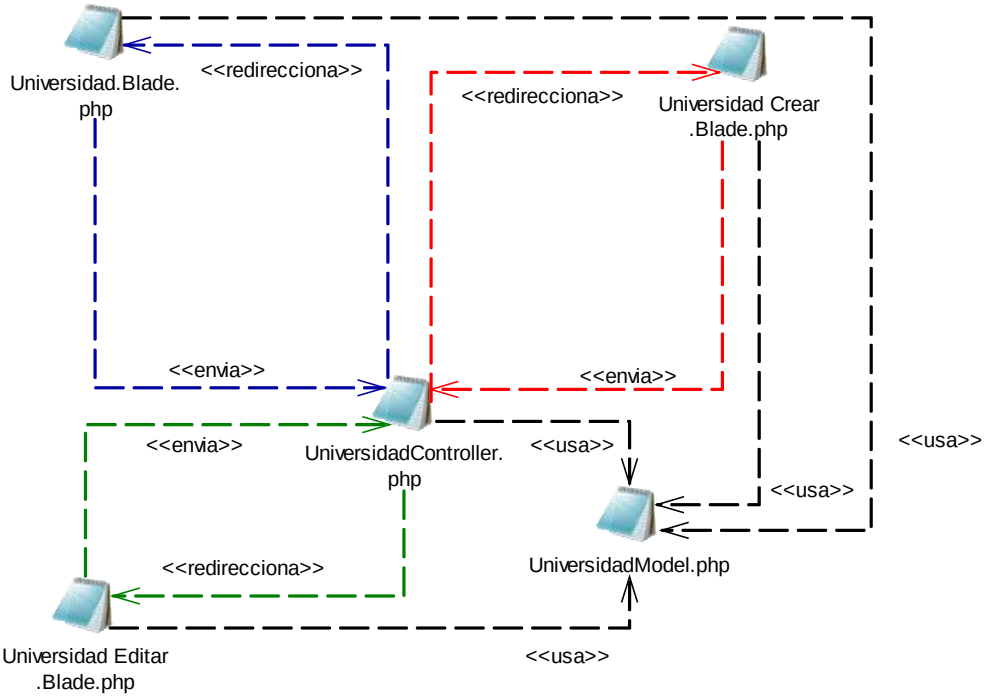


Figura 63. Diagrama de Paquetes por Dependencia Universidad Detallado

Como se puede observar en la figura anterior, el módulo Universidad tiene una estructura definida de comunicación entre capas. Aquí se muestran tres tipos de capas que interactúan entre sí: la capa GUI que tiene una extensión .Blade.php (esta es la capa de visualización que presenta y gestiona datos para interactuar con el usuario con respecto a Universidad en un entorno gráfico), la capa DP o Controller, tiene la extensión .php (esta es la capa de lógica del negocio/desarrollo, es decir, controla los métodos de la clase Universidad que son invocados por los Blade, los cuales envían datos recibidos por el Usuario en las interfaces y que a su vez usan Model para comunicarse con la base de datos y gestionar los registros de la base de datos), y por último, la capa MD o Model, tiene una extensión .php (esta es la capa de manejo de datos que se comunica directamente con la base de datos del sistema y permite realizar instrucciones SQL con respecto a la tabla Universidades).

Cabe recalcar que las dependencias/relaciones son de suma importancia para conocer la función que cumple cada capa con respecto a otra, o desde otro punto de vista como interactúan entre ellas. Por ejemplo: el Blade llamado Universidad Crear.Blade.php es <<redireccionado>> y desplegado al Usuario a través de una llamada desde la interface Universidad.Blade.php al presionar el botón nuevo registro. Aquí, el Usuario interactúa con la interface ingresando por teclado y seleccionando los datos que sirven para agregar una nueva Universidad en base a lo solicitado por las etiquetas (tags de la página web) y por detrás se <<envía>> estos datos; los cuales posteriormente serán manejados en el Controlador UniversidadController.php con su función Store que con ayuda del modelo que <<usado>> por el controlador permita ingresar un registro de Universidad en la Base de Datos del Sistema tabla Universidades.

Diagrama de Paquetes por Asociación

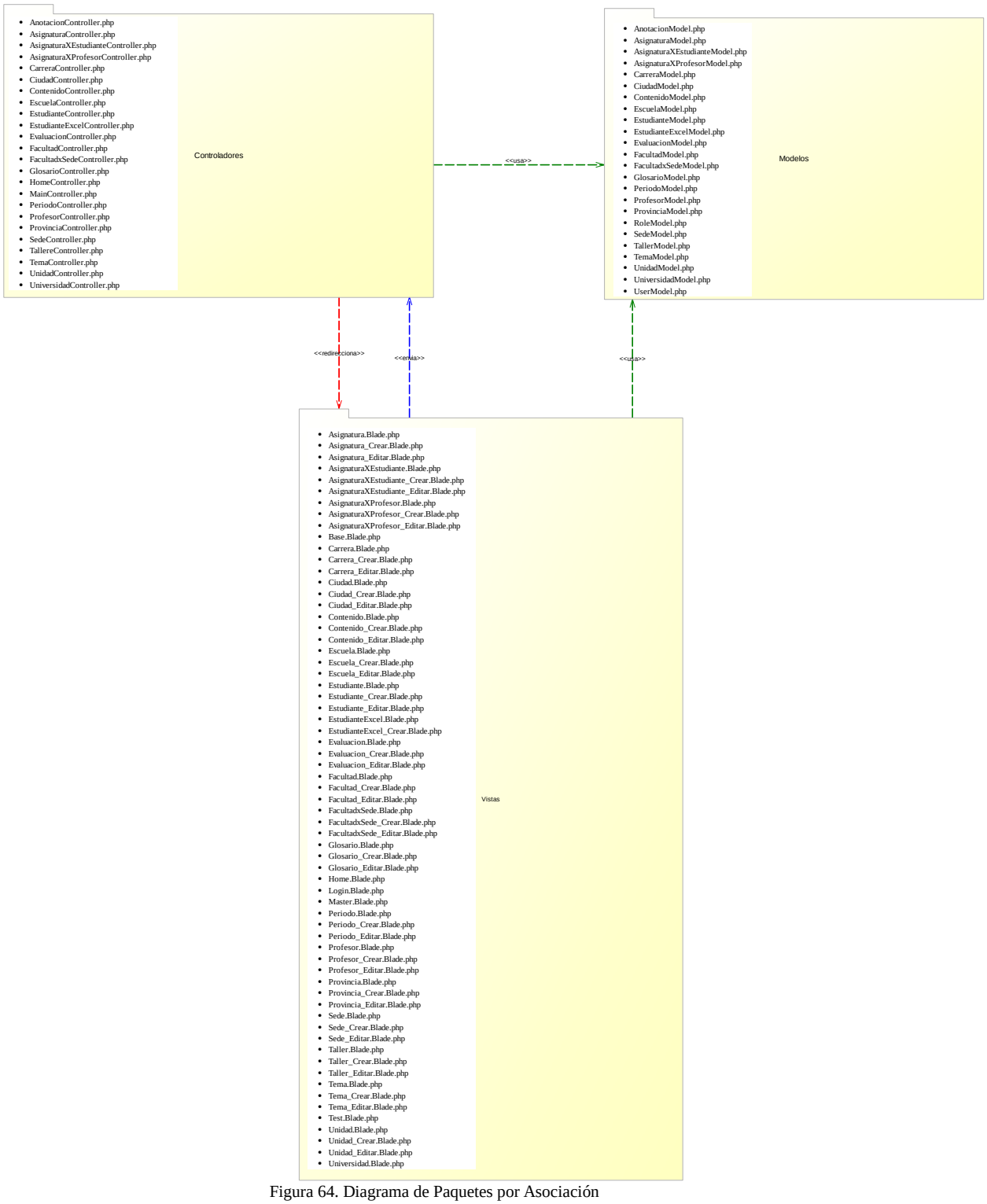


Figura 64. Diagrama de Paquetes por Asociación

Modelo Entidad-Relación

El modelo de Entidad-Relación en el Sistema de Agenda Electrónica del Estudiante tiene como propósito gestionar el almacenamiento de datos dentro de tablas. Para esto, existen 3 tipos de modelos que manejan diversos tipos de niveles de modelización, el modelo conceptual que analiza de forma general la Base de Datos, el modelo físico que analiza las tablas con cardinalidad y el modelo lógico que ya maneja atributos, relaciones y cardinalidad.

Modelo Conceptual

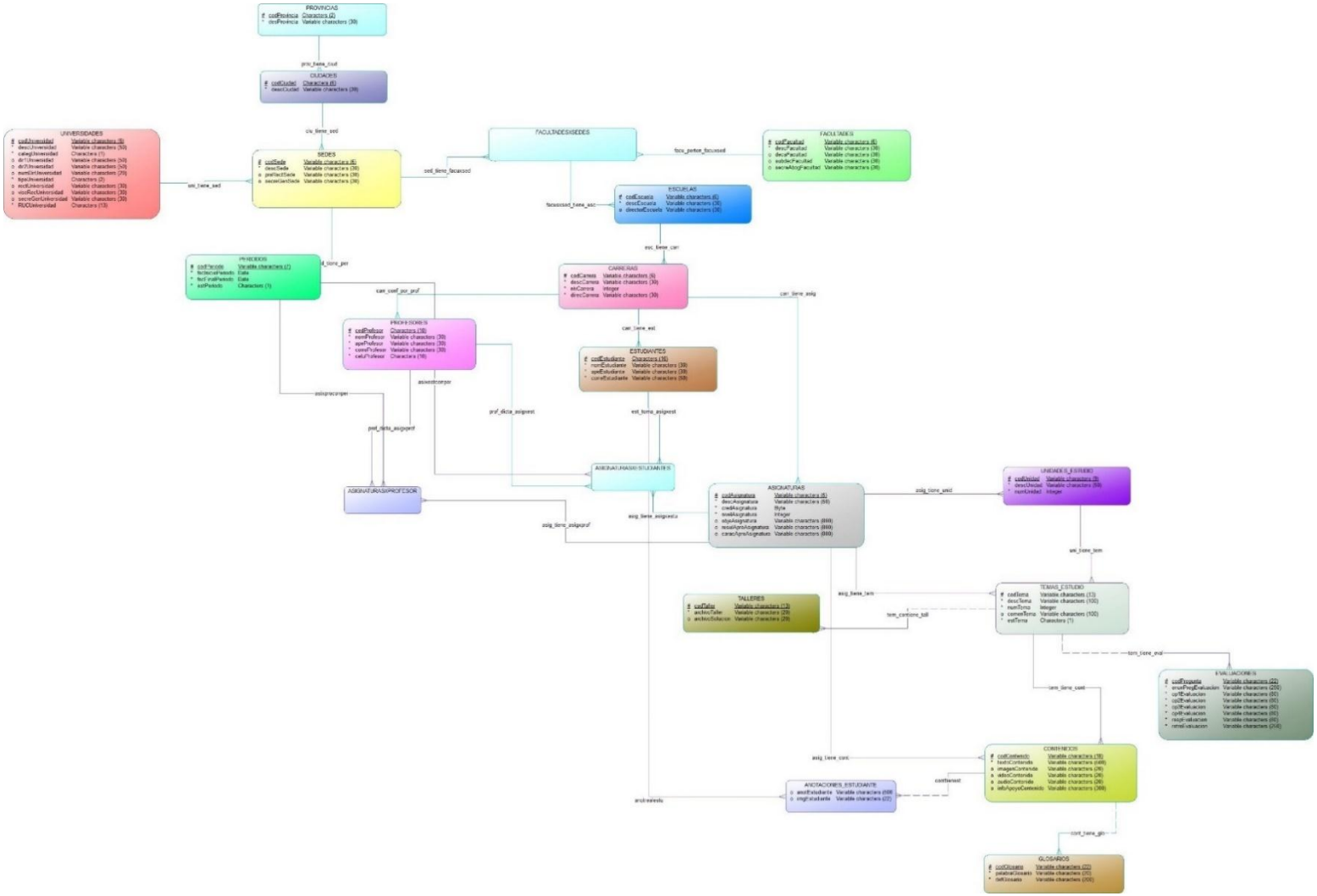


Figura 65. Modelo Conceptual de la Base de Datos del Sistema de Agenda Electrónica para el Estudiante

Según la figura anterior se puede observar el nombre de las relaciones que componen al modelo conceptual fueron detalladas siguiendo el estándar de la hoja de validación. Sin embargo, cabe recalcar que una entidad intermediaria no necesariamente es un proceso como es el caso de FacultadesxSedes, sino que se lo considera como una tabla intermedia de la cual se consumen resultados para agilizar el tiempo de respuesta y simplemente tener que usar un Modelo con instrucciones SQL y no 2; es por esto por lo que se la coloca vacía.

Modelo Físico

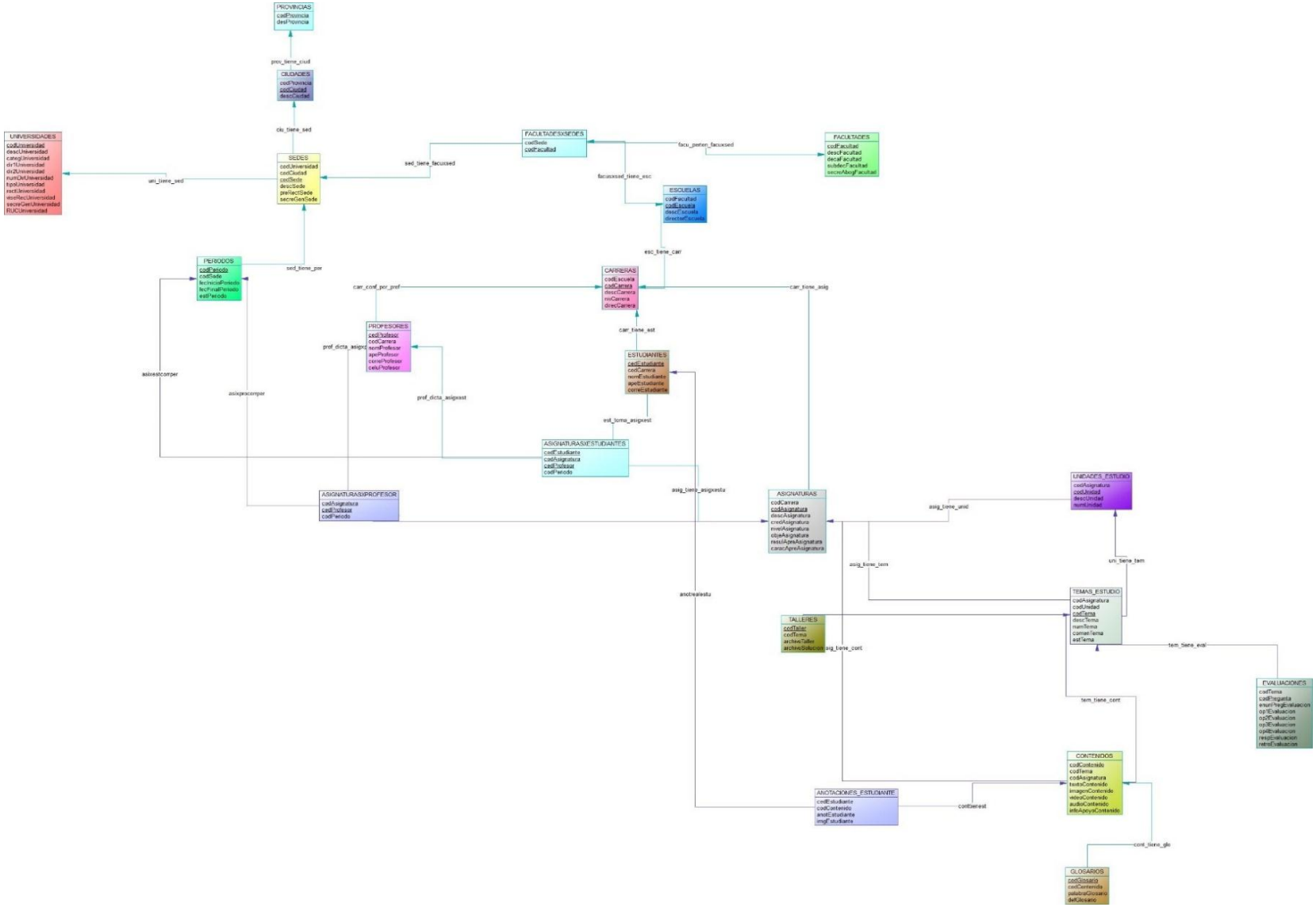


Figura 66. Modelo Físico de la Base de Datos del Sistema de Agenda Electrónica para el Estudiante

Según la figura anterior se puede observar el nombre de las relaciones que componen al modelo conceptual fueron detalladas siguiendo el estándar de la hoja de validación. Sin embargo, cabe recalcar que una entidad intermediaria no necesariamente es un proceso como es el caso de FacultadesxSedes, sino que se lo considera como una tabla intermedia de la cual se consumen resultados para agilizar el tiempo de respuesta y simplemente tener que usar un Modelo con instrucciones SQL y no 2; es por esto por lo que se la coloca vacía. La diferencia entre el modelo conceptual es que aquí ya existe cardinalidad de las flechas de las relaciones, que permite saber qué entidad depende de otra para existir.

Modelo Lógico

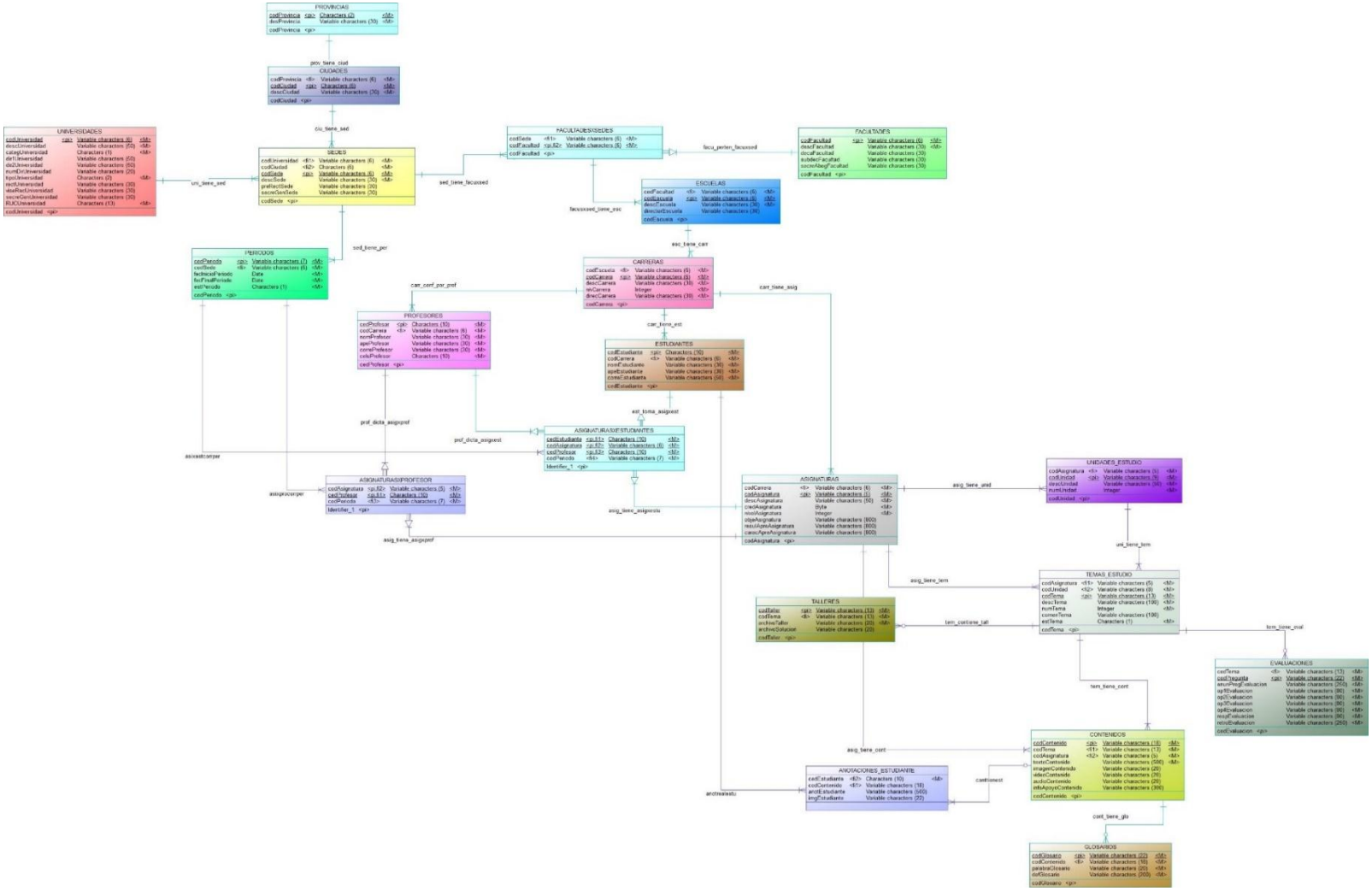


Figura 67. Modelo Lógico de la Base de Datos del Sistema de Agenda Electrónica para el Estudiante

Según la figura anterior se puede observar el nombre de las relaciones que componen al modelo conceptual fueron detalladas siguiendo el estándar de la hoja de validación. Este modelo tiene claramente definidas las tablas intermedia y procesos del Sistema mediante el usa de un Modelo con instrucciones SQL. La diferencia entre el modelo físico con el resto es que, este es el más completo y sirve incluso para generar un script de carga de datos y de carga de la tabla de la base de datos. Por otra parte, aquí ya existe cardinalidad de las flechas de las relaciones, que permite saber qué entidad depende de otra para existir.

Diagramas de Secuencia

F0: Gestionar Inicio de Sesión

F0.1: Iniciar Sesión

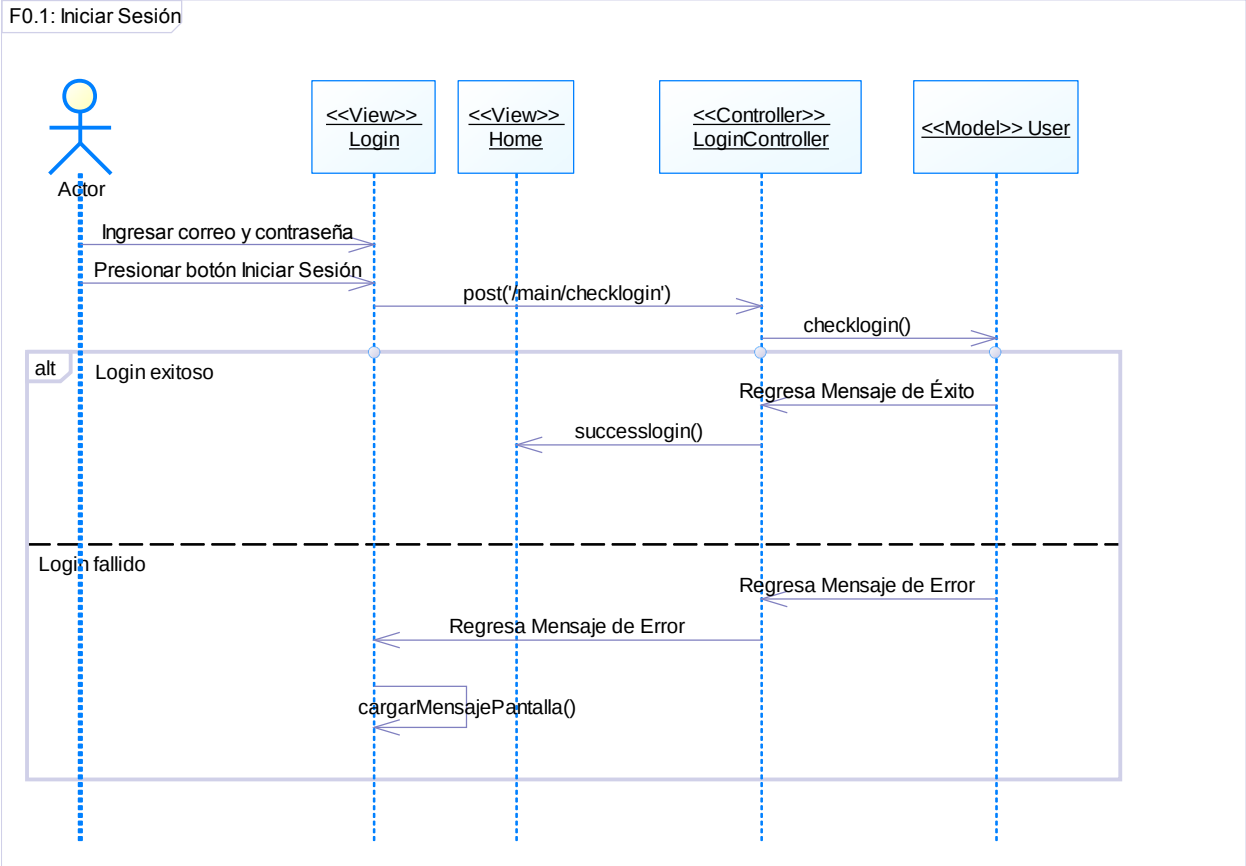


Figura 68. Diagrama Secuencia F0.1: Gestionar Inicio de Sesión

Actores

1. Administrador
2. Profesor
3. Estudiante

Flujo Principal

1. El actor ingresa su correo y contraseña
2. El actor presión Iniciar Sesión

3. El sistema verifica y comprueba que el usuario se encuentre registrado
4. El sistema presenta la ventana Home

Flujo Alterno

3. El sistema verifica y no logra comprobar que el usuario se encuentre registrado
4. El sistema presenta un mensaje de error por pantalla

F0.2: Cambiar Contraseña

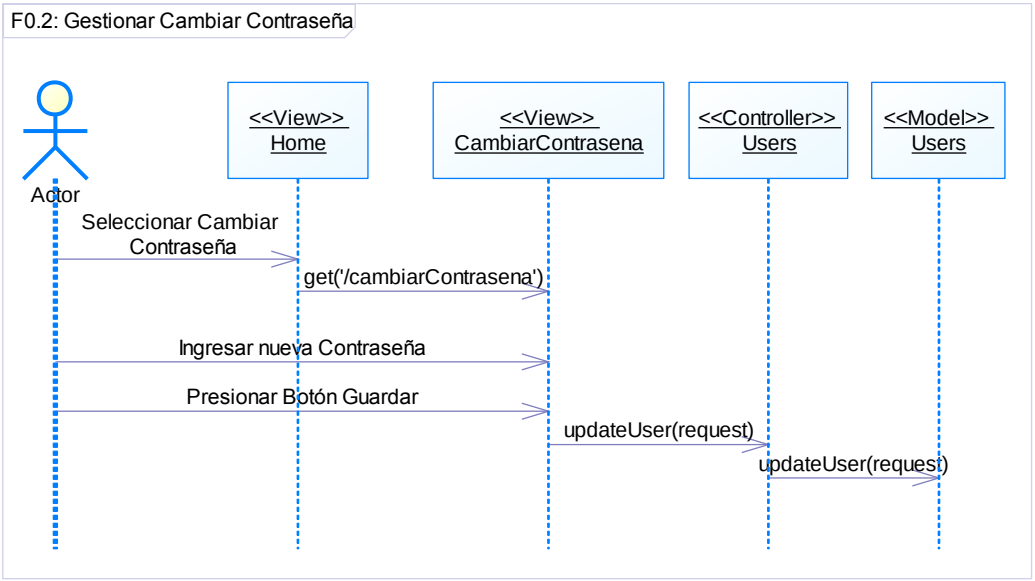


Figura 69. Diagrama Secuencia F0.2: Gestionar Cambiar Contraseña

Actores

1. Administrador
2. Profesor
3. Estudiante

Flujo Principal

1. El actor selecciona la Opción Cambiar Contraseña
2. El sistema presenta la ventana CambiarContrasena
3. El actor ingresa su nueva contraseña y presiona el botón Guardar
4. El sistema modifica y almacena la nueva contraseña

F1: Gestionar Provincia

F1.1: Ingresar Provincia

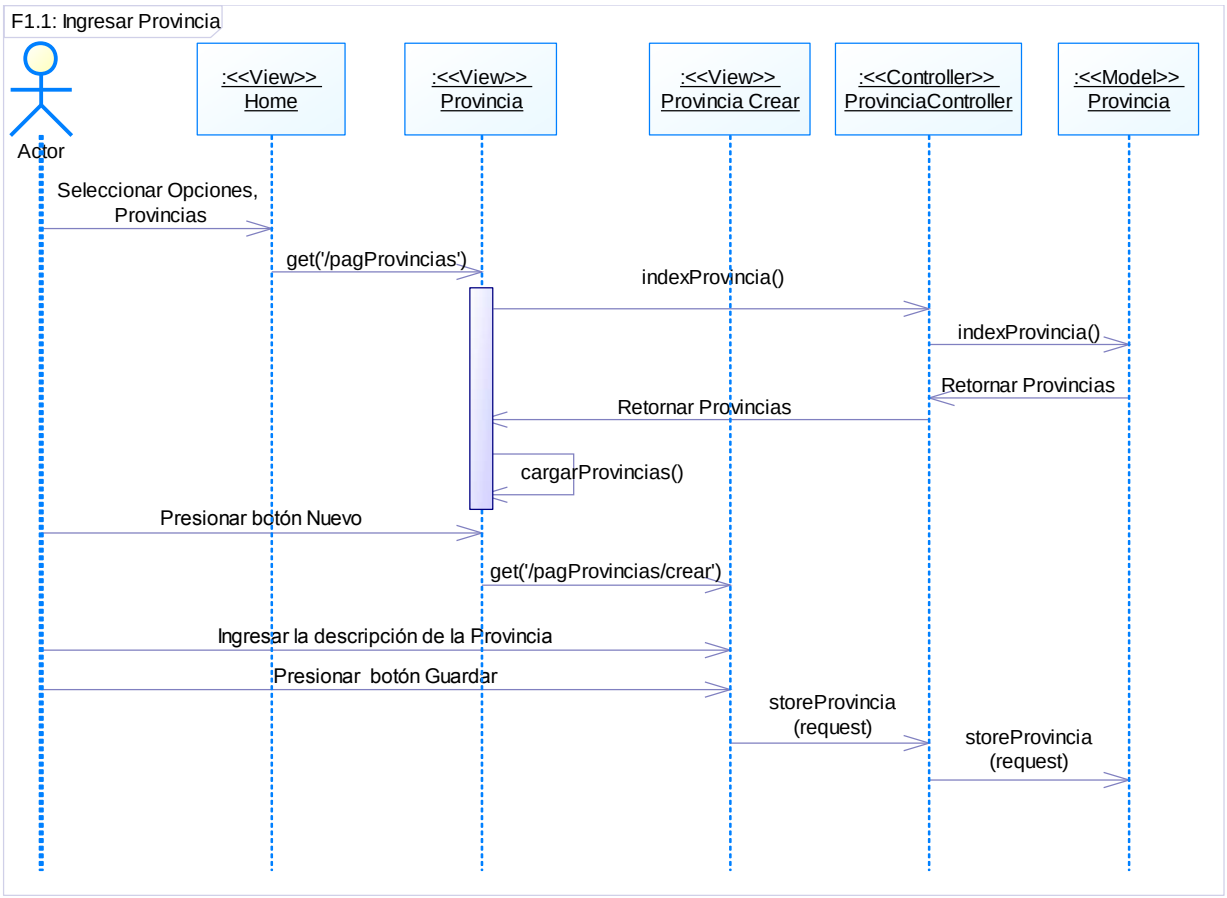


Figura 70. Diagrama Secuencia F1.1: Ingresar Provincia

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Provincias
2. El sistema despliega la ventana de Provincia Crear
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Provincias ingresados previamente
4. El actor presiona el botón de Nuevo para ingresar un nuevo registro

5. El sistema presenta dentro de la ventana el campo Código lleno, y el campo descripción vacío
6. El actor ingresa el campo de la descripción de la Provincia
7. El actor presiona el botón Guardar
8. El sistema almacena los datos

F1.2: Modificar Provincia

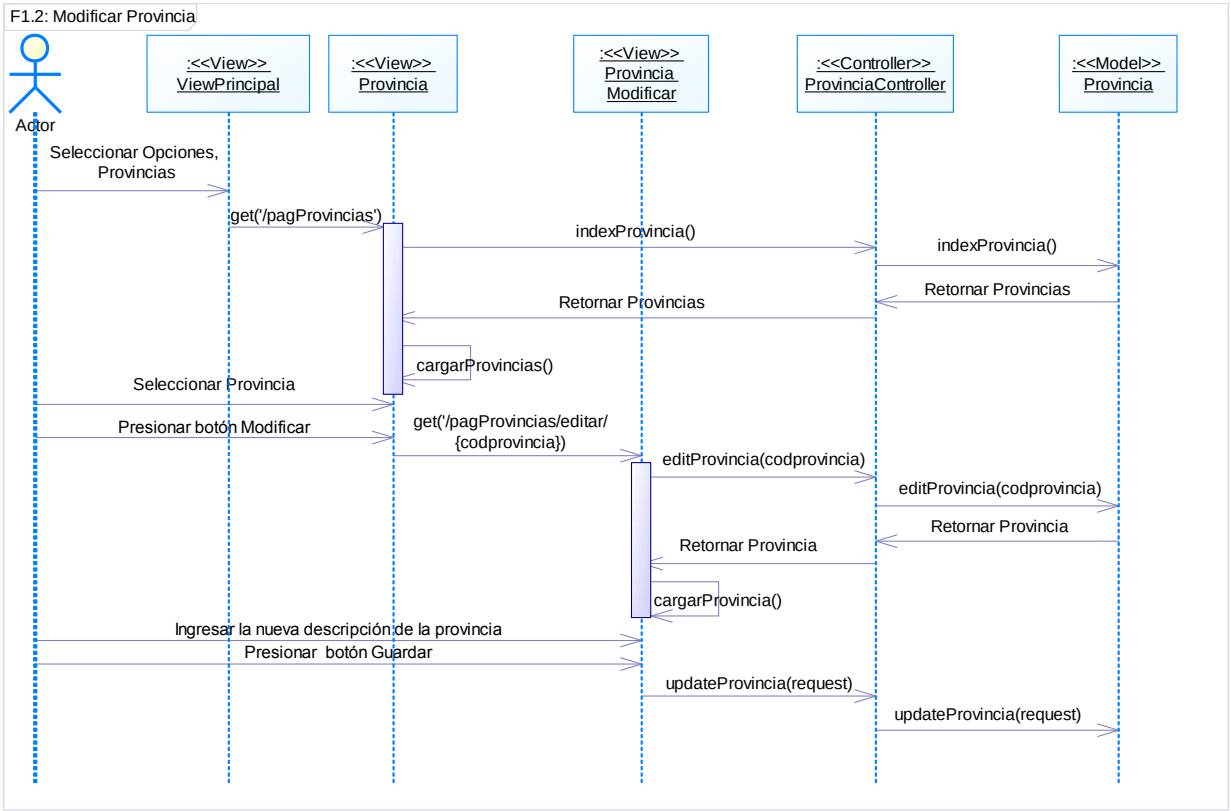


Figura 71. Diagrama Secuencia F1.2: Modificar Provincia

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Provincias
2. El sistema despliega la ventana de Provincia
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Provincias ingresados previamente

4. El actor presiona el botón de lápiz (Modificar) en el registro que desee modificar
5. El sistema presenta la ventana Provincias Editar con todos los campos llenos (código, descripción de Provincia) y bloquea el campo código
6. El actor ingresa nuevamente el campo de la descripción de la Provincia
7. El actor presiona el botón Guardar
8. El sistema modifica los datos

F1.3: Eliminar Provincia

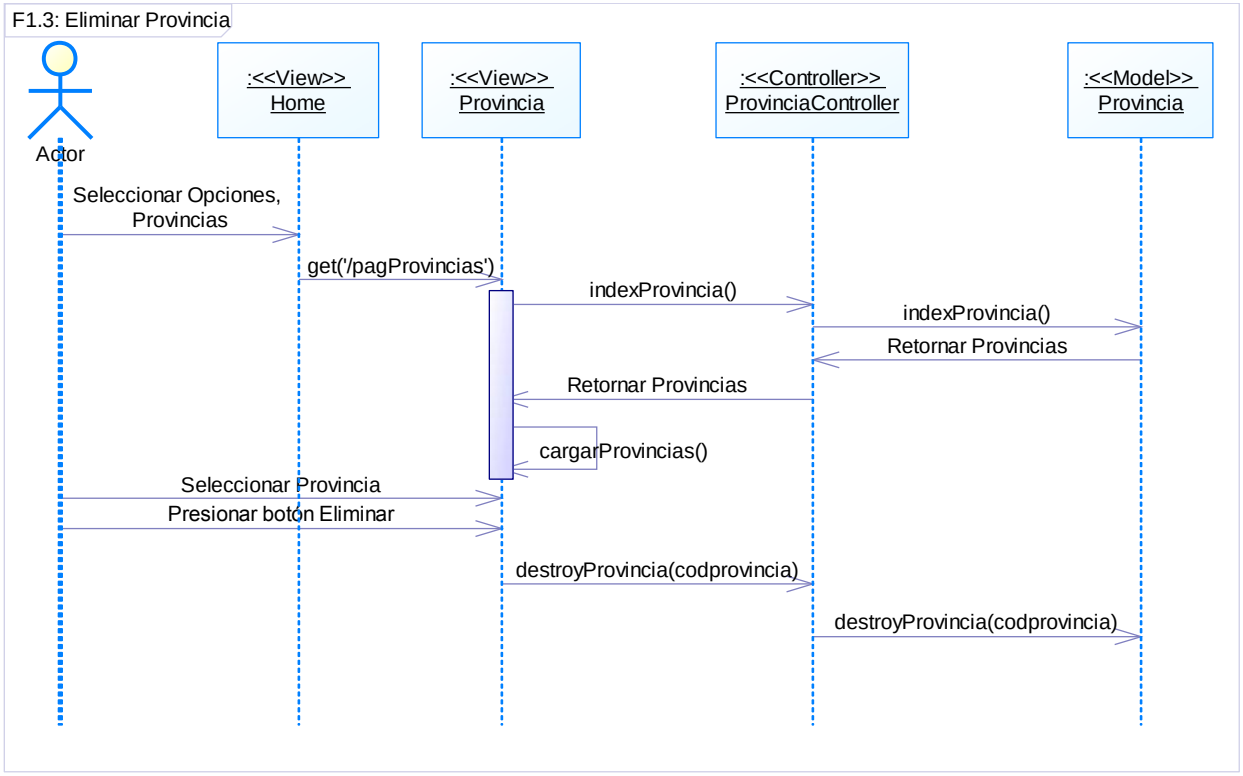


Figura 72. Diagrama Secuencia F1.3: Eliminar Provincia

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Provincias
2. El sistema despliega la ventana de Provincia
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Provincias ingresados previamente
4. El actor presiona el botón de basurero (Eliminar) en el registro que desee eliminar

5. El sistema elimina el registro seleccionado

F1.4: Consultar Provincia General

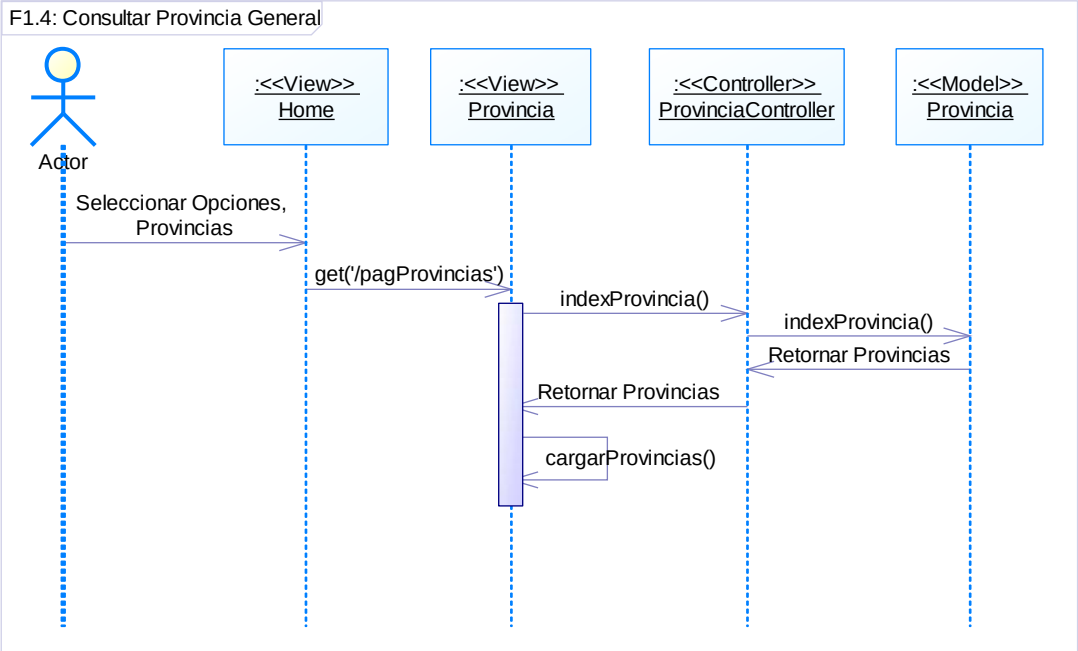


Figura 73. Diagrama Secuencia F1.4: Consultar Provincia General

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Provincias
2. El sistema despliega la ventana de Provincia
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Provincias ingresados previamente

F1.4.1: Consultar Provincia por Parámetros

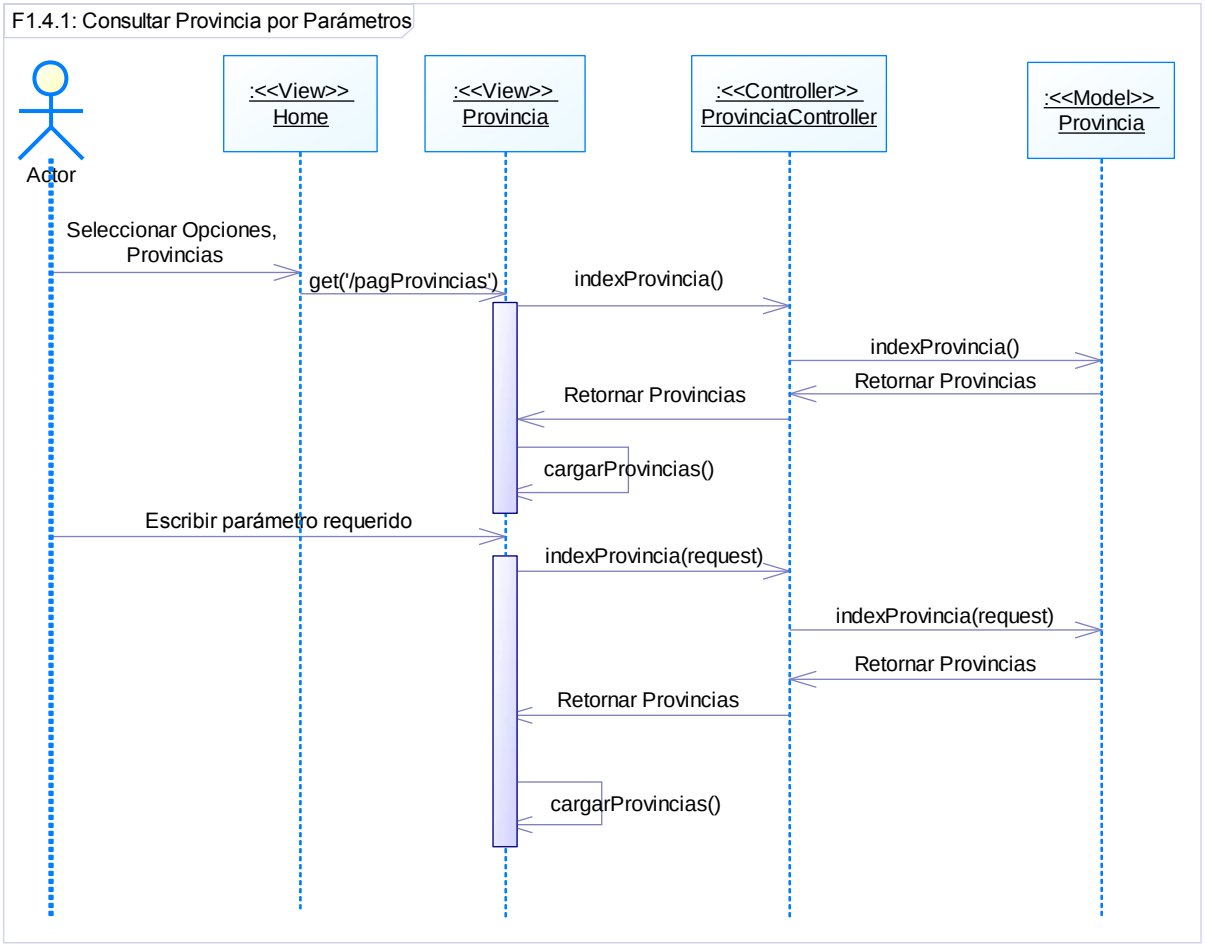


Figura 74. Diagrama Secuencia F1.4.1: Consultar Provincia por Parámetros

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Provincias
2. El sistema despliega la ventana de Provincia

3. El sistema presenta dentro de la ventana una tabla con todos los registros de Provincias ingresados previamente
4. El actor escribe la palabra que desea en la barra de búsqueda
5. El sistema presenta dentro de la tabla del punto 3 todos los registros que cumplen con el parámetro ingresado por el actor

F2: Gestionar Ciudad

F2.1: Ingresar Ciudad

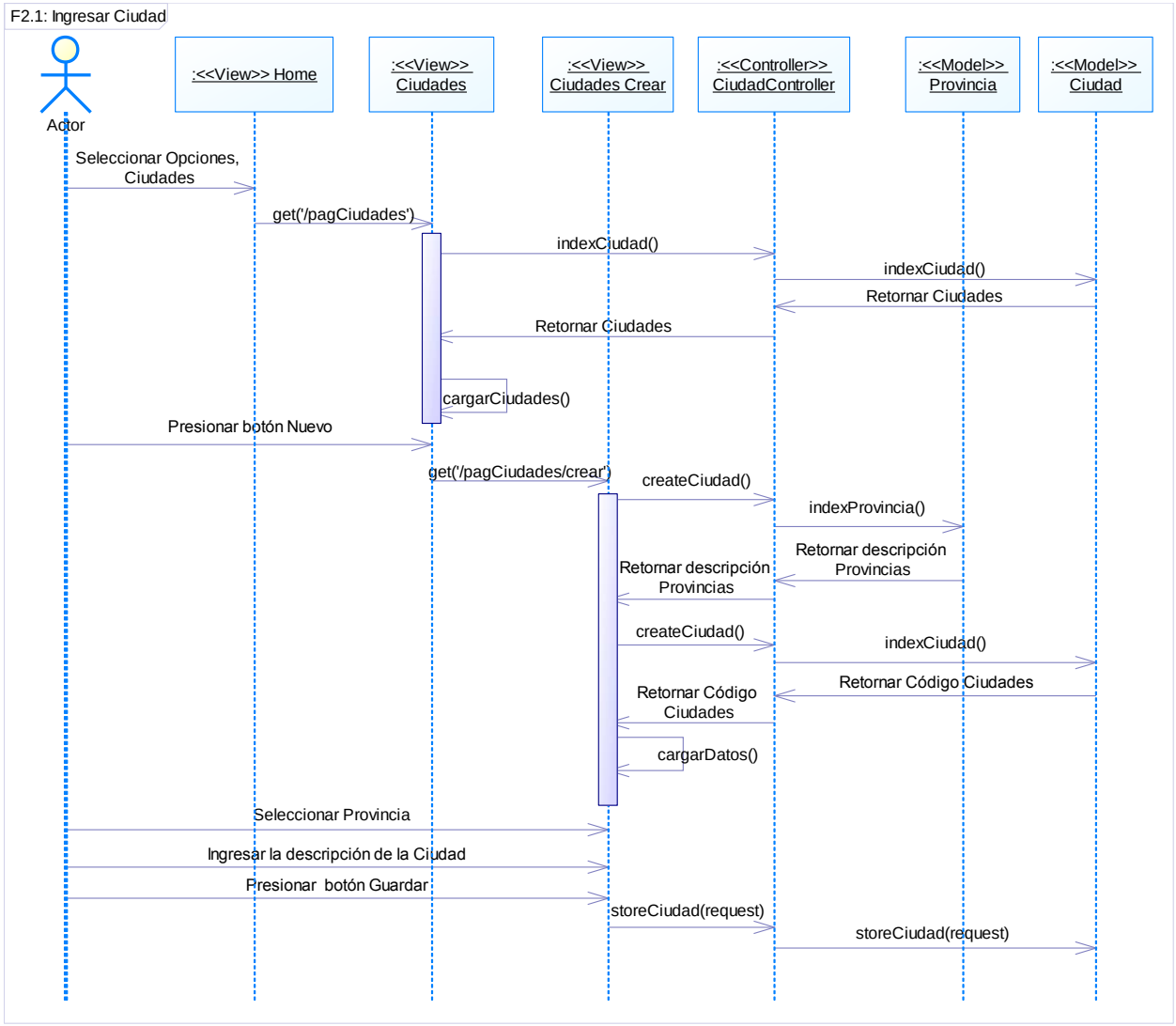


Figura 75. Diagrama Secuencia F2.1: Ingresar Ciudad

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Ciudades
2. El sistema despliega la ventana de Ciudades y presenta dentro de la ventana una tabla con todos los registros de Ciudades ingresados previamente
3. El actor presiona el botón Nuevo
4. El sistema dentro de la ventana Ciudades Crear, despliega las provincias existentes y el código de la Ciudad con la que se ingresará
5. El actor selecciona la provincia e ingresa el campo de la descripción de la Ciudad
6. El actor presiona el botón Guardar
7. El sistema almacena los datos

F2.2: Modificar Ciudad

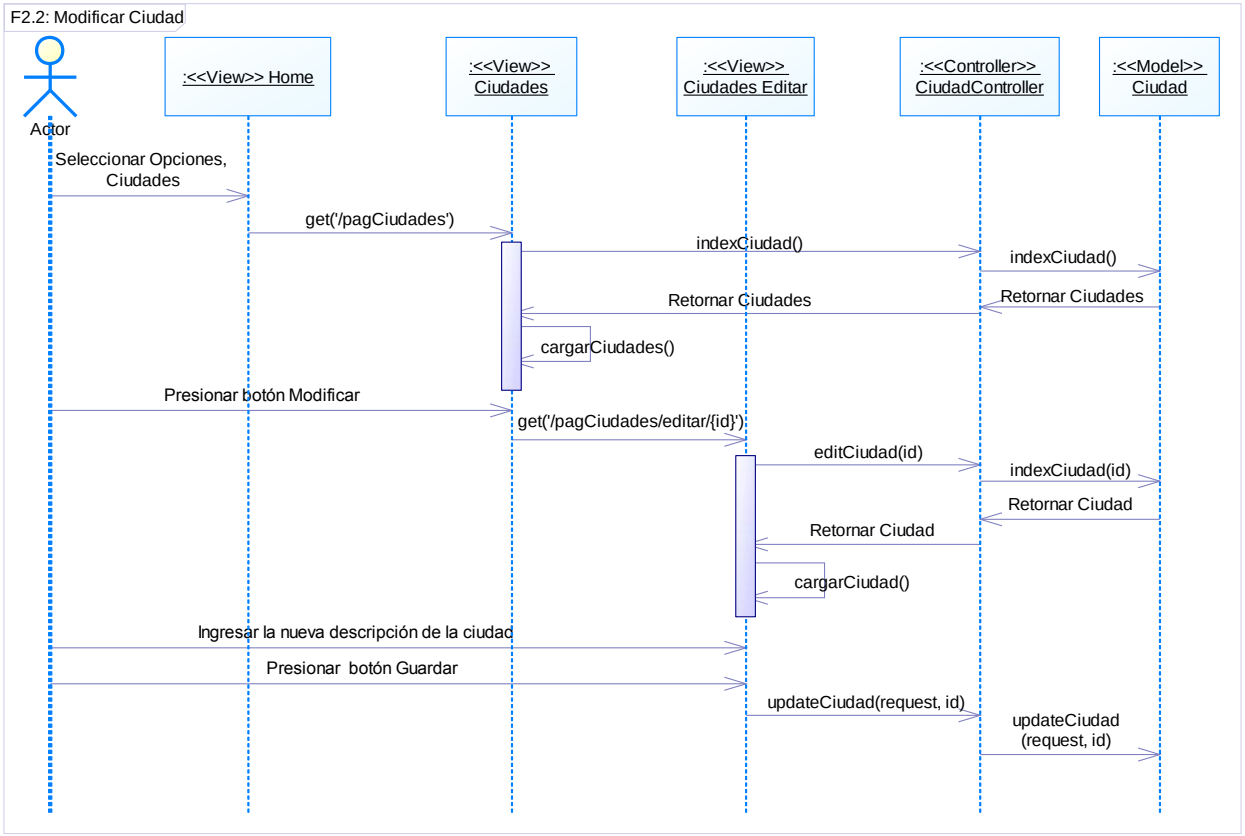


Figura 76. Diagrama Secuencia F2.2: Modificar Ciudad

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Ciudades
2. El sistema despliega la ventana de Ciudades
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Provincias ingresados previamente
4. El actor presiona el botón de lápiz (Modificar) en el registro que desee modificar
5. El sistema presenta la ventana Ciudades Editar con todos los campos llenos (código y Ciudad) y bloquea el campo código
6. El actor ingresa nuevamente el campo de la descripción de la Ciudad
7. El actor presiona el botón Guardar
8. El sistema modifica los datos

F2.3: Eliminar Ciudad

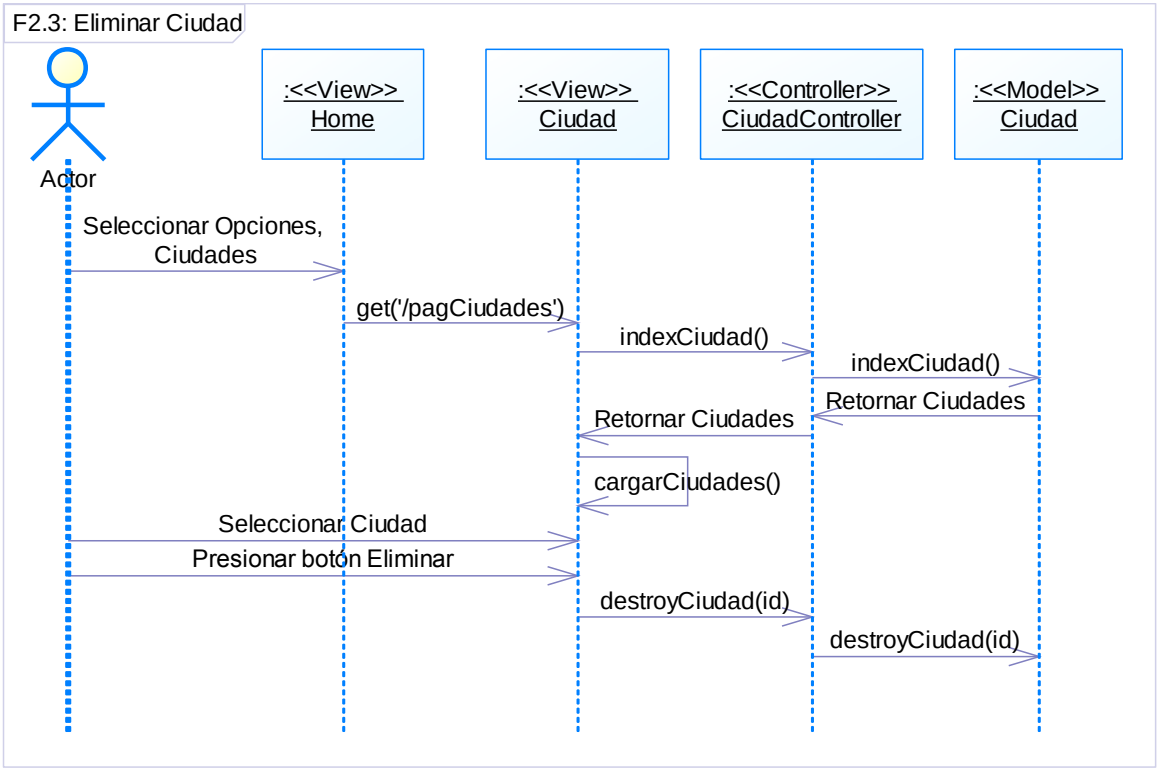


Figura 77. Diagrama Secuencia F2.3: Eliminar Ciudad

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Ciudades
2. El sistema despliega la ventana de Ciudad
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Ciudades ingresados previamente
4. El actor presiona el botón de basurero (Eliminar) en el registro que desee eliminar
5. El sistema elimina el registro seleccionado

F2.4: Consultar Ciudad General

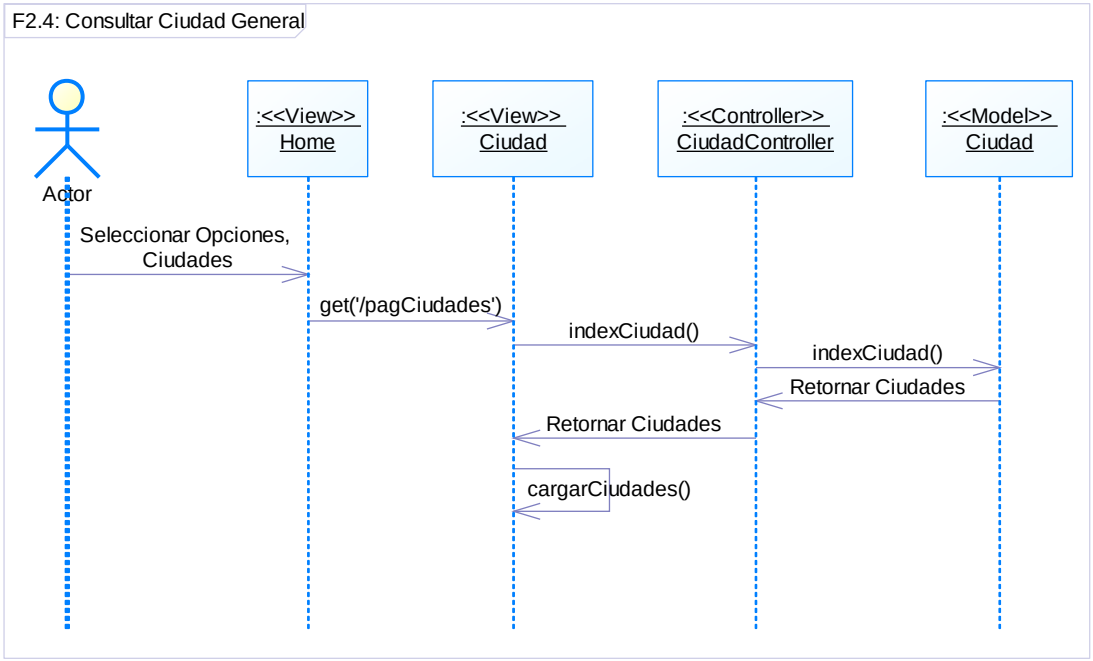


Figura 78. Diagrama Secuencia F2.4: Consultar Ciudad General

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Ciudades
2. El sistema despliega la ventana de Ciudades
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Ciudades ingresados previamente

F2.4.1: Consultar Ciudad por Parámetros

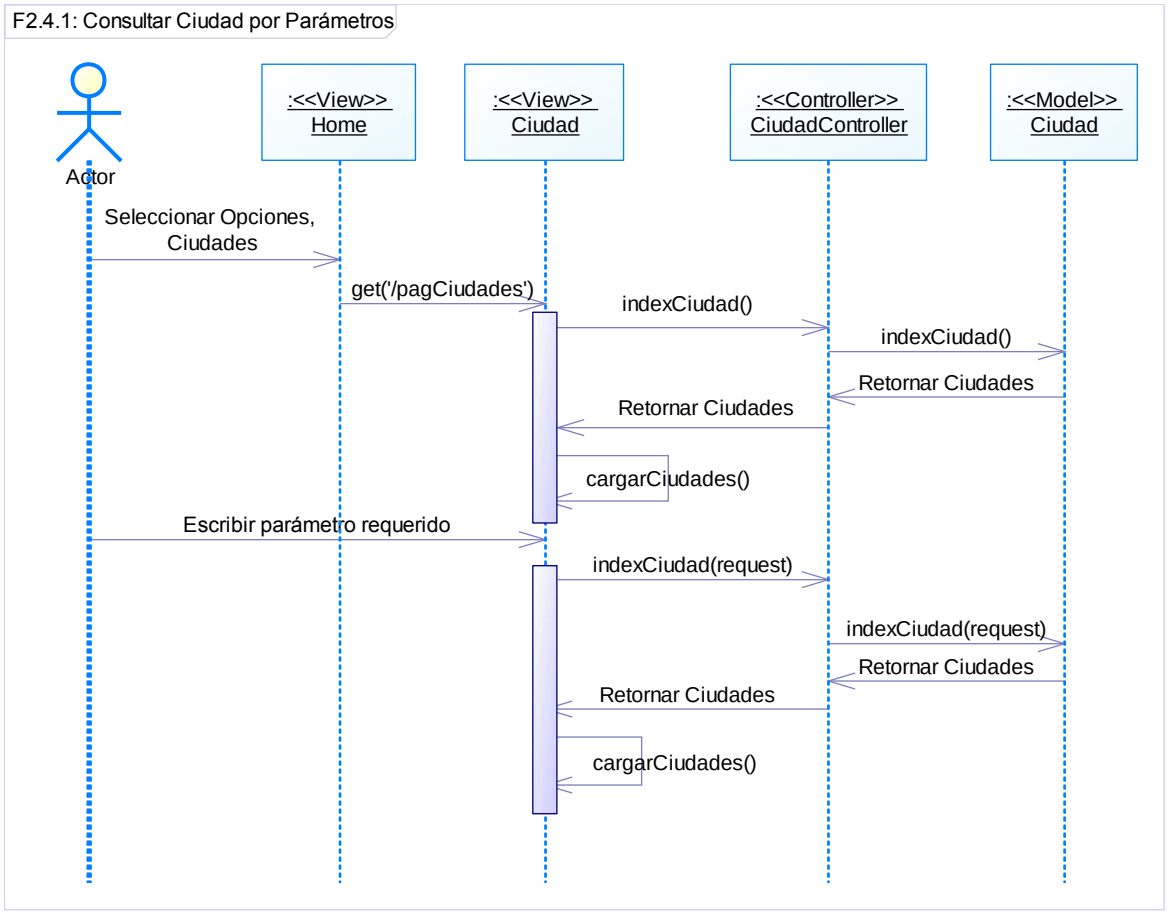


Figura 79. Diagrama Secuencia F2.4.1: Consultar Ciudad por Parámetros

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Ciudades
2. El sistema despliega la ventana de Ciudades
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Ciudades ingresados previamente
4. El actor escribe la palabra que desea en la barra de búsqueda
5. El sistema presenta dentro de la tabla del punto 3 todos los registros que cumplen con el parámetro ingresado por el actor

F3: Gestionar Universidad

F3.1: Ingresar Universidad

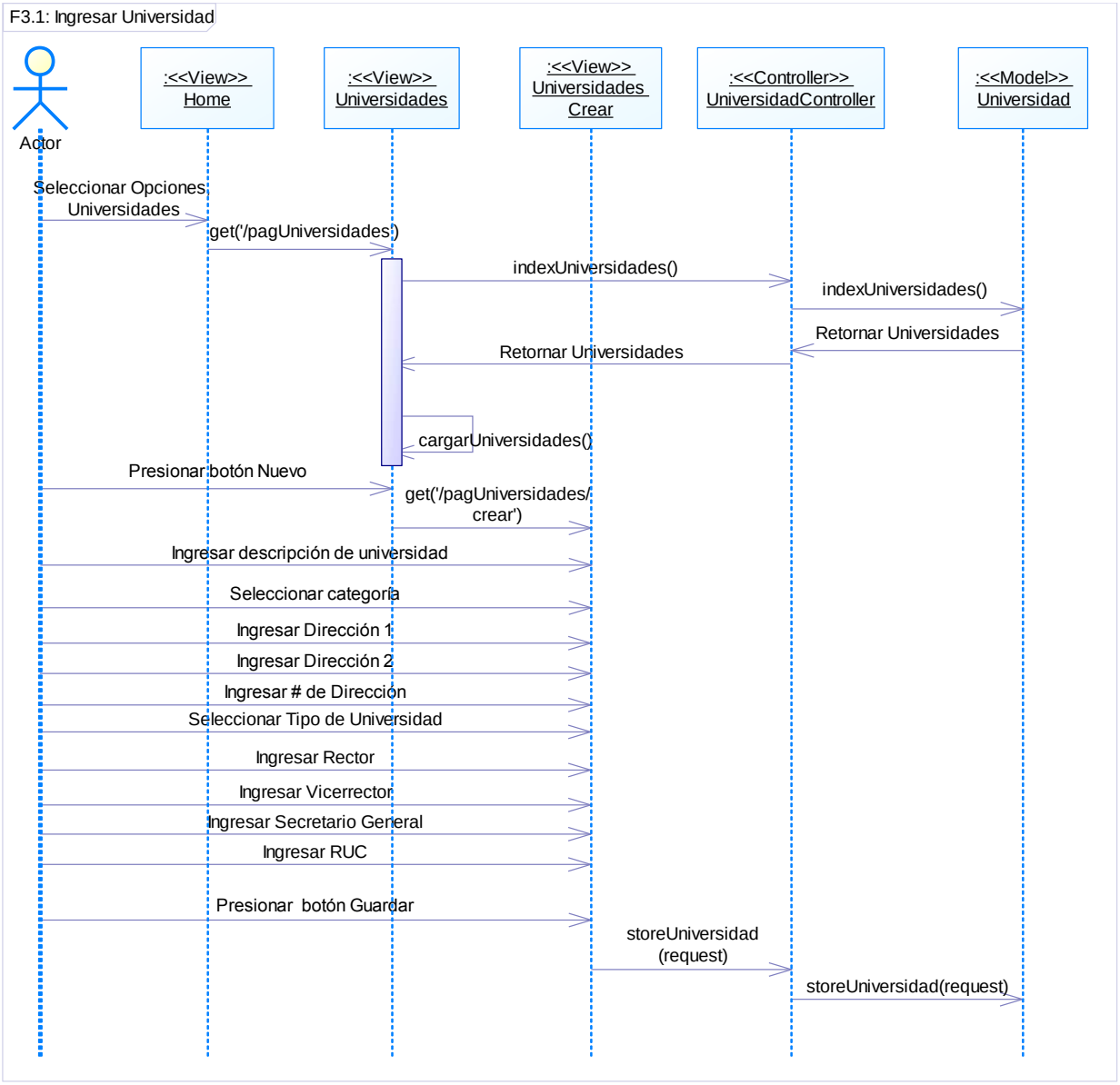


Figura 80. Diagrama Secuencia F3.1: Ingresar Universidad

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Universidades
2. El sistema despliega la ventana de Universidades
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Universidades ingresados previamente
4. El actor presiona el botón Nuevo
5. El sistema presenta dentro de la ventana Universidades Crear, las categorías (A, B, C, D) y los tipos (PR: Privada y PU: Pública)
6. El actor ingresa los campos de Descripción, Dirección 1 y 2, el número de la Dirección, el Rector, Vicerrector, Secretario General, RUC de la Universidad y selecciona los campos Categoría y Tipo.
7. El actor presiona el botón Guardar
8. El sistema almacena los datos

F3.2: Modificar Universidad

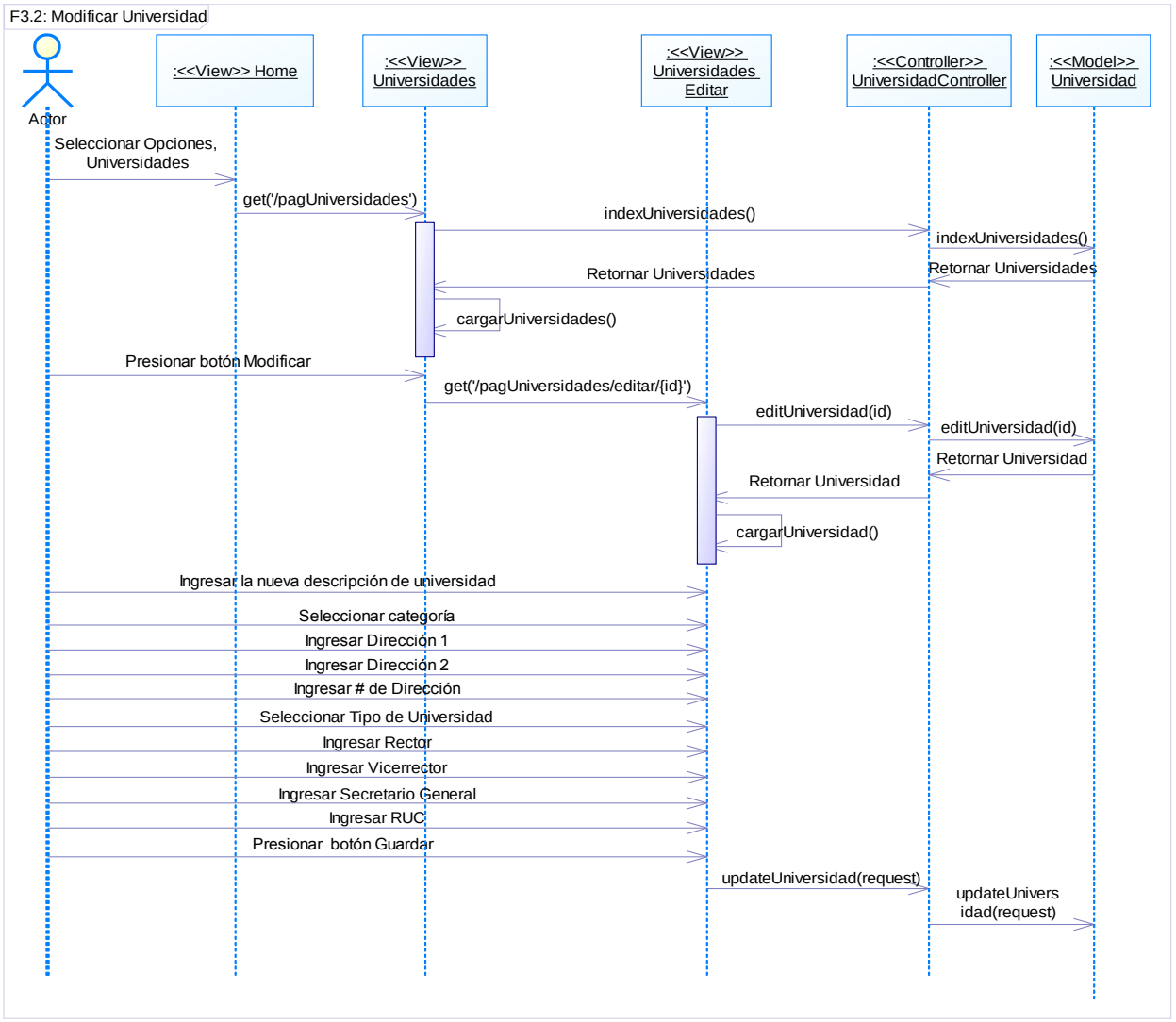


Figura 81. Diagrama Secuencia F3.2: Modificar Universidad

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Universidades
2. El sistema despliega la ventana de Universidades
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Universidades ingresados previamente

4. El actor presiona el botón de lápiz (Modificar) en el registro que desee modificar
5. El sistema presenta la ventana Universidades Editar con todos los campos llenos (Descripción, Categoría, Dirección 1 y 2, Tipo, el número de la Dirección, el Rector, Vicerrector, Secretario General, RUC de la Universidad)
6. El actor ingresa nuevamente nuevos datos en los campos que desee
7. El actor presiona el botón Guardar
8. El sistema modifica los datos

F3.3: Eliminar Universidad

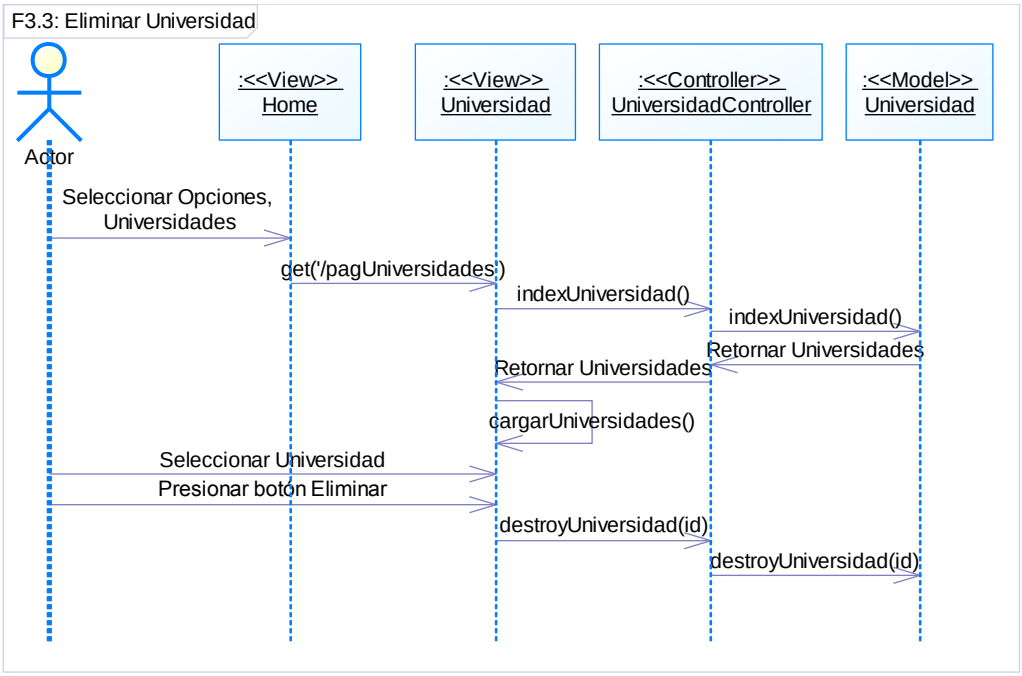


Figura 82. Diagrama Secuencia F3.3: Eliminar Universidad

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Universidades
2. El sistema despliega la ventana de Universidades
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Universidades ingresados previamente
4. El actor presiona el botón de basurero (Eliminar) en el registro que desee eliminar

5. El sistema elimina el registro seleccionado

F3.4: Consultar Universidad General

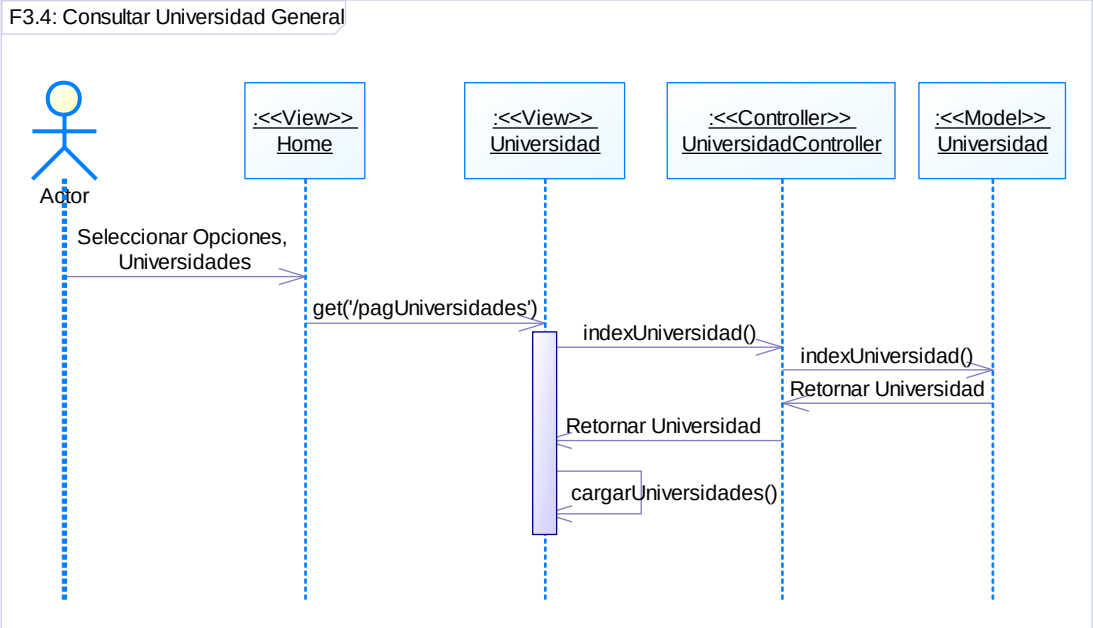


Figura 83. Diagrama Secuencia F3.4: Consultar Universidad General

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Universidades
2. El sistema despliega la ventana de Universidades
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Universidades ingresados previamente

F3.4.1: Consultar Universidad por Parámetros

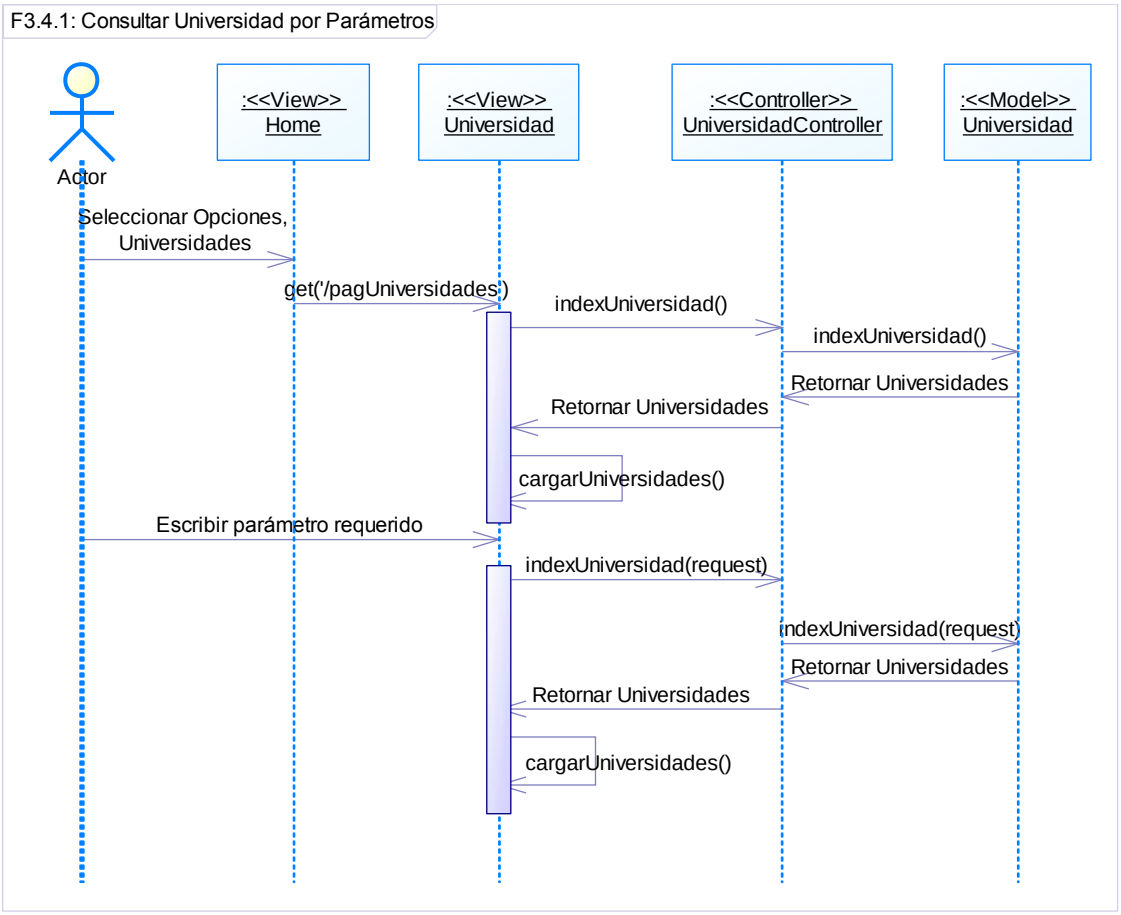


Figura 84. Diagrama Secuencia F3.4.1: Consultar Universidad por Parámetros

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Universidades
2. El sistema despliega la ventana de Universidades
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Universidades ingresados previamente
4. El actor escribe la palabra que desea en la barra de búsqueda
5. El sistema presenta dentro de la tabla del punto 3 todos los registros que cumplen con el parámetro ingresado por el actor

F4: Gestionar Sede

F4.1: Ingresar Sede

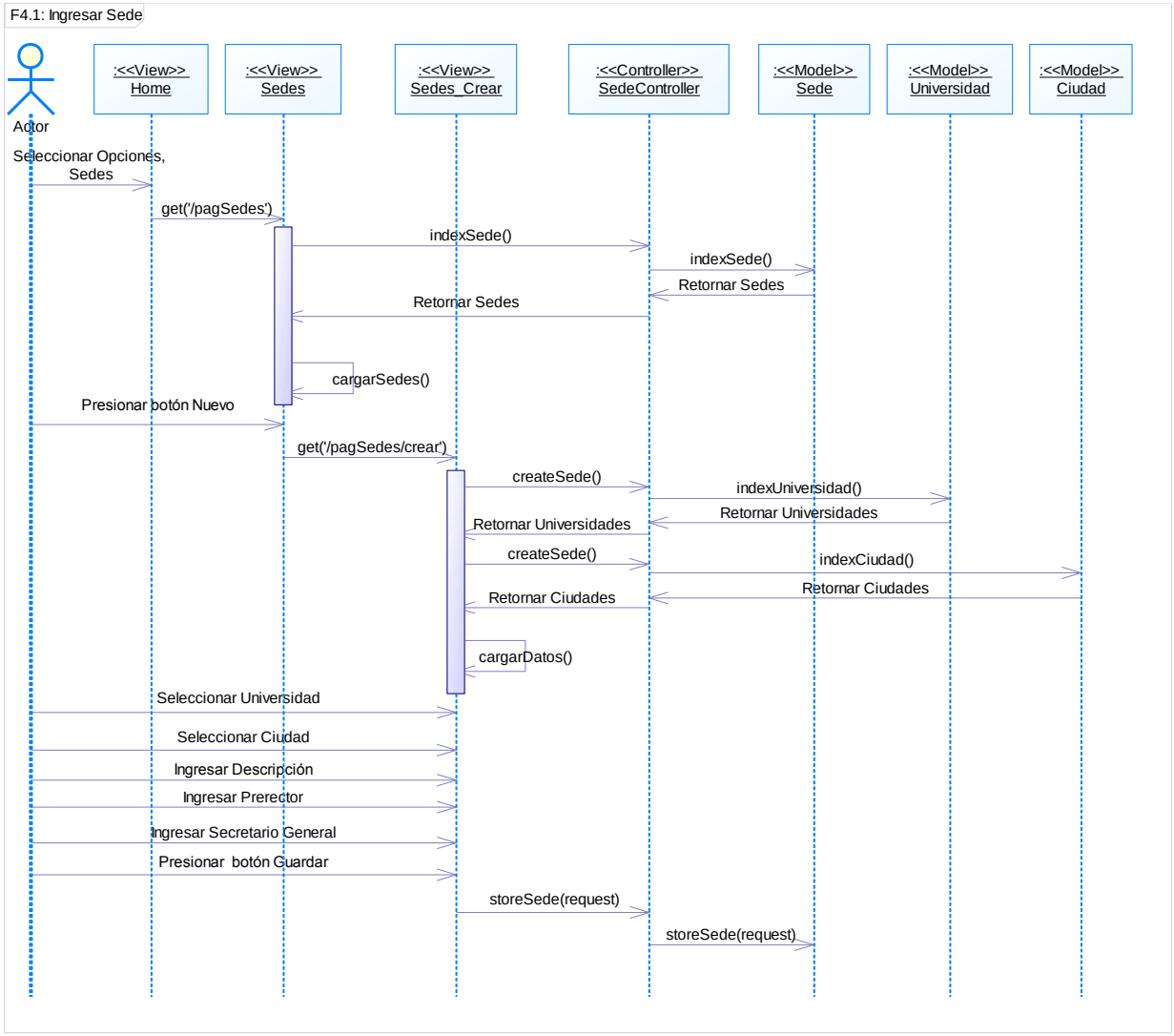


Figura 85. Diagrama Secuencia F4.1: Ingresar Sede

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Sedes
2. El sistema despliega la ventana de Sedes
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Sedes ingresados previamente
4. El actor presiona el botón Nuevo
5. El sistema dentro de la ventana Sedes Crear, despliega las Universidades y las Ciudades, permite ingresar la Descripción de la Sede, el Prerector y Secretario General.
6. El actor selecciona los campos de las Universidades y las Ciudades e ingresa la Descripción de la Sede, el Prerector y Secretario General.
7. El actor presiona el botón Guardar
8. El sistema almacena los datos

F4.2: Modificar Sede

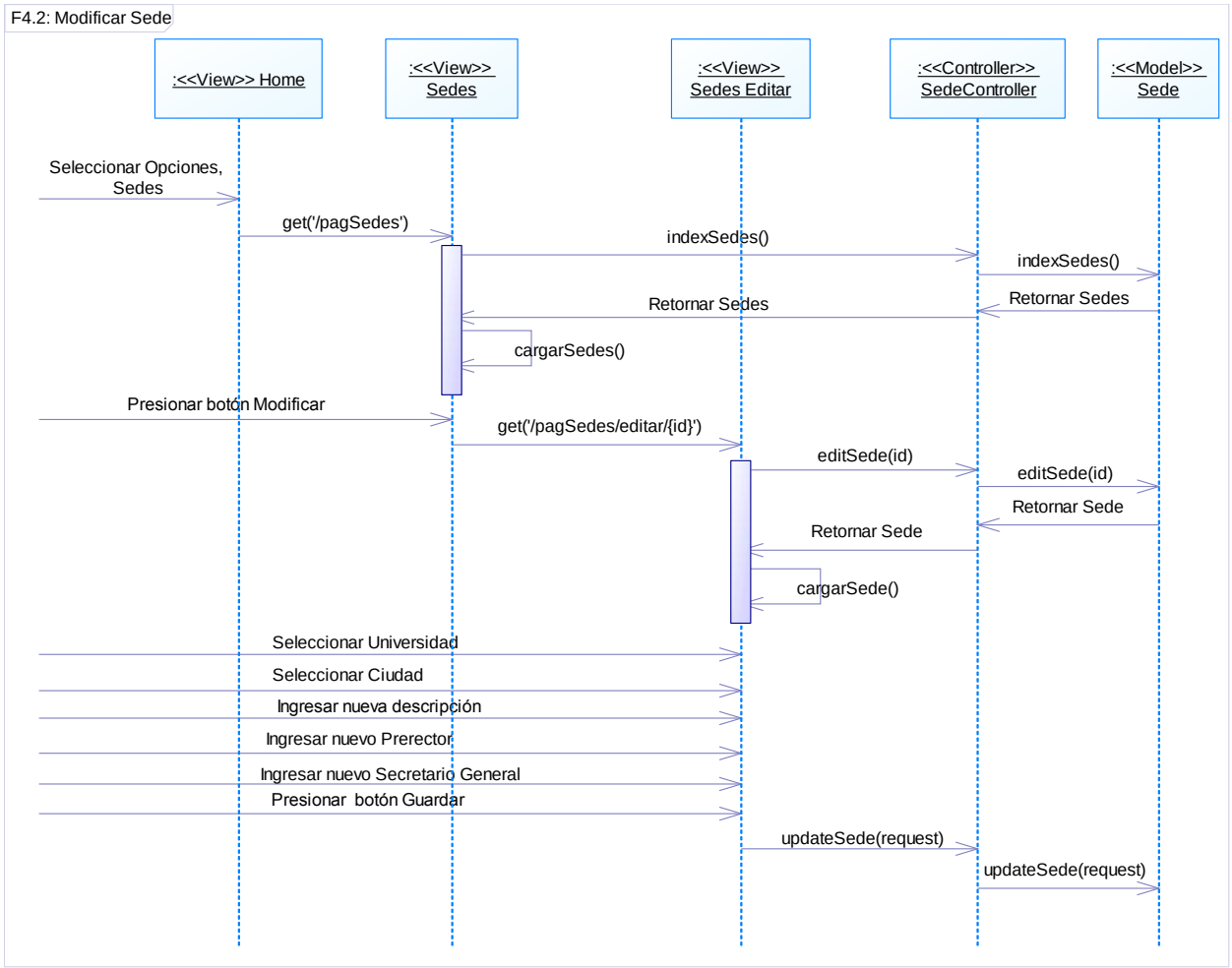


Figura 86. Diagrama Secuencia F4.2: Modificar Sede

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Sedes
2. El sistema despliega la ventana de Sedes
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Sedes ingresados previamente
4. El actor presiona el botón de lápiz (Modificar) en el registro que desee modificar
5. El sistema presenta la ventana Sedes Editar con todos los campos llenos (Universidad, Ciudad, Descripción de Sede, Prerector y Secretario General)
6. El actor ingresa nuevamente nuevos datos en los campos que desee
7. El actor presiona el botón Guardar
8. El sistema modifica los datos

F4.3: Eliminar Sede

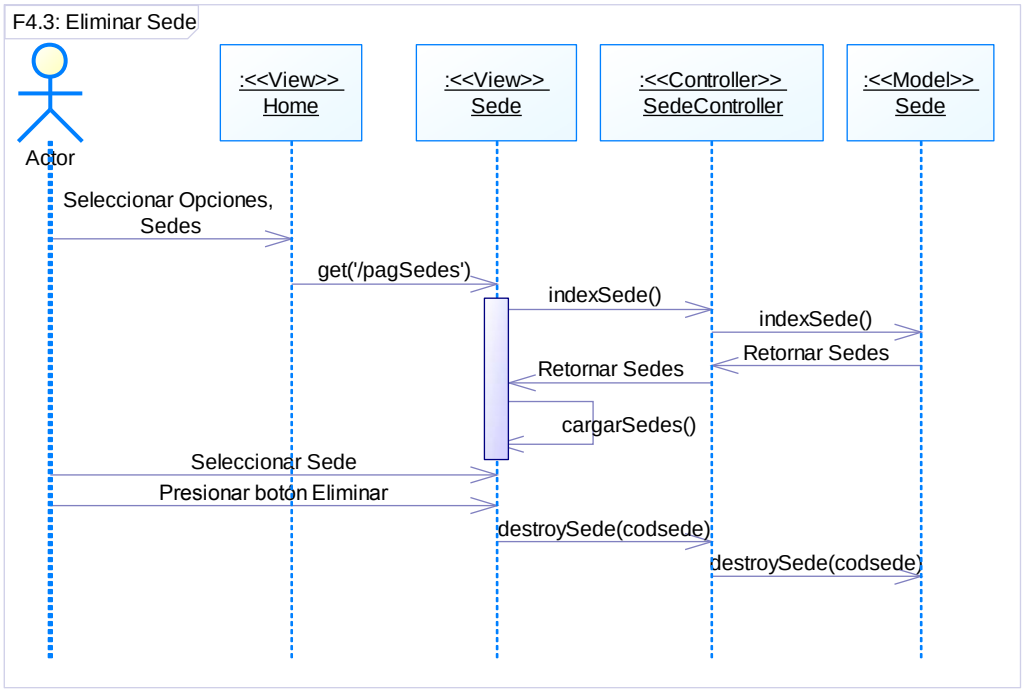


Figura 87. Diagrama Secuencia F4.3: Eliminar Sede

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Sedes
2. El sistema despliega la ventana de Sedes
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Sedes ingresados previamente
4. El actor presiona el botón de basurero (Eliminar) en el registro que desee eliminar
5. El sistema elimina el registro seleccionado

F4.4: Consultar Sede General

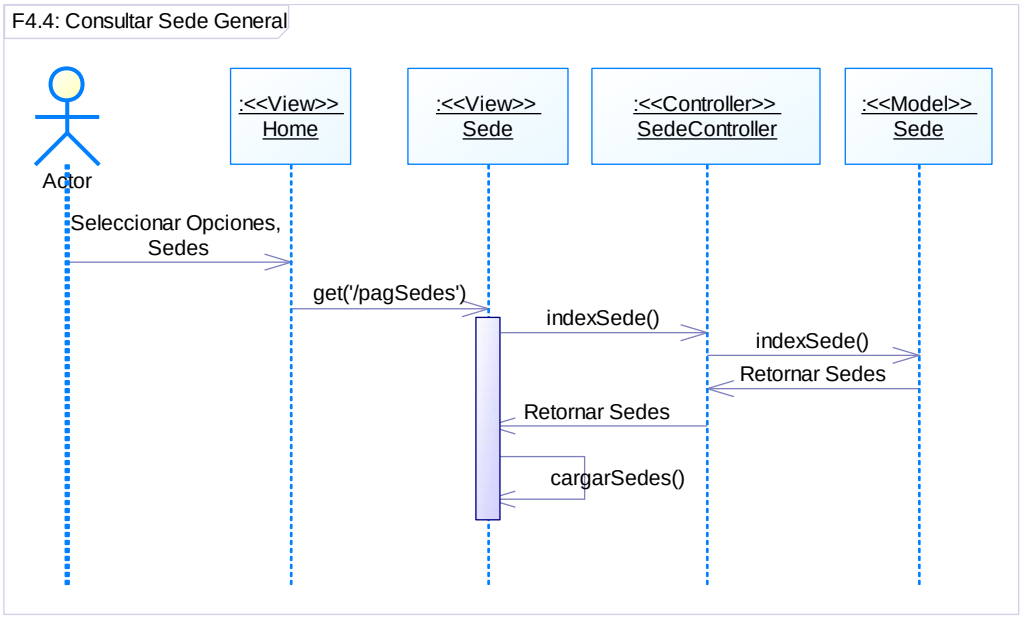


Figura 88. Diagrama Secuencia F4.4: Consultar Sede General

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Sedes
2. El sistema despliega la ventana de Sedes
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Sedes ingresados previamente

F4.4.1: Consultar Sede por Parámetros

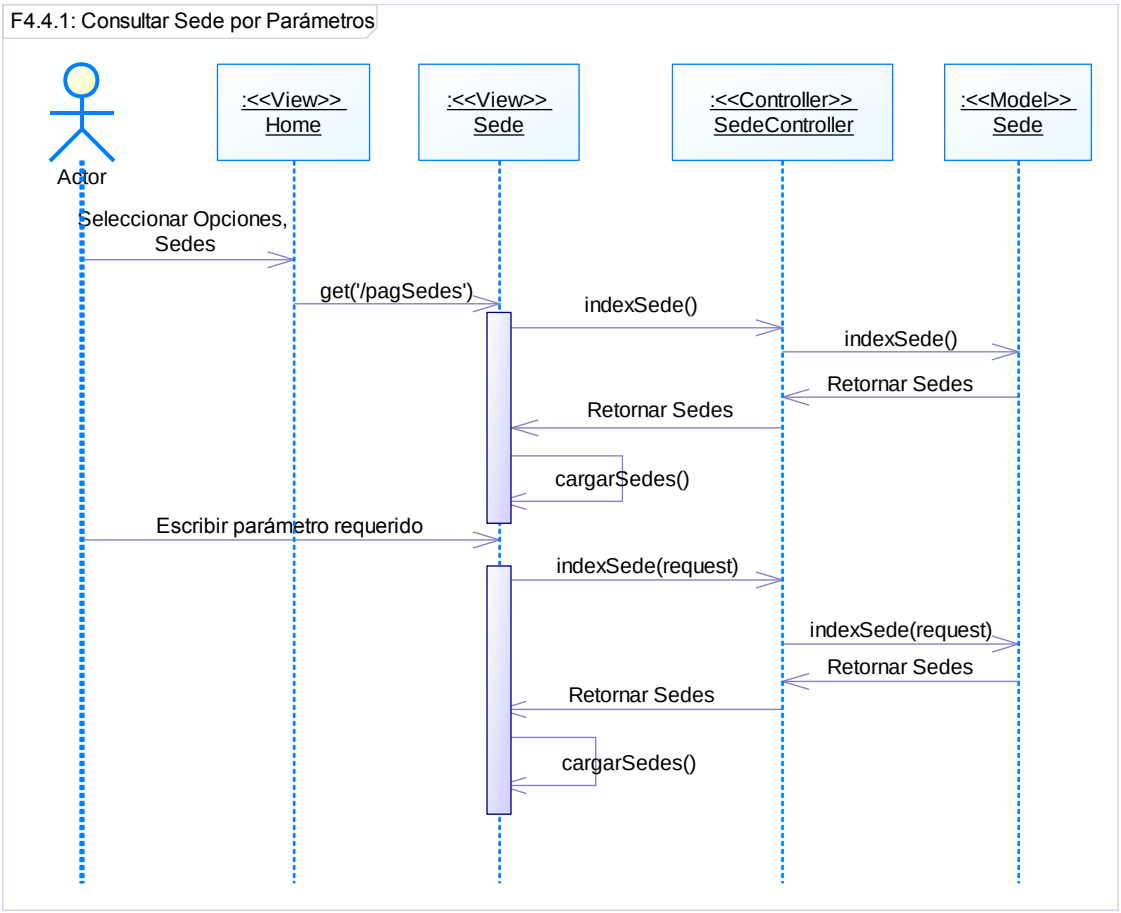


Figura 89. Diagrama Secuencia F4.4.1: Consultar Sede por Parámetros

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Sedes
2. El sistema despliega la ventana de Sedes
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Sedes ingresados previamente
4. El actor escribe la palabra que desea en la barra de búsqueda
5. El sistema presenta dentro de la tabla del punto 3 todos los registros que cumplen con el parámetro ingresado por el actor

F5: Gestionar Periodo

F5.1: Ingresar Periodo

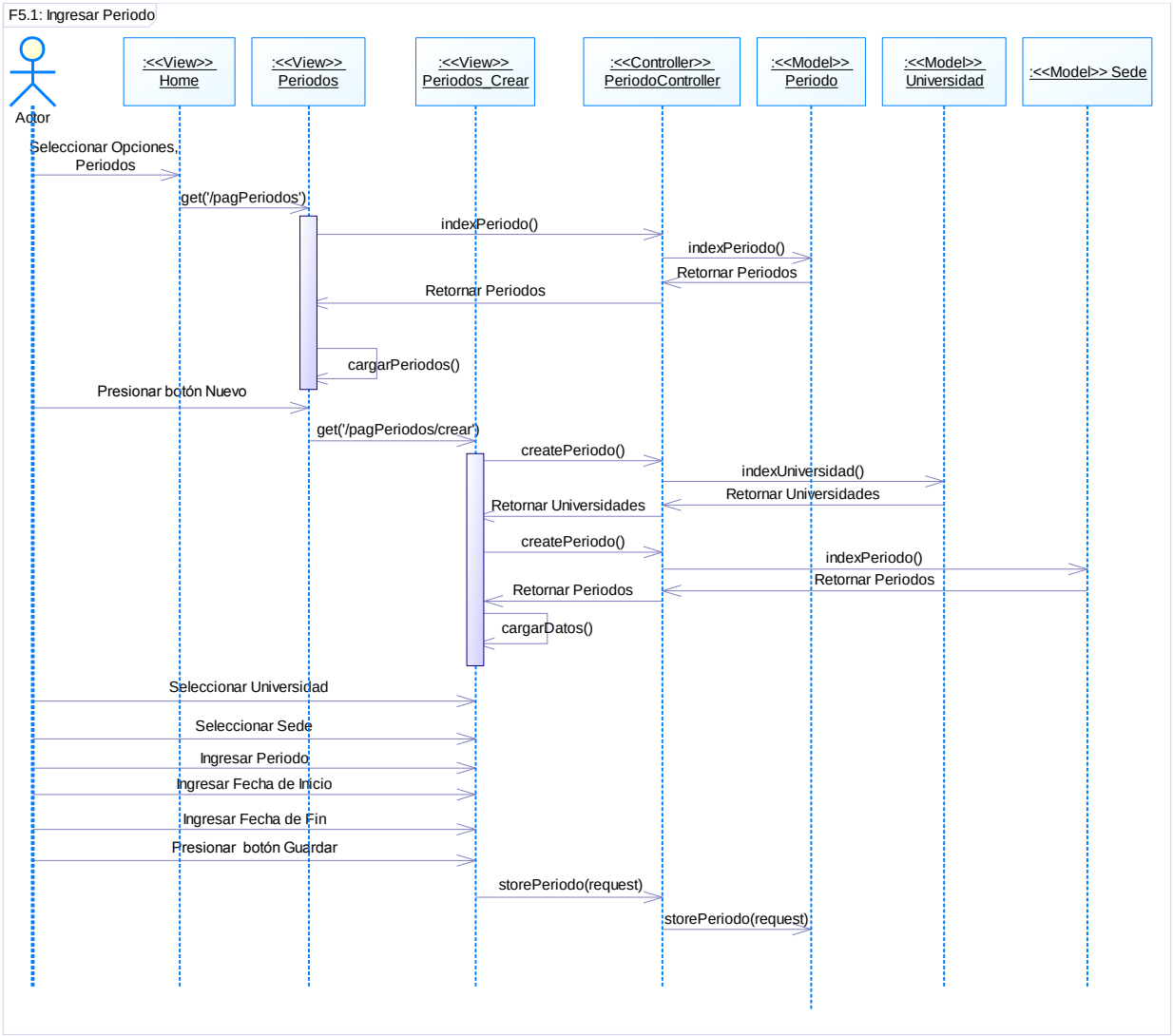


Figura 90. Diagrama Secuencia F5.1: Ingresar Periodo

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Periodos
2. El sistema despliega la ventana de Periodos
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Periodos ingresados previamente
4. El actor presiona el botón Nuevo
5. El sistema dentro de la ventana Periodos Crear, despliega las Universidades y las Sedes, permite ingresar el Periodo, la Fecha de Inicio y la Fecha de Fin.
6. El actor selecciona los campos de las Universidades y las Sedes, e ingresa el Periodo, la Fecha de Inicio y la Fecha de Fin.
7. El actor presiona el botón Guardar
8. El sistema almacena los datos

F5.2: Modificar Periodo

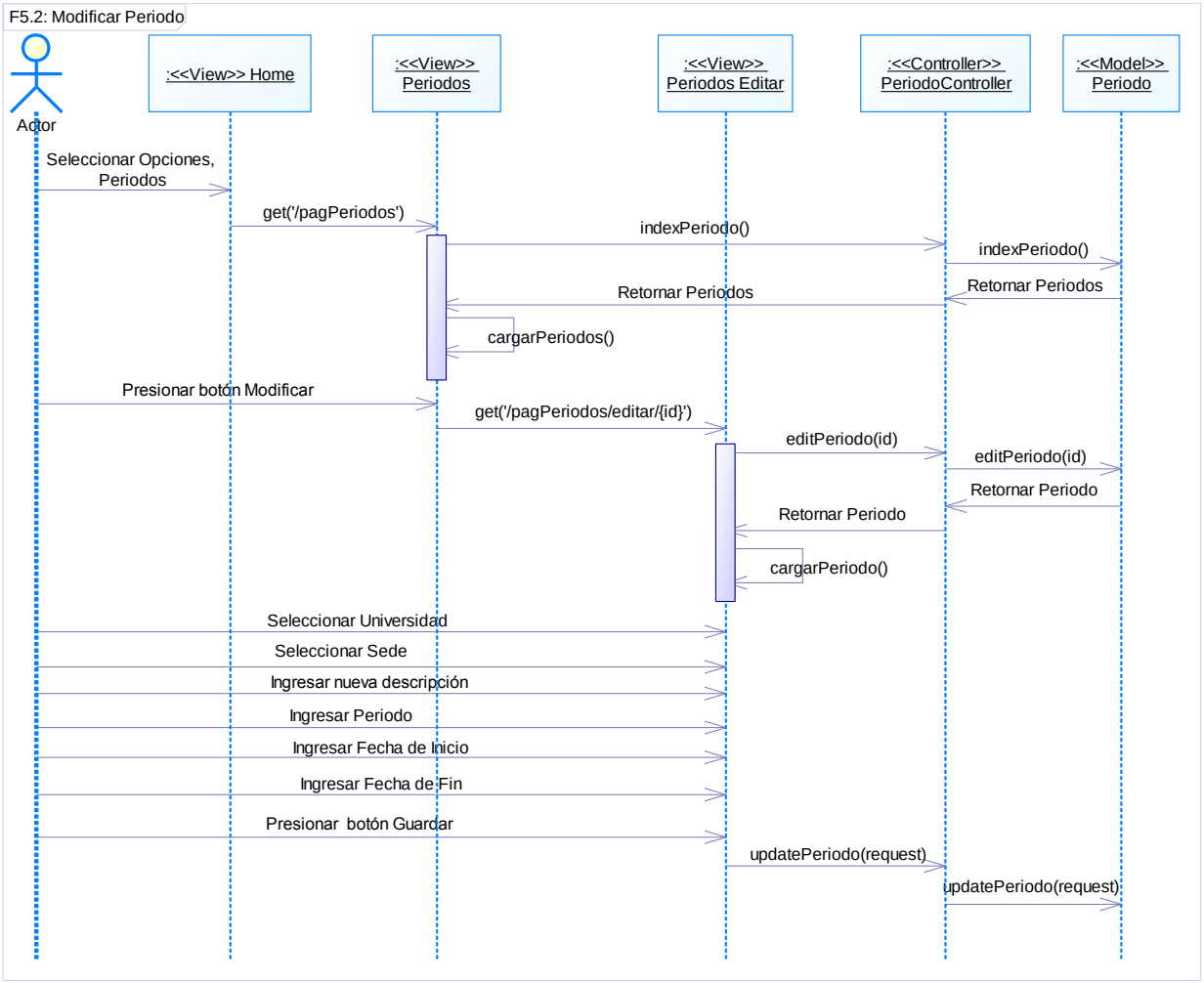


Figura 91. Diagrama Secuencia F5.2: Modificar Periodo

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Periodos
2. El sistema despliega la ventana de Periodos
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Periodos ingresados previamente
4. El actor presiona el botón de lápiz (Modificar) en el registro que desee modificar
5. El sistema presenta la ventana Sedes Editar con todos los campos llenos (Universidades, Sedes, Periodo, la Fecha de Inicio, la Fecha de Fin y el estado del Periodo)
6. El actor ingresa nuevamente nuevos datos en los campos que desee
7. El actor presiona el botón Guardar
8. El sistema modifica los datos

F5.3: Eliminar Periodo

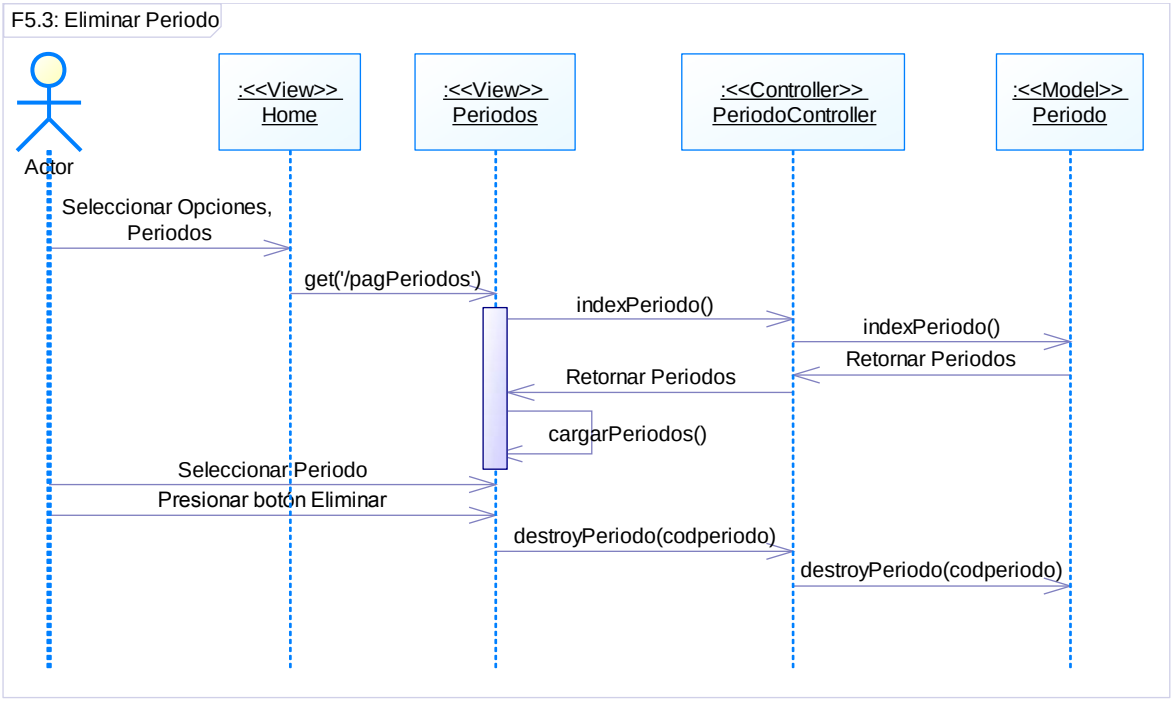


Figura 92. Diagrama Secuencia F5.3: Eliminar Periodo

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Periodos
2. El sistema despliega la ventana de Periodos
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Periodos ingresados previamente
4. El actor presiona el botón de basurero (Eliminar) en el registro que desee eliminar
5. El sistema elimina el registro seleccionado

F5.4: Consultar Periodo General

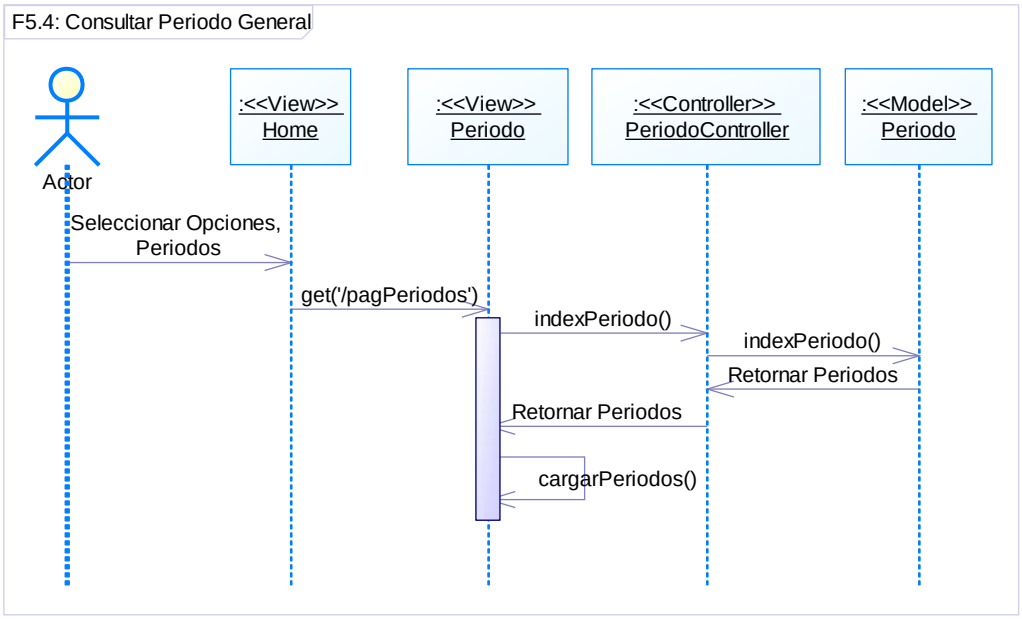


Figura 93. Diagrama Secuencia F5.4: Consultar Periodo General

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Periodos
2. El sistema despliega la ventana de Periodos

3. El sistema presenta dentro de la ventana una tabla con todos los registros de Periodos ingresados previamente

F5.4.1: Consultar Periodo por Parámetros

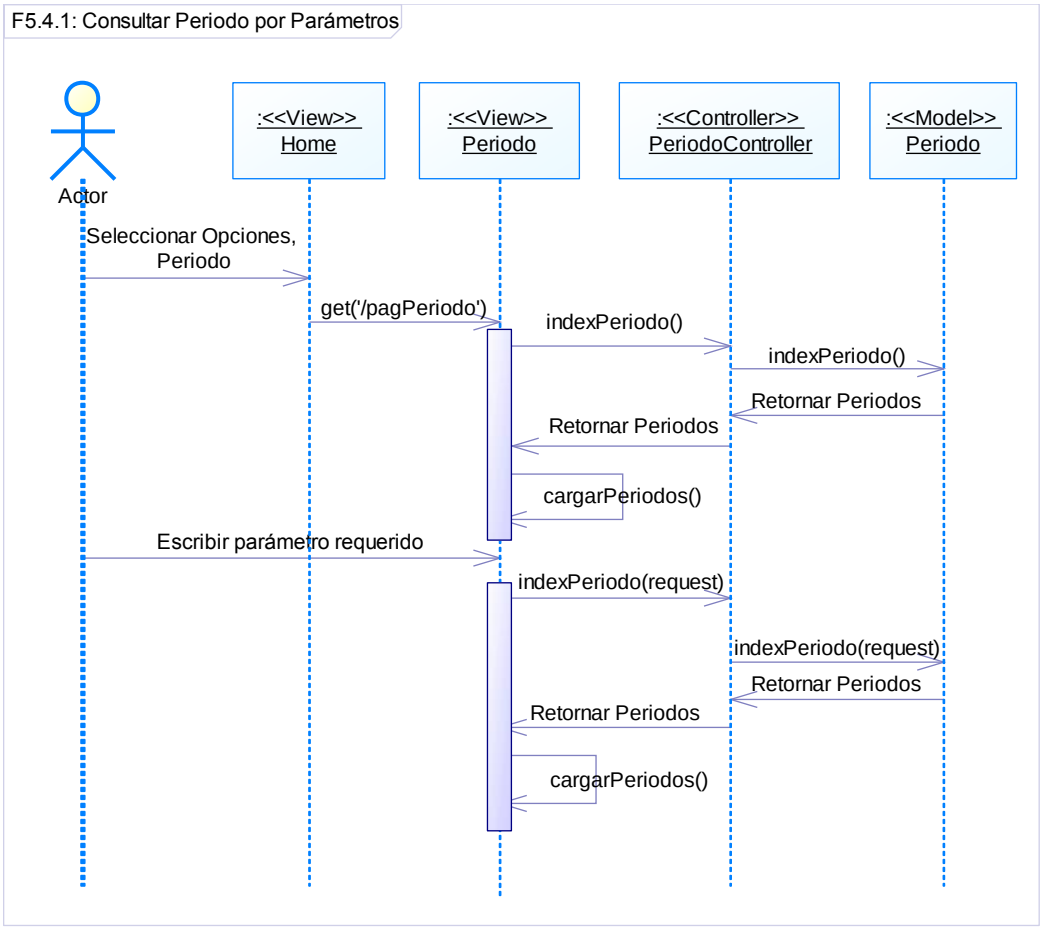


Figura 94. Diagrama Secuencia F5.4.1: Consultar Periodo por Parámetros

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Periodos
2. El sistema despliega la ventana de Periodos
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Periodos ingresados previamente
4. El actor escribe la palabra que desea en la barra de búsqueda

5. El sistema presenta dentro de la tabla del punto 3 todos los registros que cumplen con el parámetro ingresado por el actor

F6: Gestionar Facultad

F6.1: Ingresar Facultad

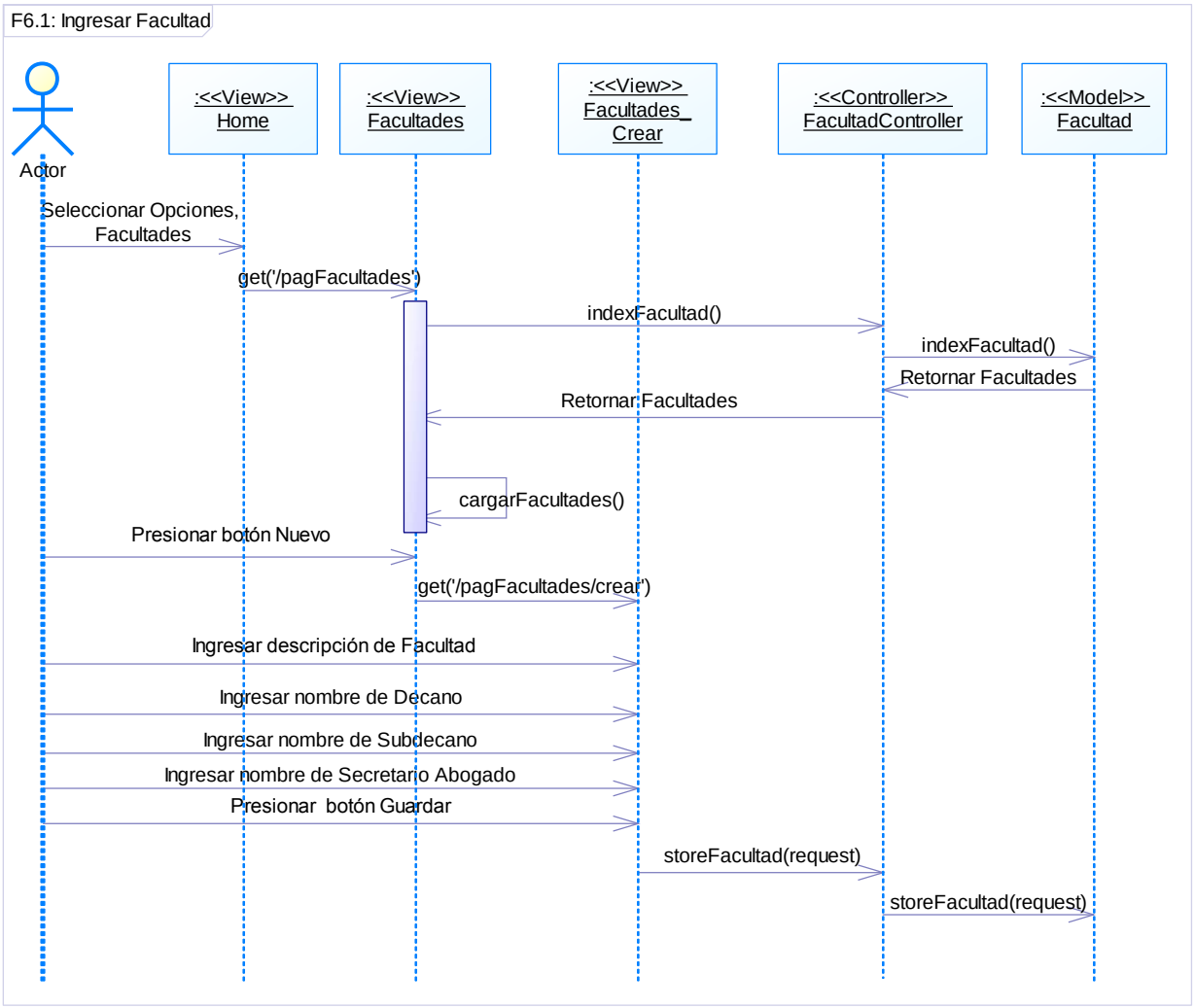


Figura 95. Diagrama Secuencia F6.1: Ingresar Facultad

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Facultades
2. El sistema despliega la ventana de Facultades
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Facultades ingresados previamente
4. El actor presiona el botón Nuevo
5. El sistema dentro de la ventana Facultades Crear, despliega el Código de la Facultad y permite ingresar la Descripción, el Decano, Subdecano y Secretario Abogado.
6. El actor ingresa la Descripción, el Decano, Subdecano y Secretario Abogado y presiona el botón Guardar
7. El sistema almacena los datos

F6.2: Modificar Facultad

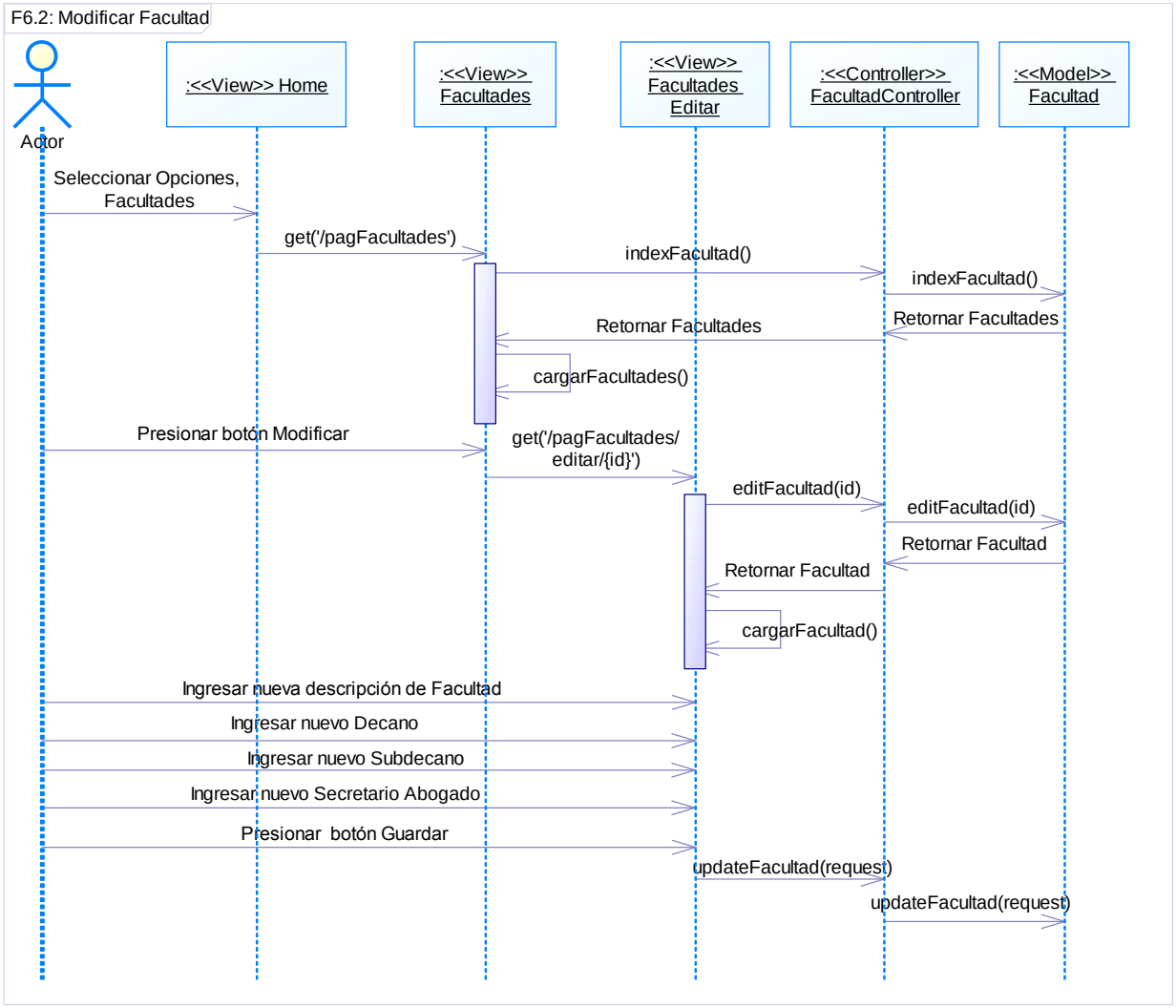


Figura 96. Diagrama Secuencia F6.2: Modificar Facultad

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Facultades
2. El sistema despliega la ventana de Facultades
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Facultades ingresados previamente
4. El actor presiona el botón de lápiz (Modificar) en el registro que desee modificar
5. El sistema presenta la ventana Facultades Editar con todos los campos llenos (Código de la Facultad , la Descripción, el Decano, Subdecano y Secretario Abogado)
6. El actor ingresa nuevamente nuevos datos en los campos que desee
7. El actor presiona el botón Guardar
8. El sistema modifica los datos

F6.3: Eliminar Facultad

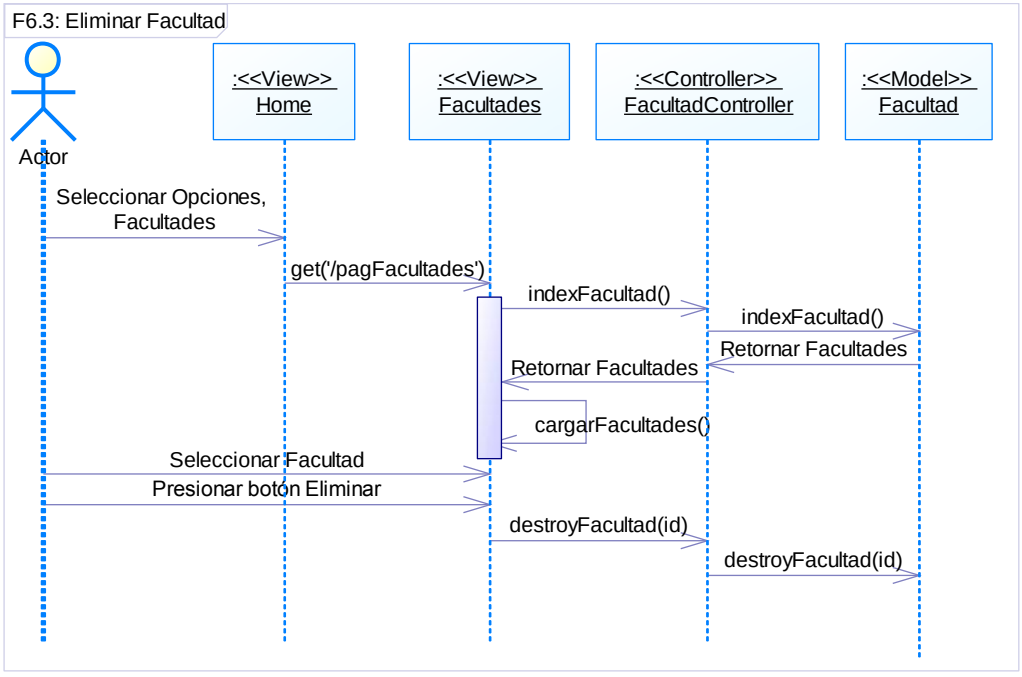


Figura 97. Diagrama Secuencia F6.3: Eliminar Facultad

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Facultades
2. El sistema despliega la ventana de Facultades
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Facultades ingresados previamente
4. El actor presiona el botón de basurero (Eliminar) en el registro que desee eliminar
5. El sistema elimina el registro seleccionado

F6.4: Consultar Facultad General

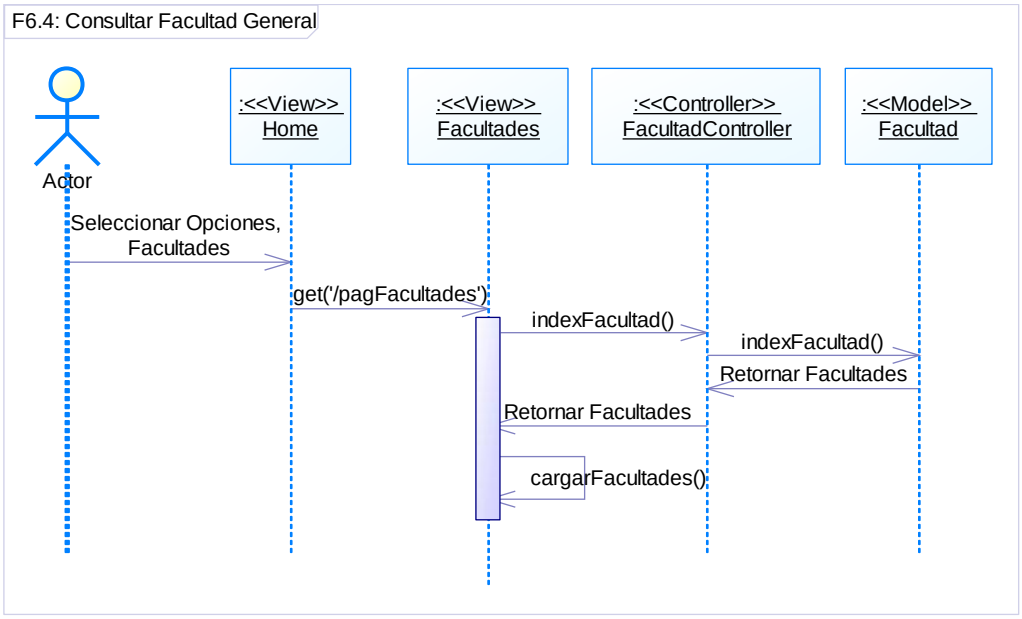


Figura 98. Diagrama Secuencia F6.4: Consultar Facultad General

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Facultades
2. El sistema despliega la ventana de Facultades
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Facultades ingresados previamente

F6.4.1: Consultar Facultad por Parámetros

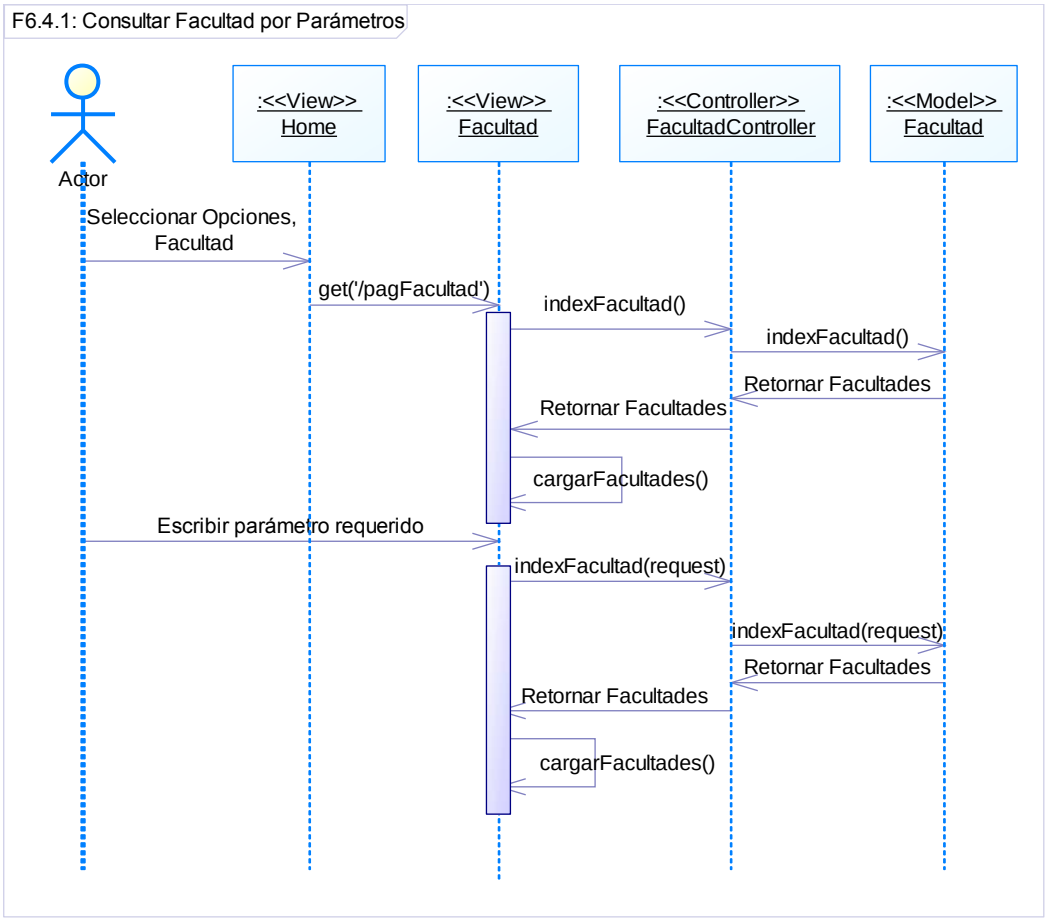


Figura 99. Diagrama Secuencia F6.4.1: Consultar Facultad por Parámetros

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Facultades
2. El sistema despliega la ventana de Facultades
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Facultades ingresados previamente
4. El actor escribe la palabra que desea en la barra de búsqueda
5. El sistema presenta dentro de la tabla del punto 3 todos los registros que cumplen con el parámetro ingresado por el actor

F7: Gestionar Escuela

F7.1: Ingresar Escuela

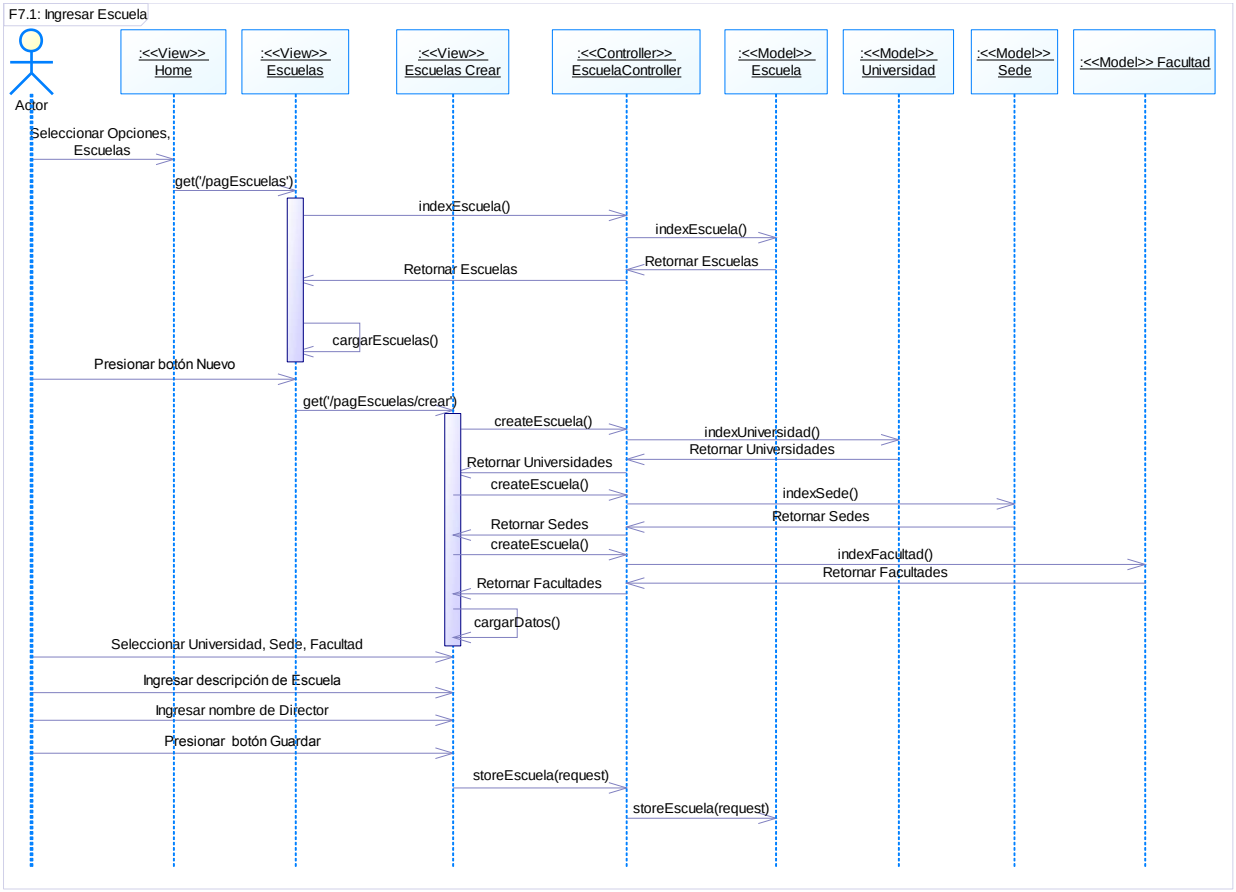


Figura 100. Diagrama Secuencia F7.1: Ingresar Escuela

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Escuelas
2. El sistema despliega la ventana de Escuelas
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Escuelas ingresados previamente
4. El actor presiona el botón Nuevo

5. El sistema dentro de la ventana Facultades Crear, despliega Universidad, Sede, Facultad y permite ingresar la Descripción de la Escuela y el Director
6. El actor selecciona la Universidad, Sede, Facultad e ingresa la Descripción de la Escuela y el Director
7. El actor presiona el botón Guardar
8. El sistema almacena los datos

F7.2: Modificar Escuela

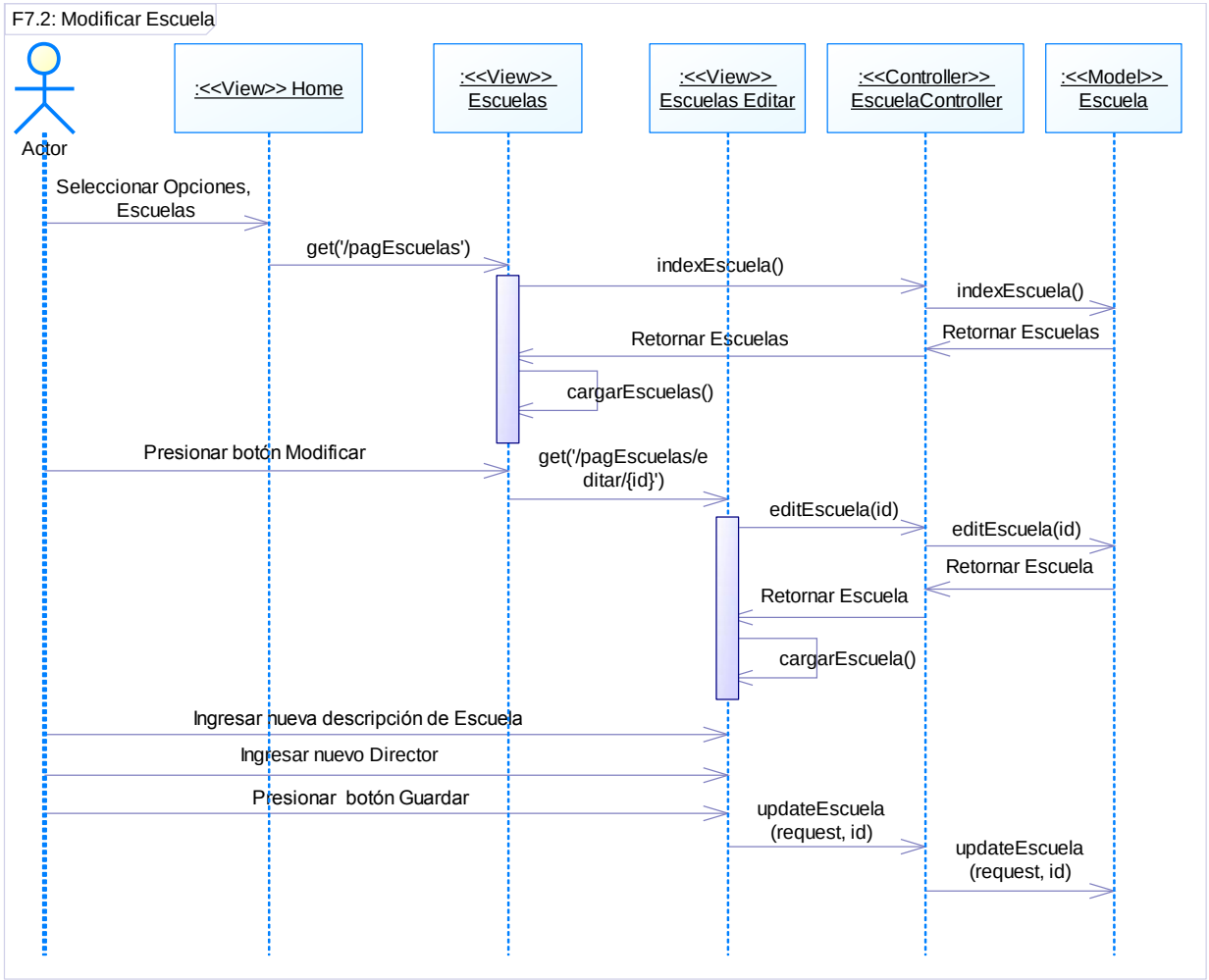


Figura 101. Diagrama Secuencia F7.2: Modificar Escuela

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Escuelas
2. El sistema despliega la ventana de Escuelas
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Escuelas ingresados previamente
4. El actor presiona el botón de lápiz (Modificar) en el registro que desee modificar
5. El sistema presenta la ventana Escuelas Editar con todos los campos llenos, Descripción de Escuela y nombre del Director
6. El actor ingresa nuevamente nuevos datos en los campos que desee
7. El actor presiona el botón Guardar
8. El sistema modifica los datos

F7.3: Eliminar Escuela

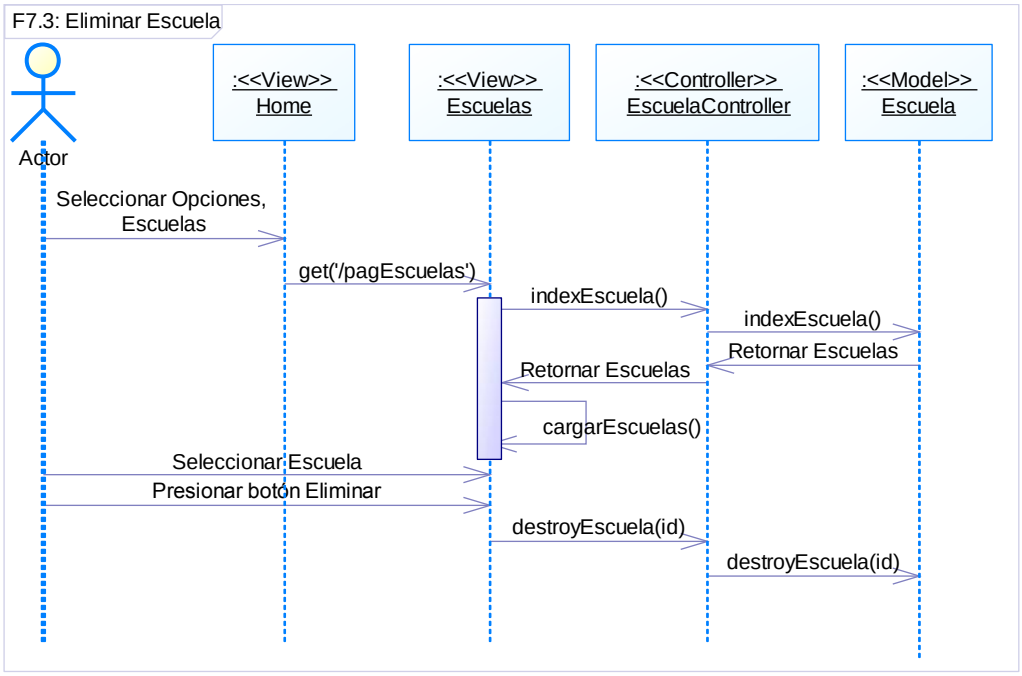


Figura 102. Diagrama Secuencia F7.3: Eliminar Escuela

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Escuelas
2. El sistema despliega la ventana de Escuelas

3. El sistema presenta dentro de la ventana una tabla con todos los registros de Escuelas ingresados previamente
4. El actor presiona el botón de basurero (Eliminar) en el registro que desee eliminar
5. El sistema elimina el registro seleccionado

F7.4: Consultar Escuela General

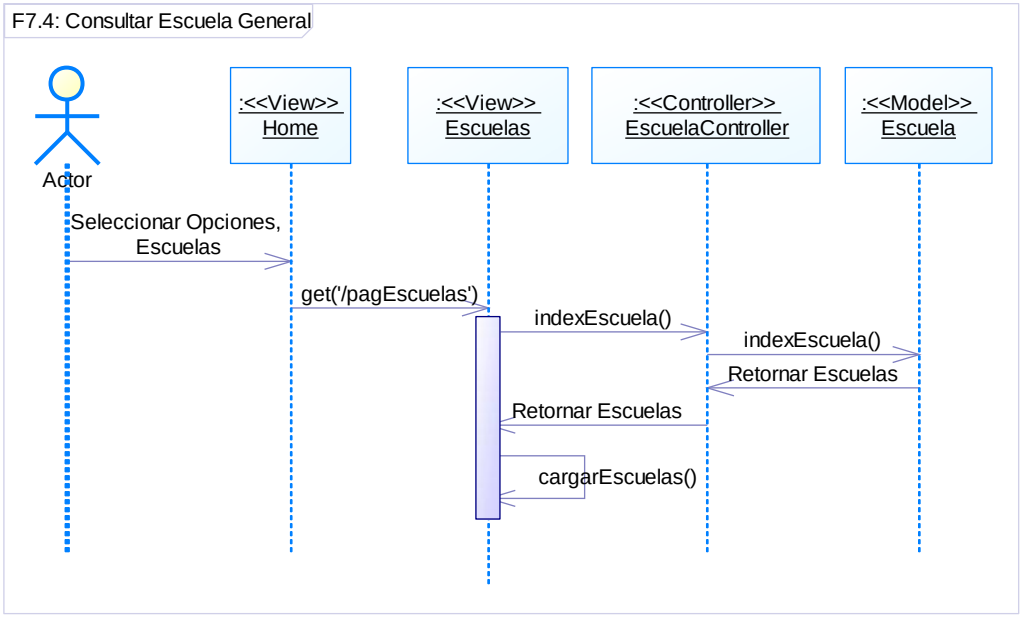


Figura 103. Diagrama Secuencia F7.4: Consultar Escuela General

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Escuelas
2. El sistema despliega la ventana de Escuelas
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Escuelas ingresados previamente

F7.4.1: Consultar Escuela por Parámetros

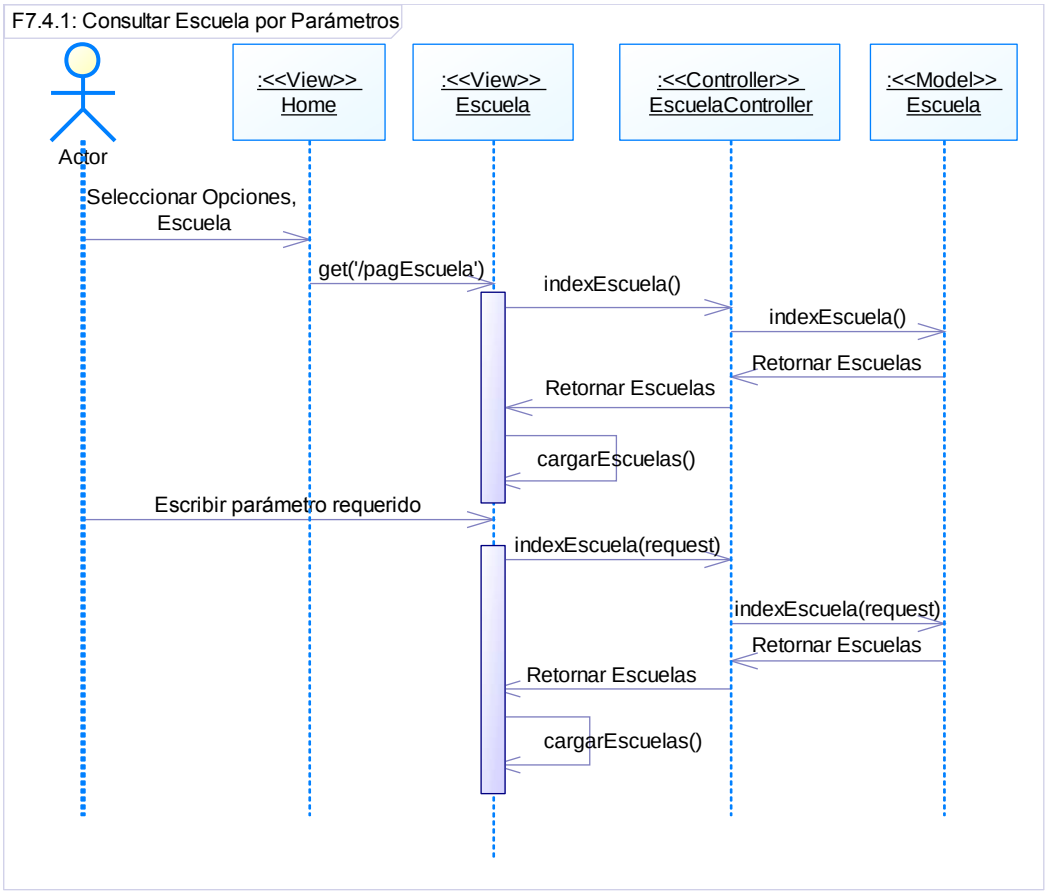


Figura 104. Diagrama Secuencia F7.4.1: Consultar Escuela por Parámetros

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Escuelas
2. El sistema despliega la ventana de Escuelas
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Escuelas ingresados previamente
4. El actor escribe la palabra que desea en la barra de búsqueda
5. El sistema presenta dentro de la tabla del punto 3 todos los registros que cumplen con el parámetro ingresado por el actor

F8: Gestionar Carrera

F8.1: Ingresar Carrera

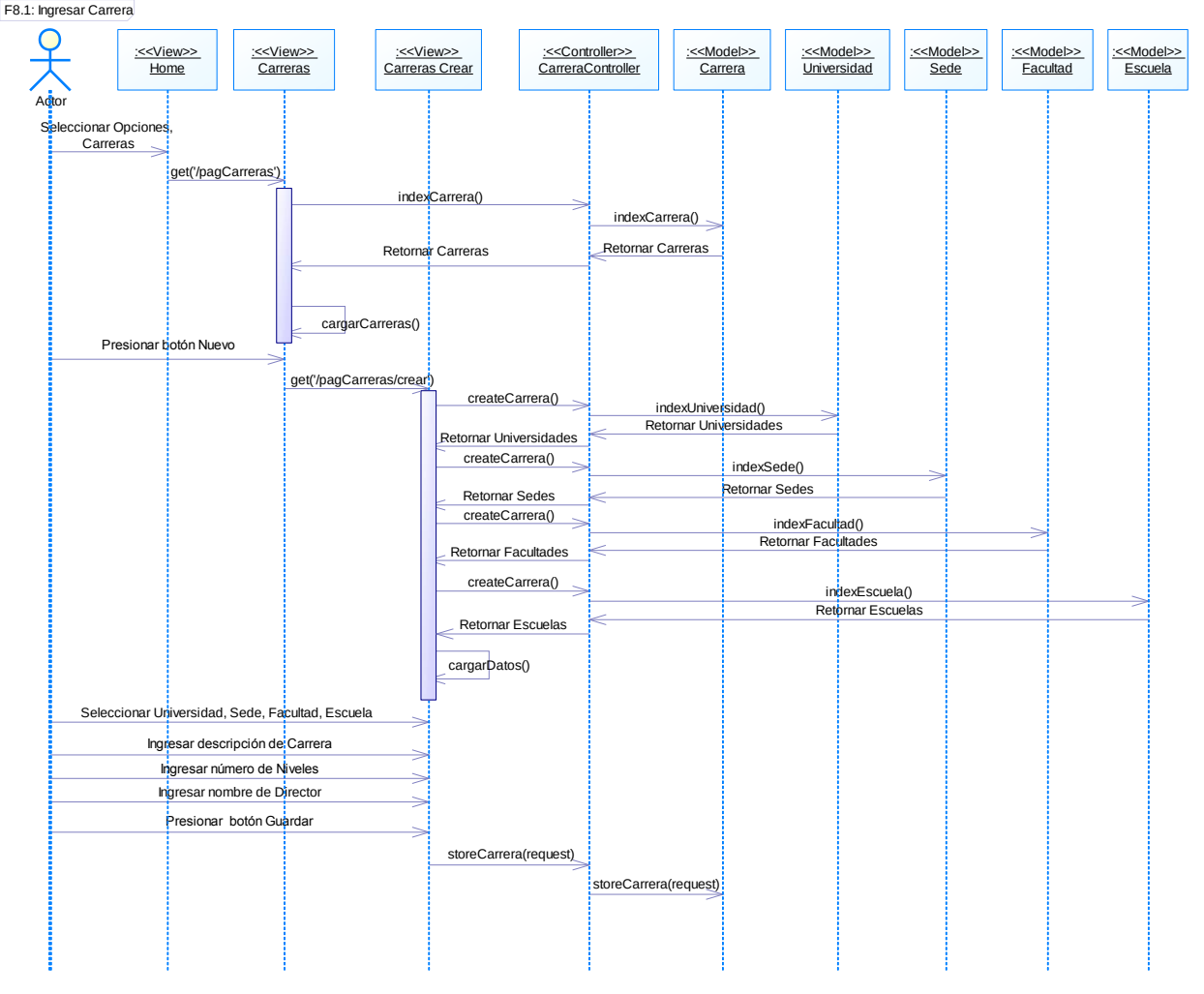


Figura 105. Diagrama Secuencia F8.1: Ingresar Carrera

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Carreras
2. El sistema despliega la ventana de Carreras

3. El sistema presenta dentro de la ventana una tabla con todos los registros de Carreras ingresados previamente
4. El actor presiona el botón Nuevo
5. El sistema dentro de la ventana Carreras Crear, despliega Universidad, Sede, Facultad, Escuela y permite ingresar la Descripción de la Carrera, el número de Niveles y el Director.
6. El actor selecciona la Universidad, Sede, Facultad, Escuela e ingresa Descripción de la Carrera, el número de niveles y el Director.
7. El actor presiona el botón Guardar
8. El sistema almacena los datos

F8.2: Modificar Carrera

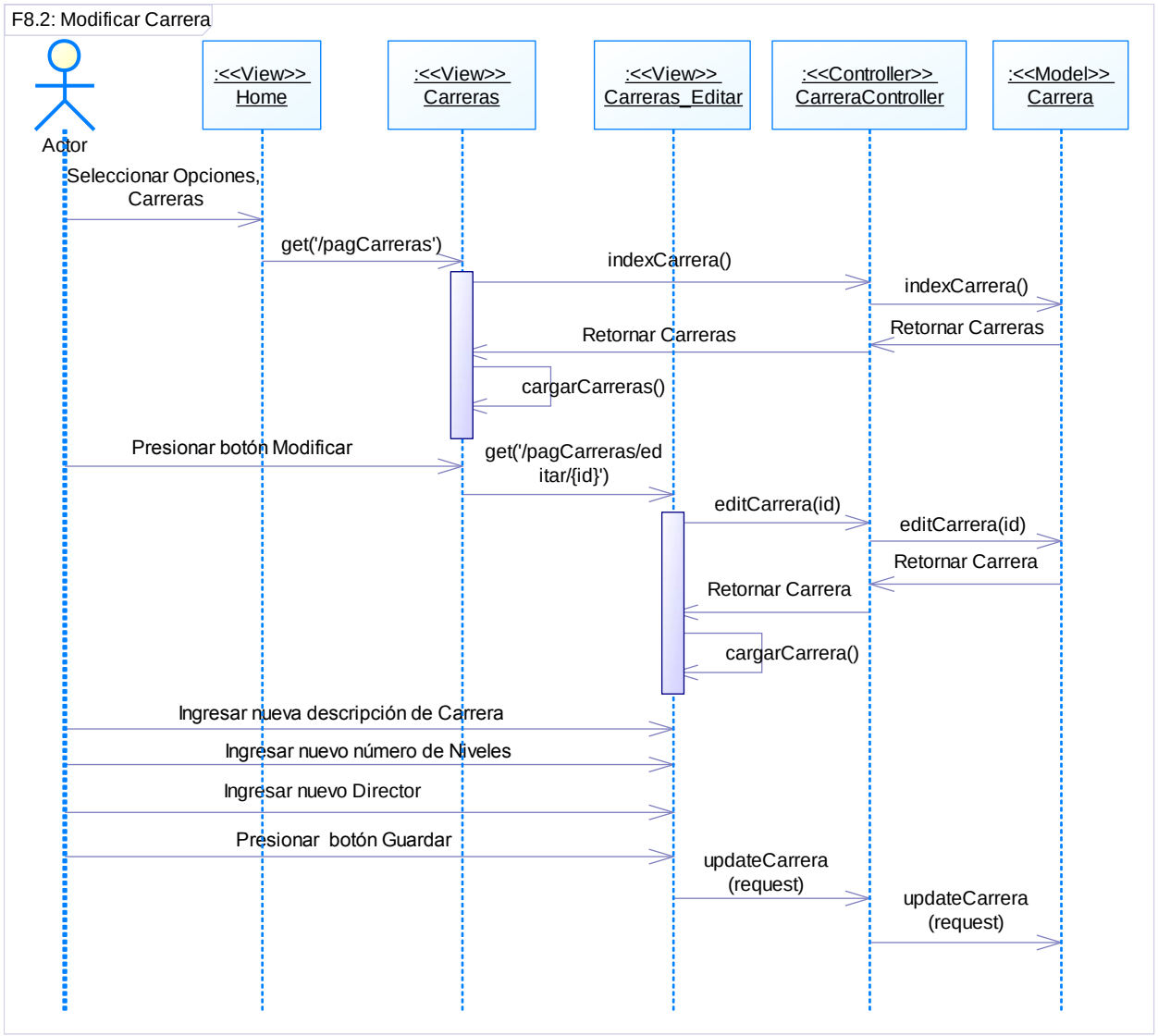


Figura 106. Diagrama Secuencia F8.2: Modificar Carrera

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Carreras
2. El sistema despliega la ventana de Carreras
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Escuelas ingresados previamente
4. El actor presiona el botón de lápiz (Modificar) en el registro que desee modificar
5. El sistema presenta la ventana Escuelas Editar con todos los campos llenos
6. El actor modifica los datos en los campos que desee
7. El actor presiona el botón Guardar
8. El sistema modifica los datos

F8.3: Eliminar Carrera

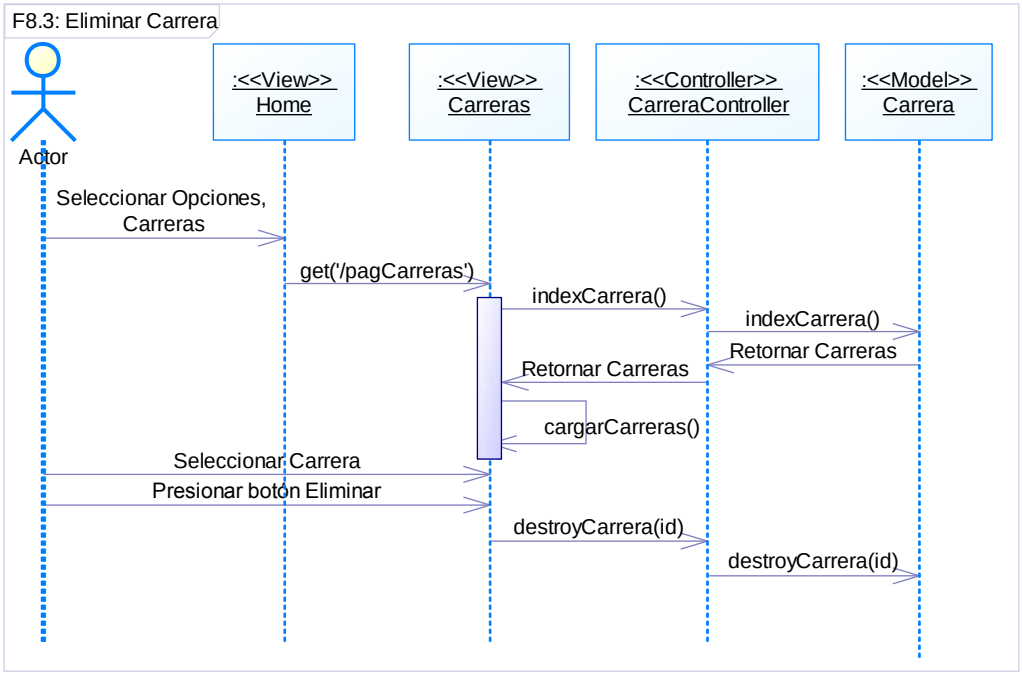


Figura 107. Diagrama Secuencia F8.3: Eliminar Carrera

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Carreras
2. El sistema despliega la ventana de Carreras
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Carreras ingresados previamente
4. El actor presiona el botón de basurero (Eliminar) en el registro que desee eliminar
5. El sistema elimina el registro seleccionado

F8.4: Consultar Carrera General

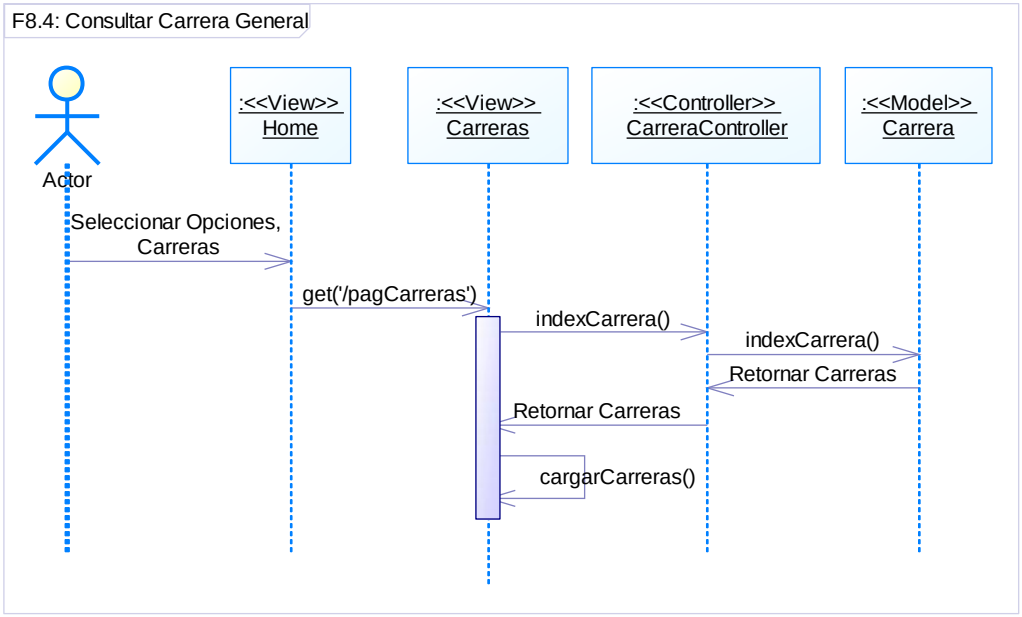


Figura 108. Diagrama Secuencia F8.4: Consultar Carrera General

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Carreras
2. El sistema despliega la ventana de Carreras
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Carreras ingresados previamente

F8.4.1: Consultar Carrera por Parámetros

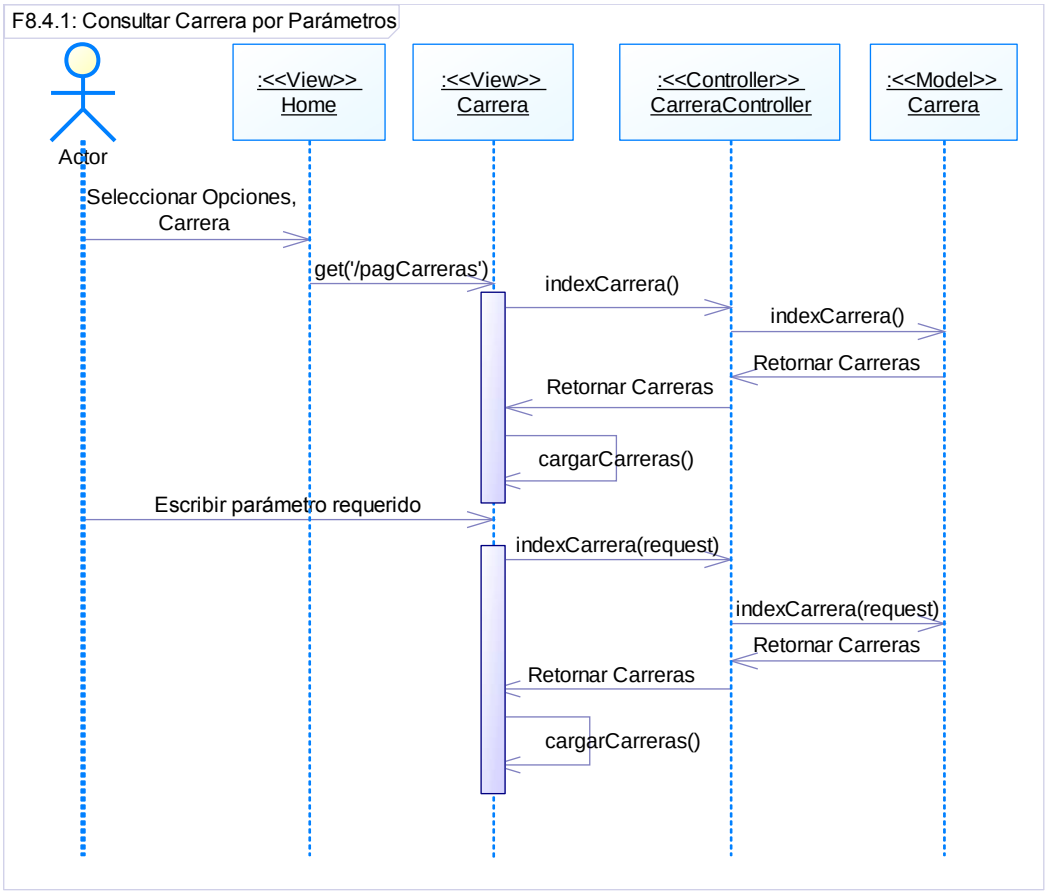


Figura 109. Diagrama Secuencia F8.4.1: Consultar Carrera por Parámetros

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Carreras
2. El sistema despliega la ventana de Carreras
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Carreras ingresados previamente
4. El actor escribe la palabra que desea en la barra de búsqueda
5. El sistema presenta dentro de la tabla del punto 3 todos los registros que cumplen con el parámetro ingresado por el actor

F9: Gestionar Estudiante

F9.1: Ingresar Estudiante

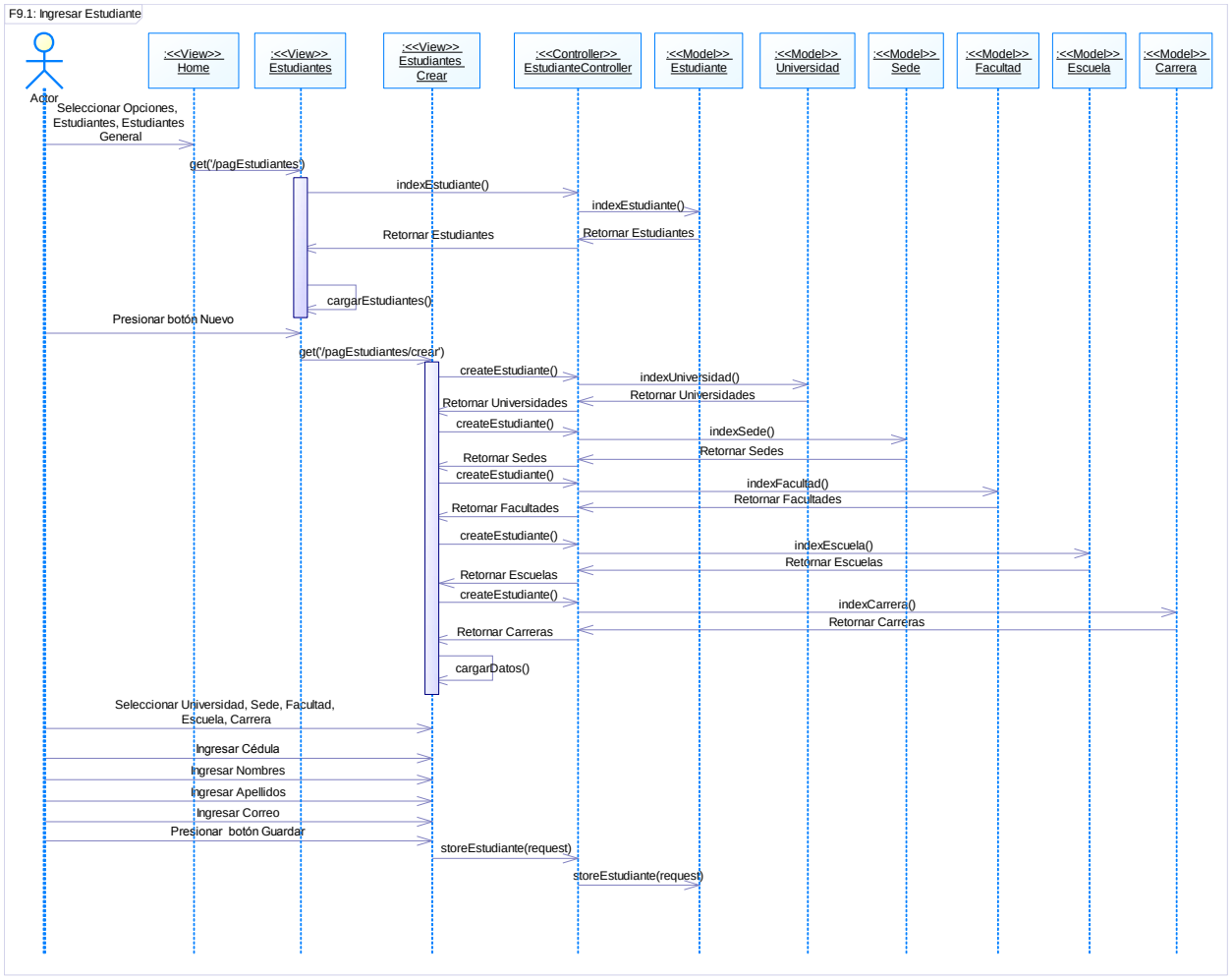


Figura 110. Diagrama Secuencia F9.1: Ingresar Estudiante

Actores

1. Administrador
2. Profesor

Flujo Principal

1. El actor selecciona Opciones, Estudiantes y en Estudiantes General
2. El sistema despliega la ventana de Estudiantes

3. El sistema presenta dentro de la ventana una tabla con todos los registros de Estudiantes ingresados previamente
4. El actor presiona el botón Nuevo
5. El sistema dentro de la ventana Estudiantes Crear, despliega Universidad, Sede, Facultad, Escuela, Carrera y permite ingresar la Cédula, los Nombres, Apellidos y el Correo
6. El actor selecciona la Universidad, Sede, Facultad, Escuela, Carrera e ingresa la Cédula, los Nombres, Apellidos y el Correo
7. El actor presiona el botón Guardar
8. El sistema almacena los datos

F9.2: Modificar Estudiante

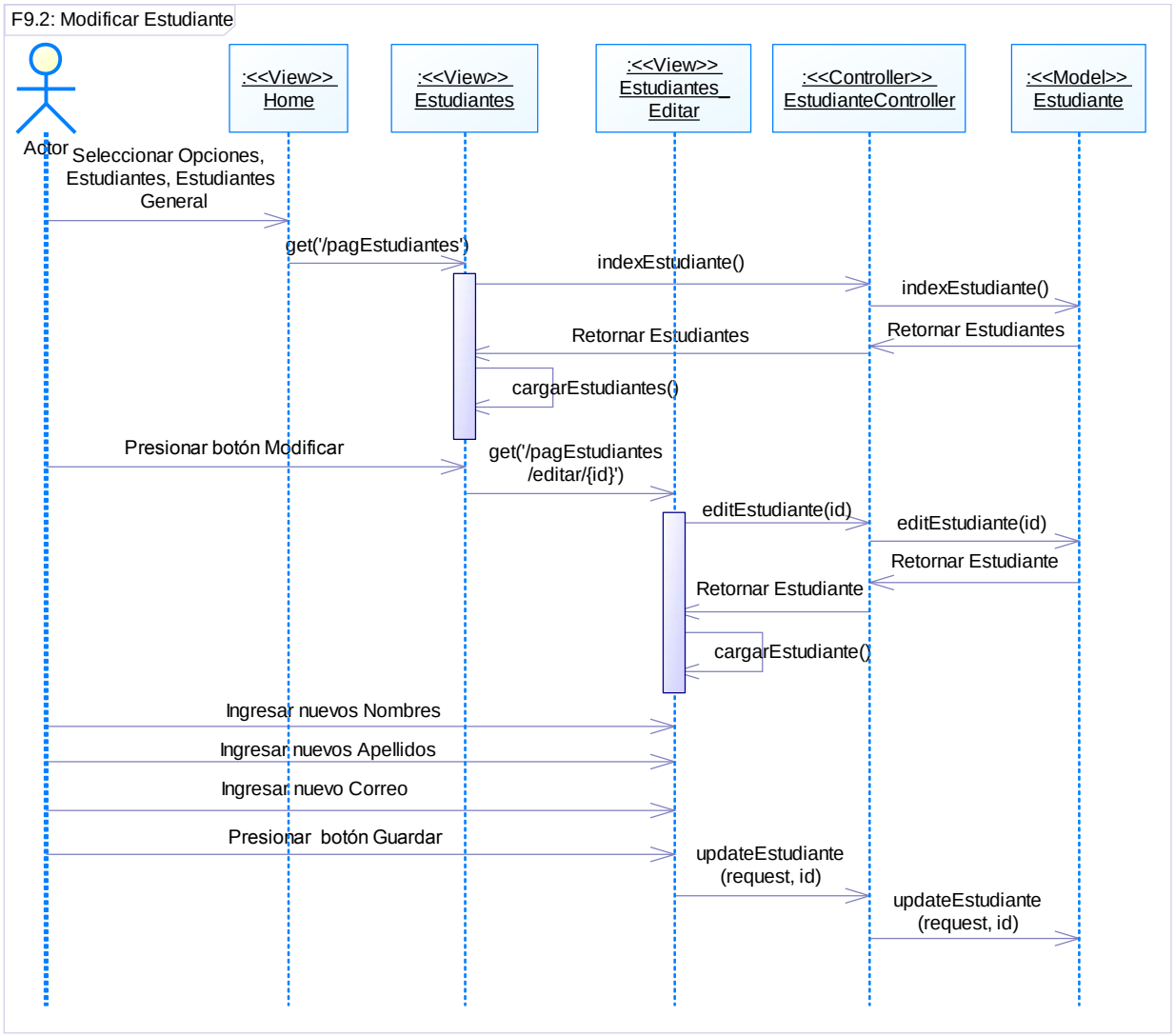


Figura 111. Diagrama Secuencia F9.2: Modificar Estudiante

Actores

1. Administrador
2. Profesor

Flujo Principal

1. El actor selecciona Opciones, Estudiantes y en Estudiantes General
2. El sistema despliega la ventana de Estudiantes
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Estudiantes ingresados previamente
4. El actor presiona el botón de lápiz (Modificar) en el registro que desee modificar
5. El sistema presenta la ventana Estudiantes Editar con todos los campos llenos
6. El actor modifica los datos en los campos que desee
7. El actor presiona el botón Guardar
8. El sistema modifica los datos

F9.3: Eliminar Estudiante

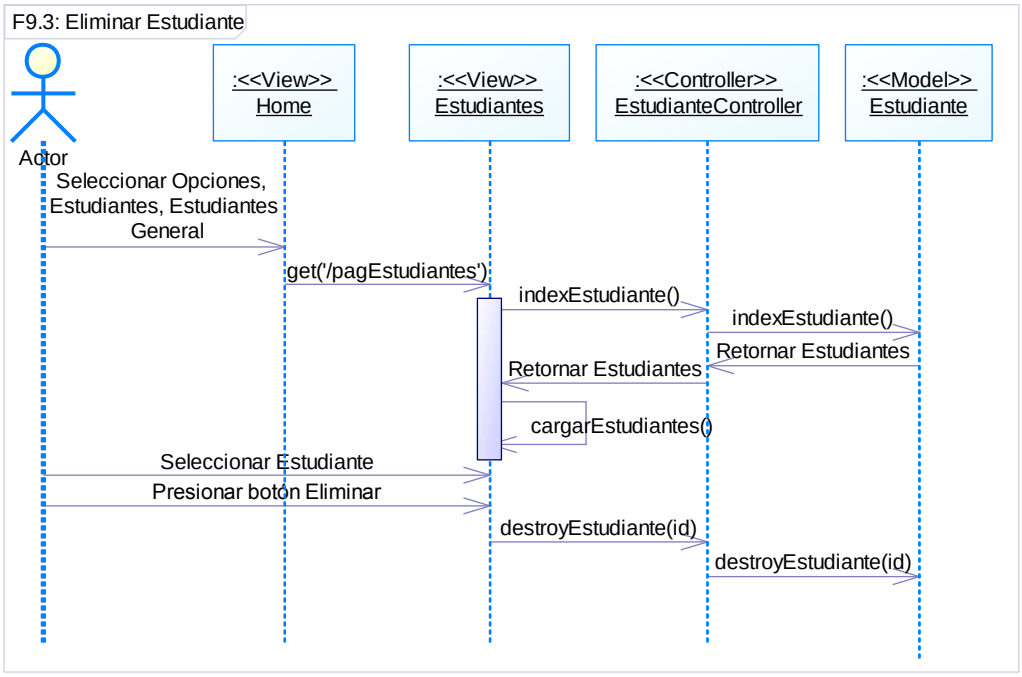


Figura 112. Diagrama Secuencia F9.3: Eliminar Estudiante

Actores

1. Administrador
2. Profesor

Flujo Principal

1. El actor selecciona Opciones, Estudiantes y en Estudiantes General
2. El sistema despliega la ventana de Estudiantes
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Estudiantes ingresados previamente
4. El actor presiona el botón de basurero (Eliminar) en el registro que desee eliminar
5. El sistema elimina el registro seleccionado

F9.4: Consultar Estudiante General

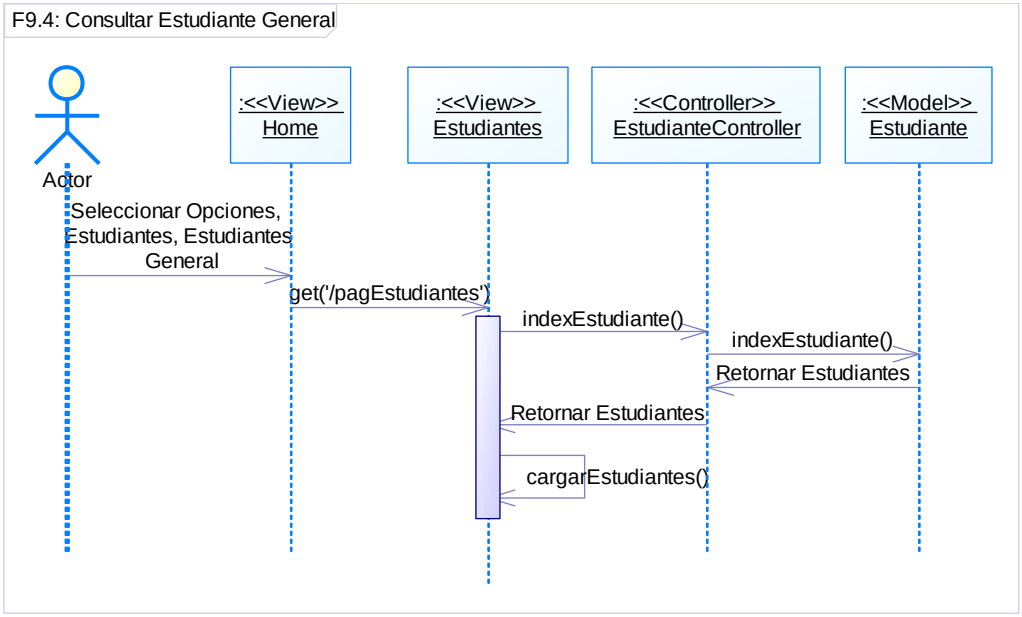


Figura 113. Diagrama Secuencia F9.4: Consultar Estudiante General

Actores

1. Administrador
2. Profesor

Flujo Principal

1. El actor selecciona Opciones, Estudiantes
2. El sistema despliega la ventana de Estudiantes

3. El sistema presenta dentro de la ventana una tabla con todos los registros de Estudiantes ingresados previamente

F9.4.1: Consultar Estudiante por Parámetros

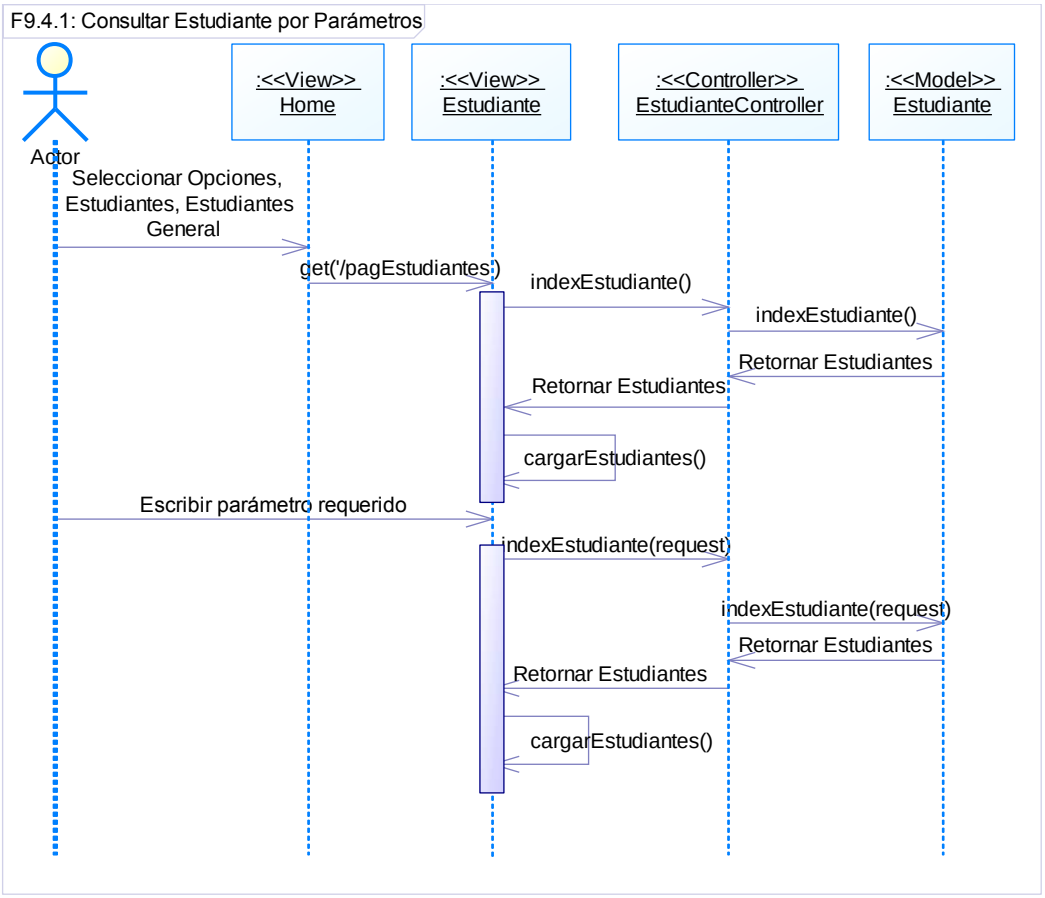


Figura 114. Diagrama Secuencia F9.4: Consultar Estudiante General

Actores

1. Administrador
2. Profesor
3. Estudiante

Flujo Principal

1. El actor selecciona Opciones, Estudiantes
2. El sistema despliega la ventana de Estudiantes

3. El sistema presenta dentro de la ventana una tabla con todos los registros de Estudiantes ingresados previamente
4. El actor escribe la palabra que desea en la barra de búsqueda
5. El sistema presenta dentro de la tabla del punto 3 todos los registros que cumplen con el parámetro ingresado por el actor

F10: Gestionar Profesor

F10.1: Ingresar Profesor

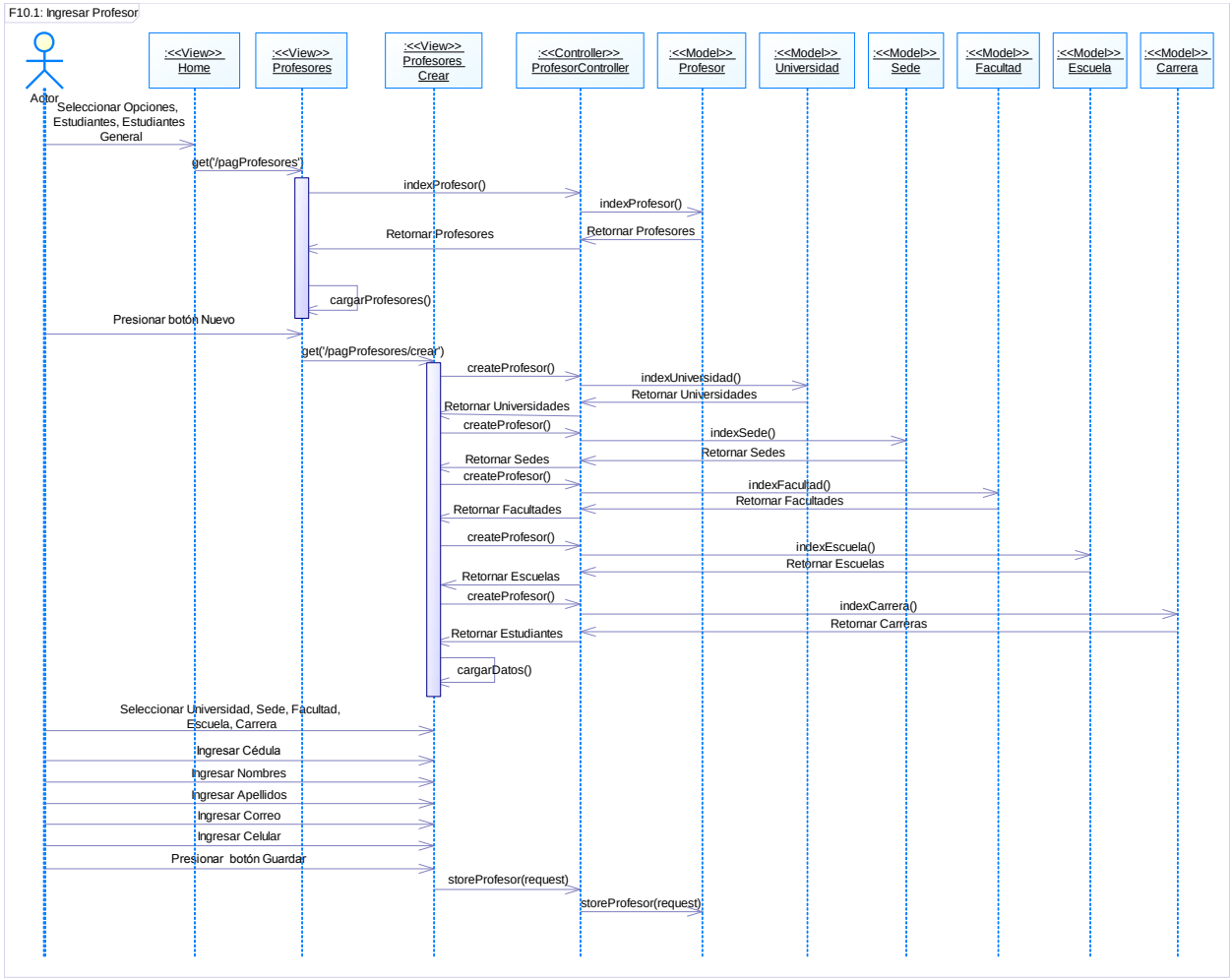


Figura 115. Diagrama Secuencia F10.1: Ingresar Profesor

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Profesores
2. El sistema despliega la ventana de Profesores
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Profesores ingresados previamente
4. El actor presiona el botón Nuevo
5. El sistema dentro de la ventana Profesores Crear, despliega Universidad, Sede, Facultad, Escuela, Carrera y permite ingresar la Cédula, los Nombres, Apellidos, el Correo y el Celular
6. El actor selecciona la Universidad, Sede, Facultad, Escuela, Carrera e ingresa la Cédula, los Nombres, Apellidos, el Correo y el Celular
7. El actor presiona el botón Guardar
8. El sistema almacena los datos

F10.2: Modificar Profesor

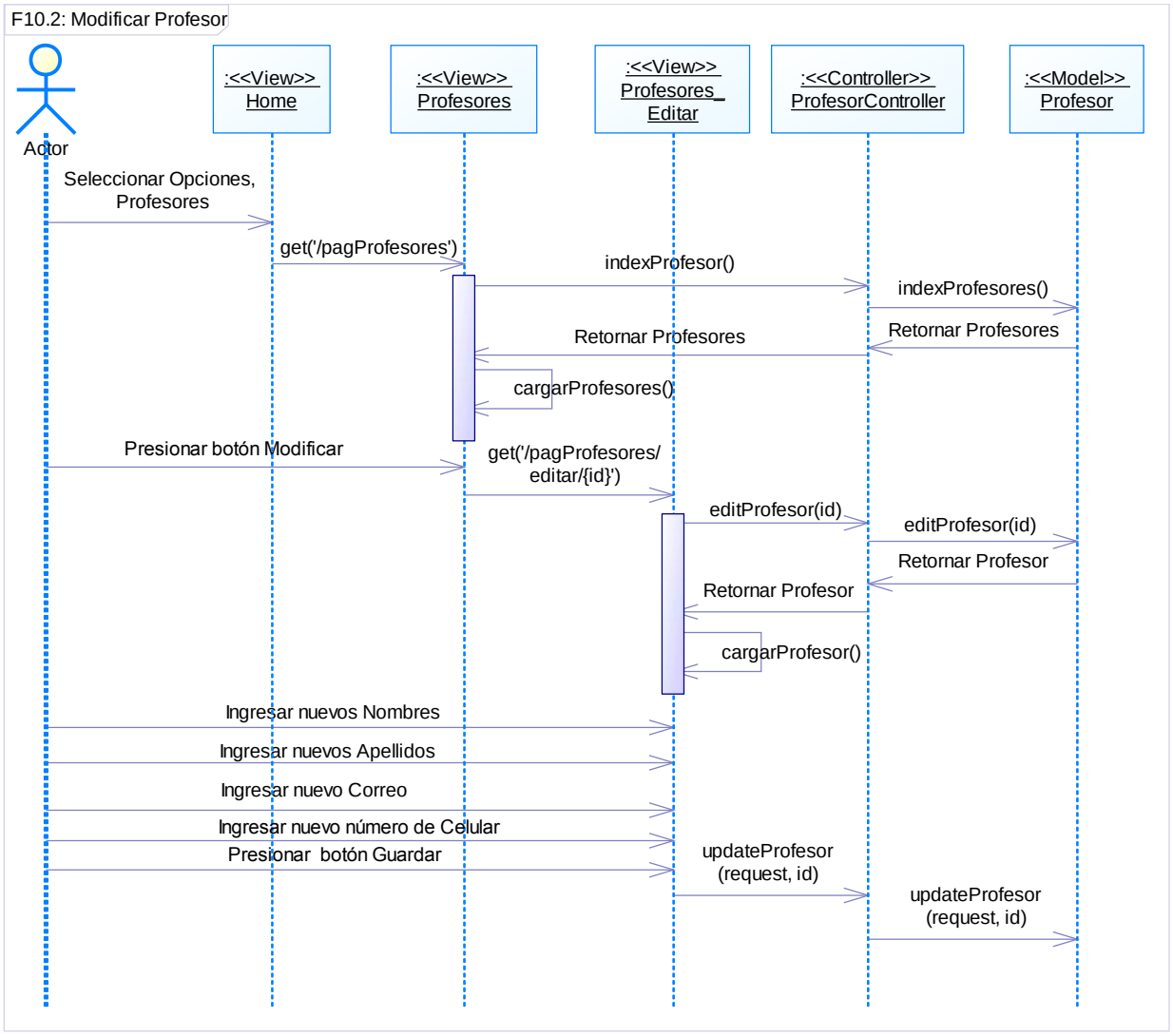


Figura 116. Diagrama Secuencia F10.2: Modificar Profesor

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Profesores
2. El sistema despliega la ventana de Profesores

3. El sistema presenta dentro de la ventana una tabla con todos los registros de Profesores ingresados previamente
4. El actor presiona el botón de lápiz (Modificar) en el registro que desee modificar
5. El sistema presenta la ventana Profesores Editar con todos los campos llenos
6. El actor modifica los datos en los campos que desee
7. El actor presiona el botón Guardar
8. El sistema modifica los datos

F10.3: Eliminar Profesor

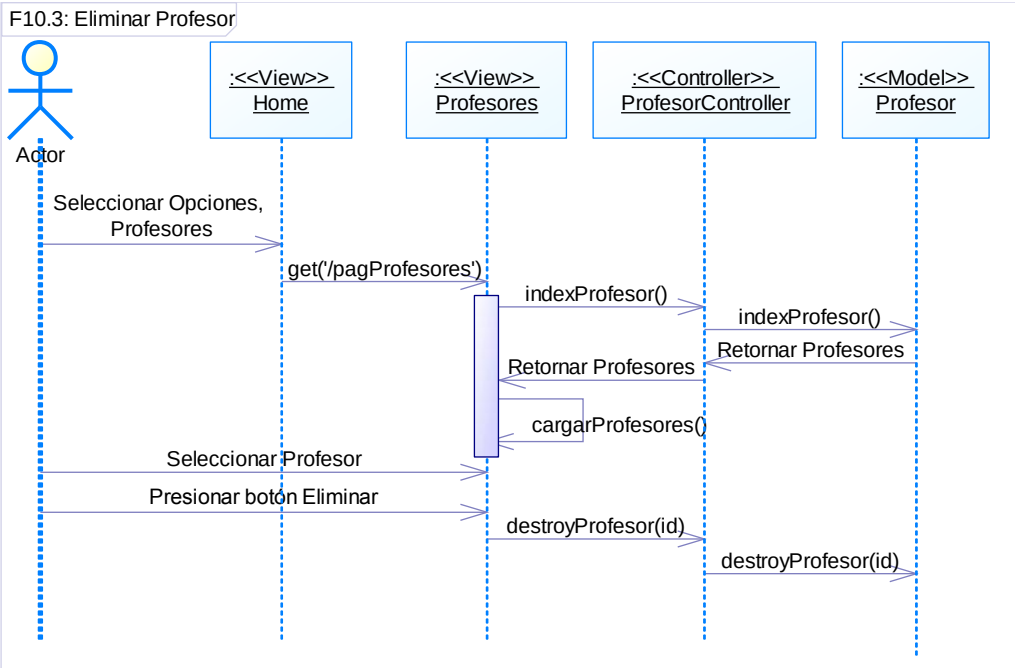


Figura 117. Diagrama Secuencia F10.3: Eliminar Profesor

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Profesores
2. El sistema despliega la ventana de Profesores
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Profesores ingresados previamente
4. El actor presiona el botón de basurero (Eliminar) en el registro que desee eliminar
5. El sistema elimina el registro seleccionado

F10.4: Consultar Profesor General

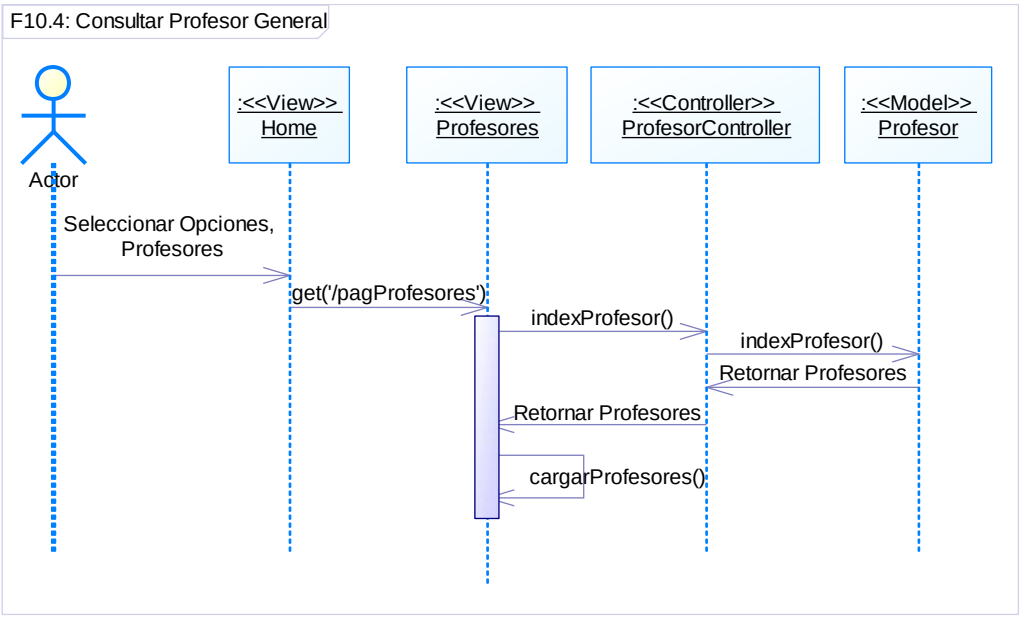


Figura 118. Diagrama Secuencia F10.4: Consultar Profesor General

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Profesores
2. El sistema despliega la ventana de Profesores
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Profesores ingresados previamente

F10.4.1: Consultar Profesor por Parámetros

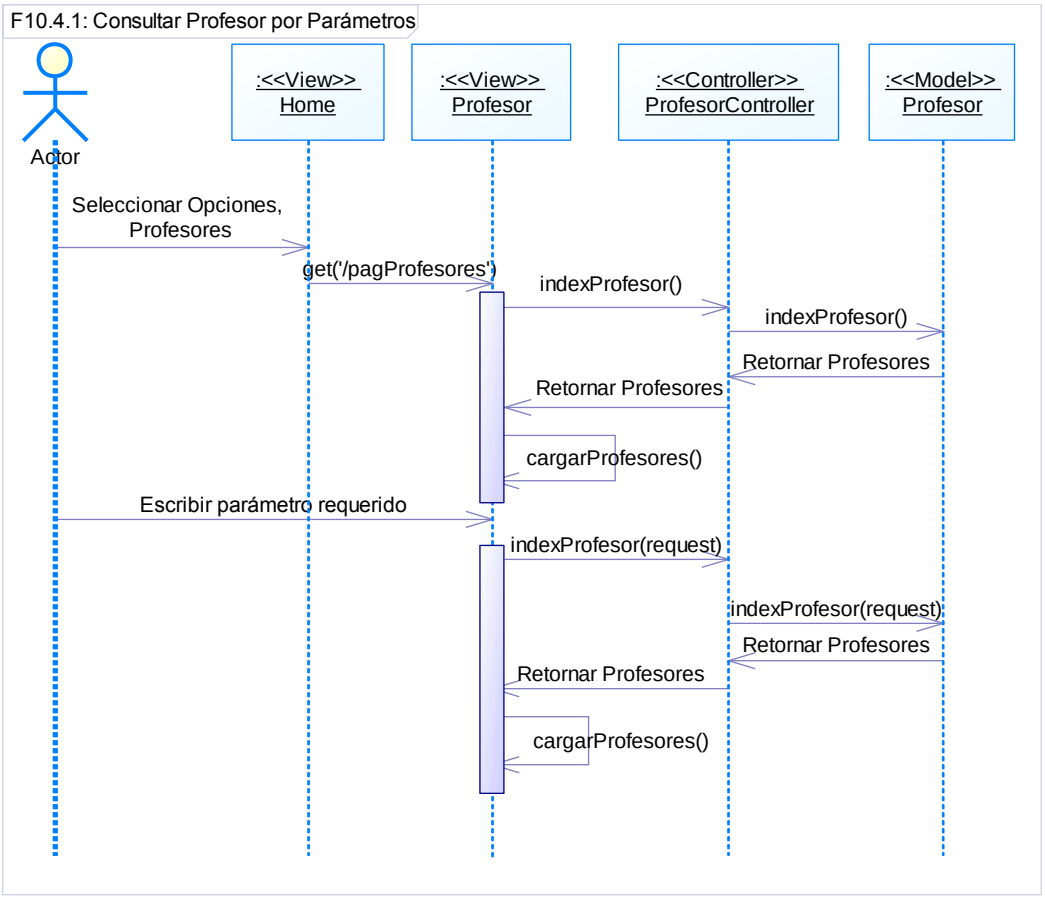


Figura 119. Diagrama Secuencia F10.4.1: Consultar Profesor por Parámetros

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Profesores
2. El sistema despliega la ventana de Profesores
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Profesores ingresados previamente
4. El actor escribe la palabra que desea en la barra de búsqueda
5. El sistema presenta dentro de la tabla del punto 3 todos los registros que cumplen con el parámetro ingresado por el actor

F11: Gestionar Asignatura

F11.1: Ingresar Asignatura

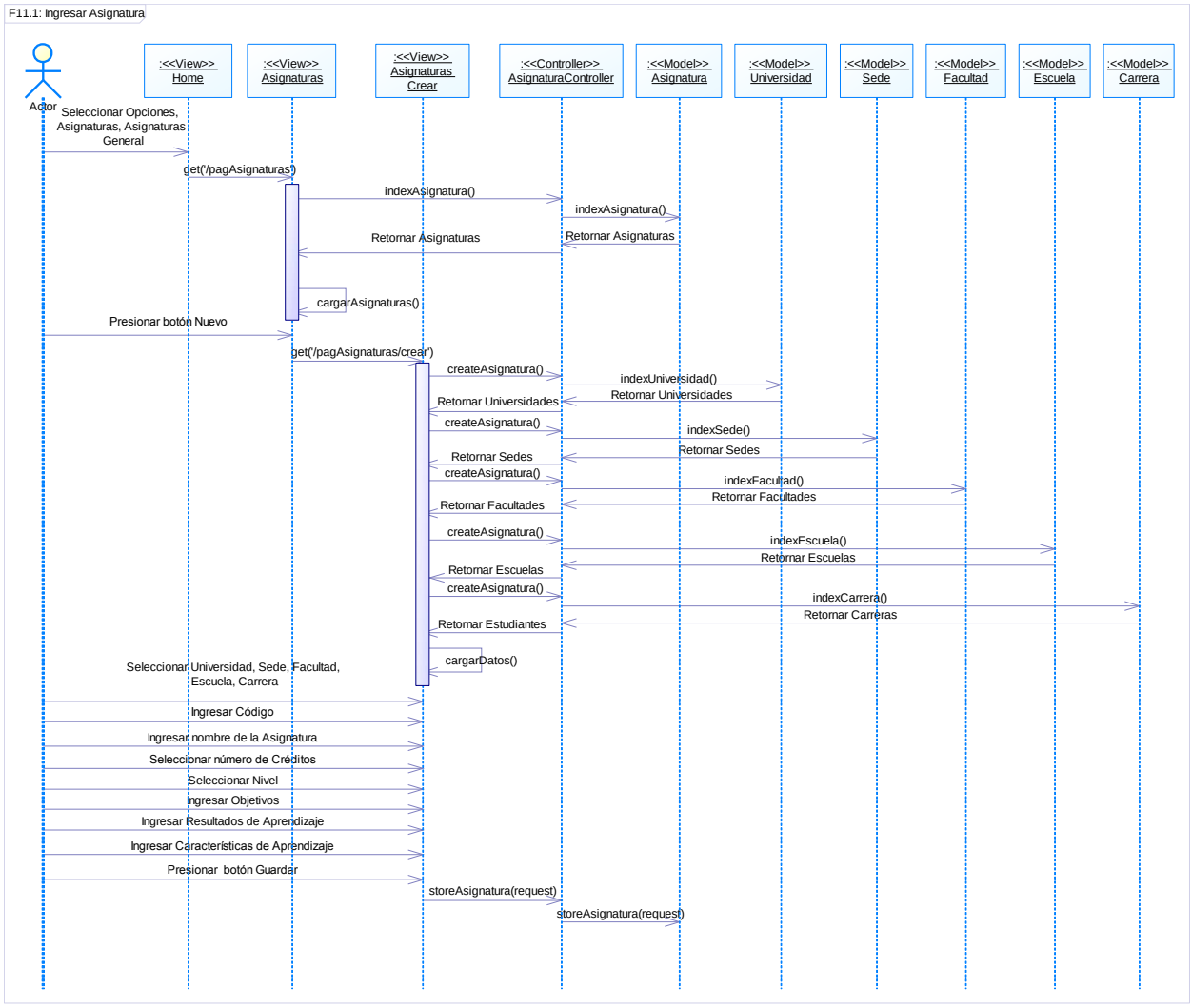


Figura 120. Diagrama Secuencia F11.1: Ingresar Asignatura

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Asignaturas y en Asignaturas General
2. El sistema despliega la ventana de Asignaturas

3. El sistema presenta dentro de la ventana una tabla con todos los registros de Asignaturas ingresados previamente
4. El actor presiona el botón Nuevo
5. El sistema dentro de la ventana Profesores Crear, despliega Universidad, Sede, Facultad, Escuela, Carrera y permite ingresar el código, el nombre de la Asignatura, el número de Créditos, permite seleccionar el Nivel en el que se dicta la Asignatura, ingresar los Resultados de Aprendizaje y las Características de Aprendizaje
6. El actor selecciona la Universidad, Sede, Facultad, Escuela, Carrera e ingresa el código, el nombre de la Asignatura, el número de Créditos, selecciona el Nivel en el que se dicta la Asignatura e ingresa los Resultados de Aprendizaje y las Características de Aprendizaje
7. El actor presiona el botón Guardar
8. El sistema almacena los datos

F11.2: Modificar Asignatura

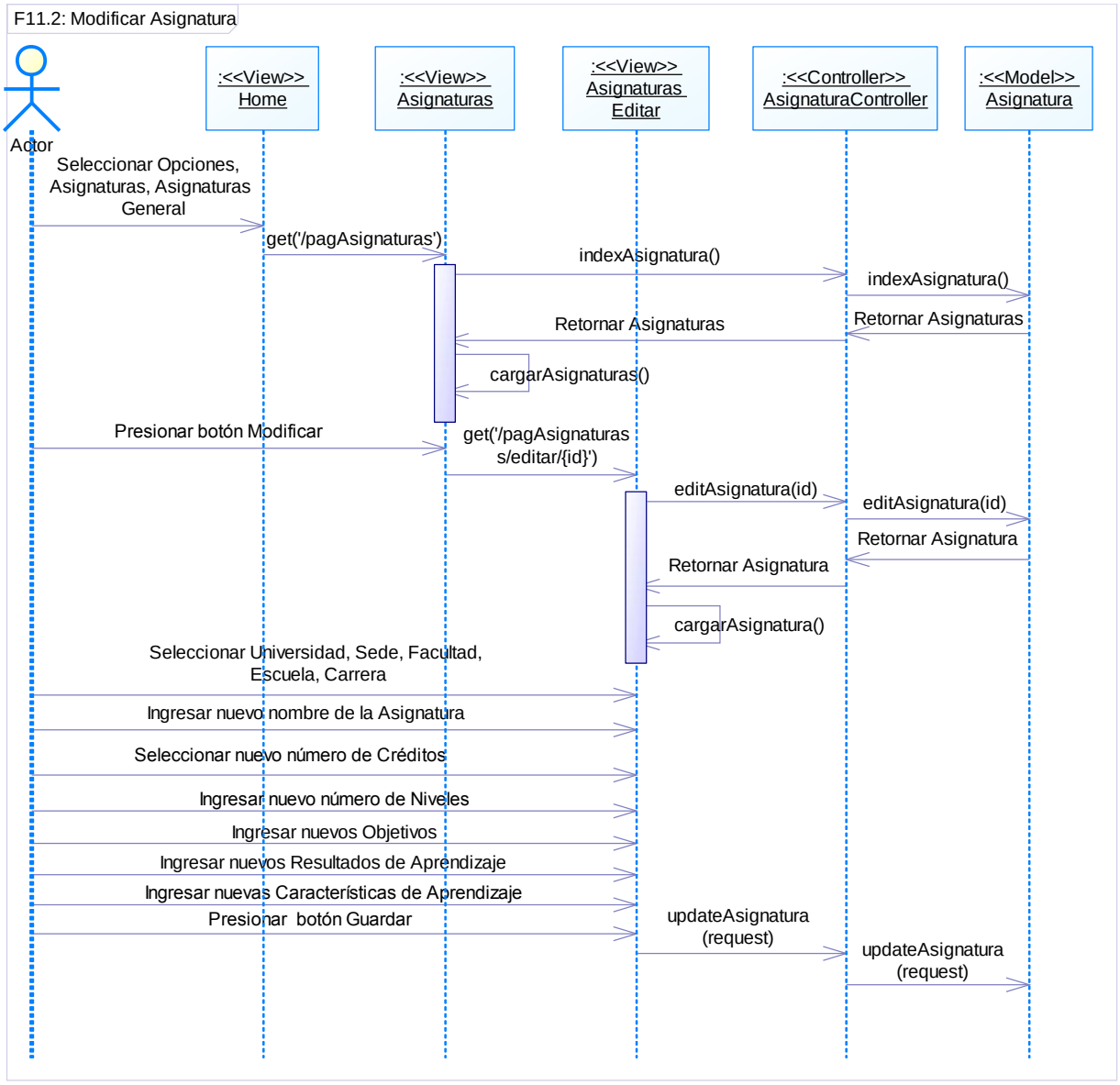


Figura 121. Diagrama Secuencia F11.2: Modificar Asignatura

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Asignaturas y en Asignaturas General

2. El sistema despliega la ventana de Asignaturas
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Asignaturas ingresados previamente
4. El actor presiona el botón de lápiz (Modificar) en el registro que desee modificar
5. El sistema presenta la ventana Asignaturas Editar con todos los campos llenos
6. El actor modifica los datos en los campos que desee
7. El actor presiona el botón Guardar
8. El sistema modifica los datos

F11.3: Eliminar Asignatura

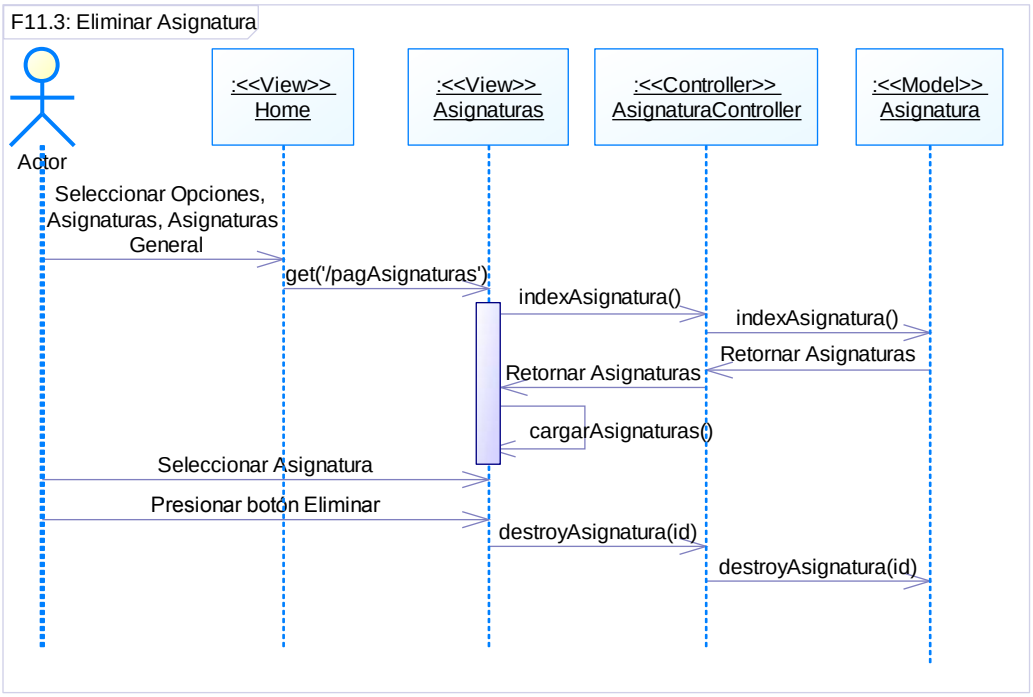


Figura 122. Diagrama Secuencia F11.3: Eliminar Asignatura

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Asignaturas y en Asignaturas General
2. El sistema despliega la ventana de Asignaturas

3. El sistema presenta dentro de la ventana una tabla con todos los registros de Asignaturas ingresados previamente
4. El actor presiona el botón de basurero (Eliminar) en el registro que desee eliminar
5. El sistema elimina el registro seleccionado

F11.4: Consultar Asignatura General

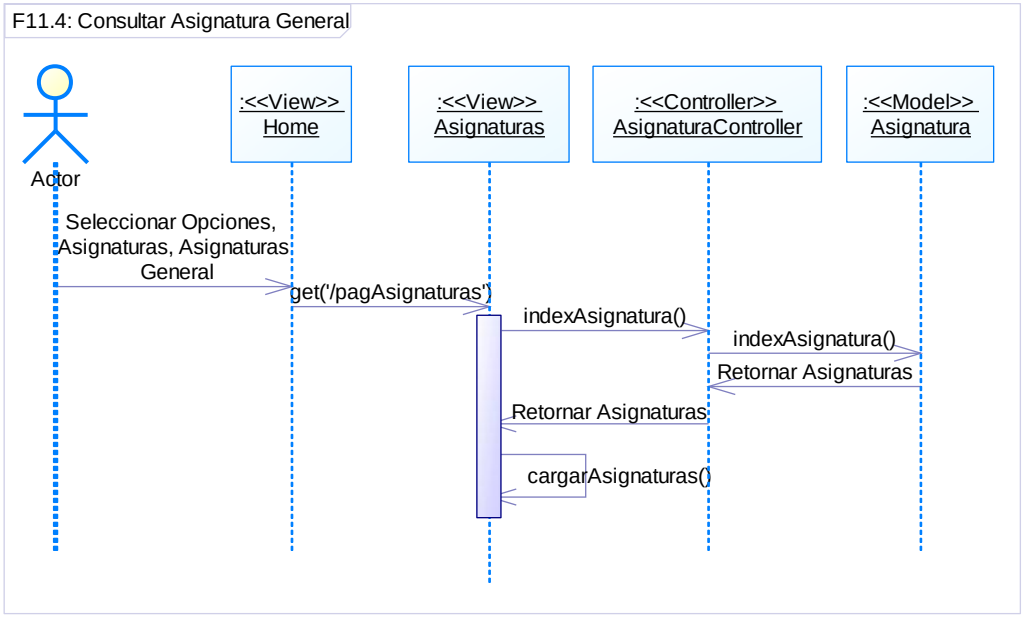


Figura 123. Diagrama Secuencia F11.4: Consultar Asignatura General

Actores

1. Administrador
2. Profesor

Flujo Principal

1. El actor selecciona Opciones, Asignaturas y en Asignaturas General
2. El sistema despliega la ventana de Asignaturas
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Asignaturas ingresados previamente

F11.4.1: Consultar Asignatura por Parámetros

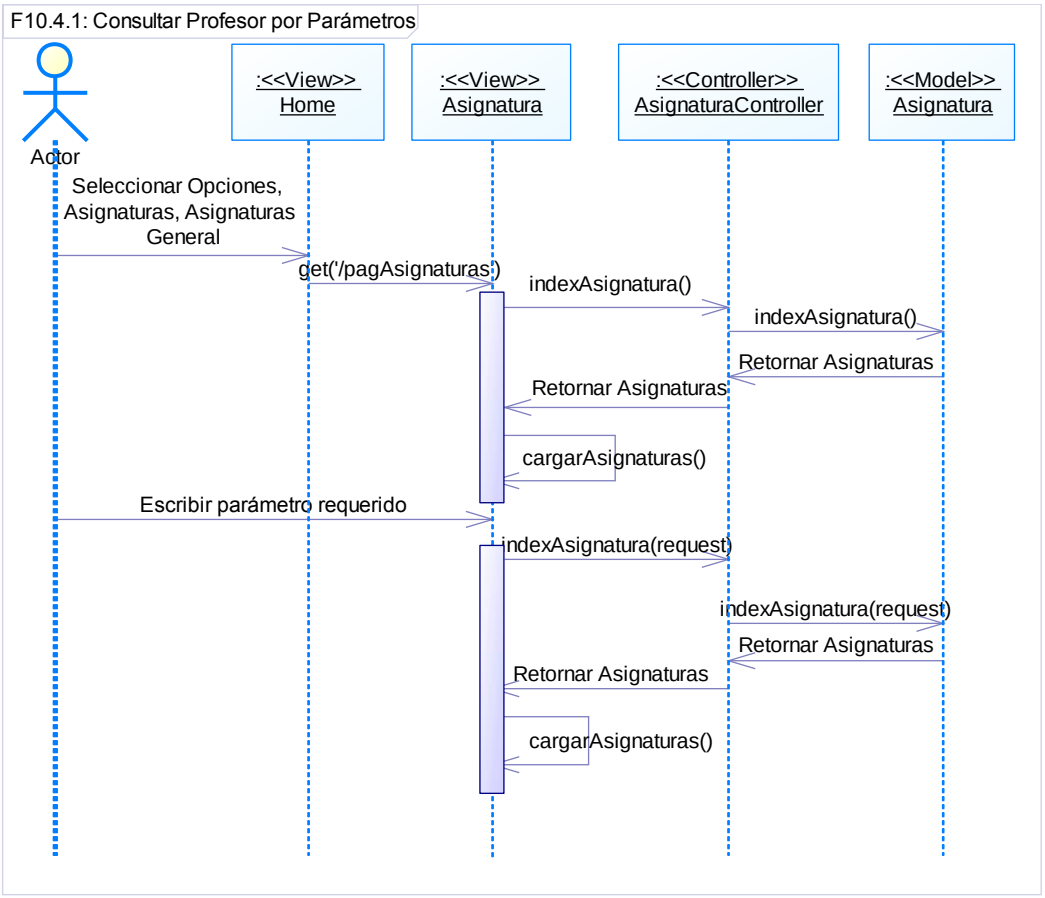


Figura 124. Diagrama Secuencia F11.4.1: Consultar Asignatura por Parámetros

Actores

1. Administrador
2. Profesor

Flujo Principal

1. El actor selecciona Opciones, Asignaturas y en Asignaturas General
2. El sistema despliega la ventana de Asignaturas
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Asignaturas ingresados previamente
4. El actor escribe la palabra que desea en la barra de búsqueda
5. El sistema presenta dentro de la tabla del punto 3 todos los registros que cumplen con el parámetro ingresado por el actor

F12: Gestionar Unidad de Estudio

F12.1: Ingresar Unidad de Estudio

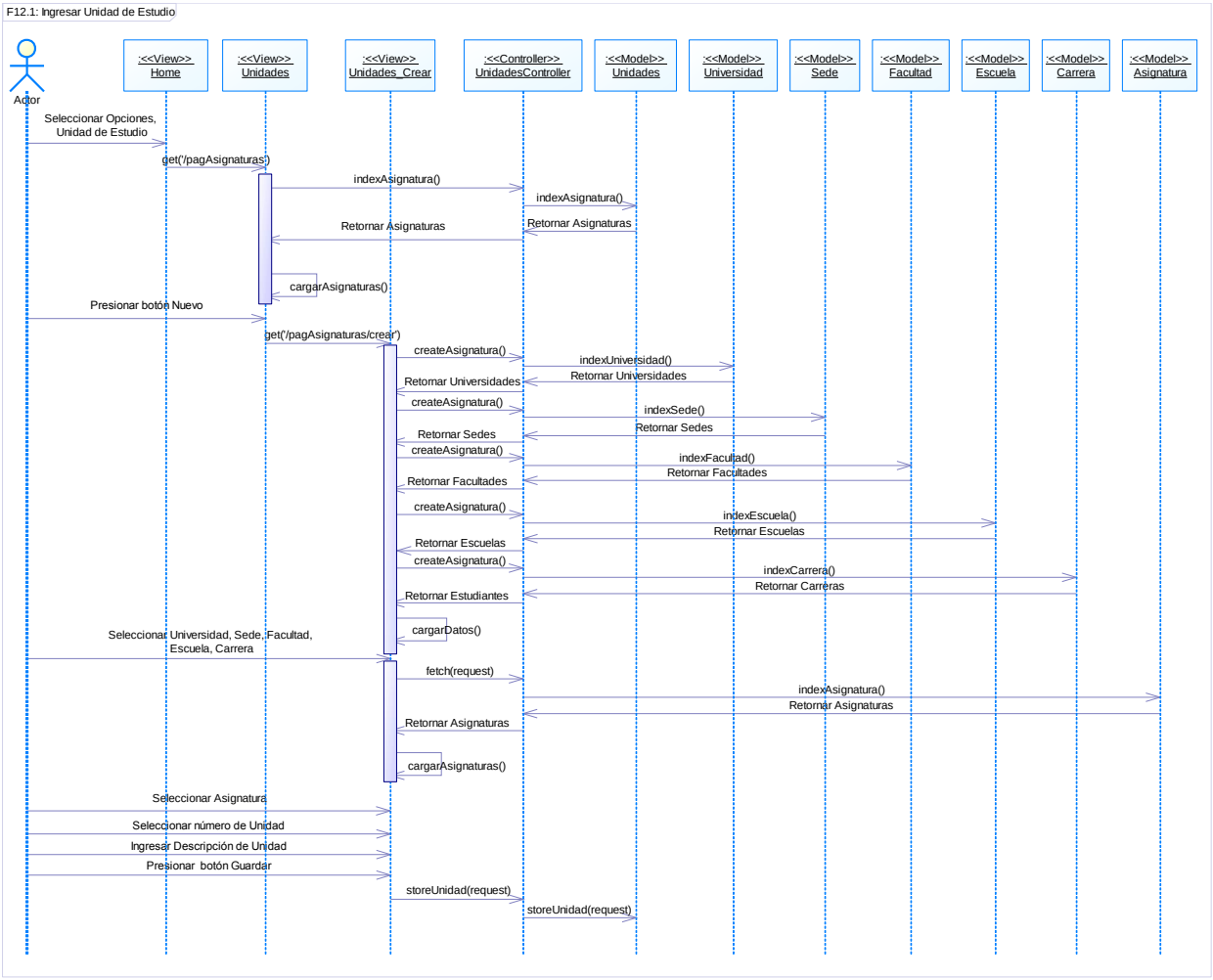


Figura 125. Diagrama Secuencia F12.1: Ingresar Unidad de Estudio

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Unidades
2. El sistema despliega la ventana de Unidades

3. El sistema presenta dentro de la ventana una tabla con todos los registros de Unidades ingresados previamente
4. El actor presiona el botón Nuevo
5. El sistema dentro de la ventana Unidades Crear, despliega Universidad, Sede, Facultad, Escuela, Carrera
6. El usuario selecciona la Universidad, Sede, Facultad, Escuela, Carrera que desea
7. El sistema carga las Asignaturas en base a la Universidad, Sede, Facultad, Escuela, Carrera previamente seleccionada
8. El usuario elige la Asignatura que desea, selecciona el número de Unidad, ingresa la descripción de la Unidad
9. El actor presiona el botón Guardar
10. El sistema almacena los datos

F12.2: Modificar Unidad de Estudio

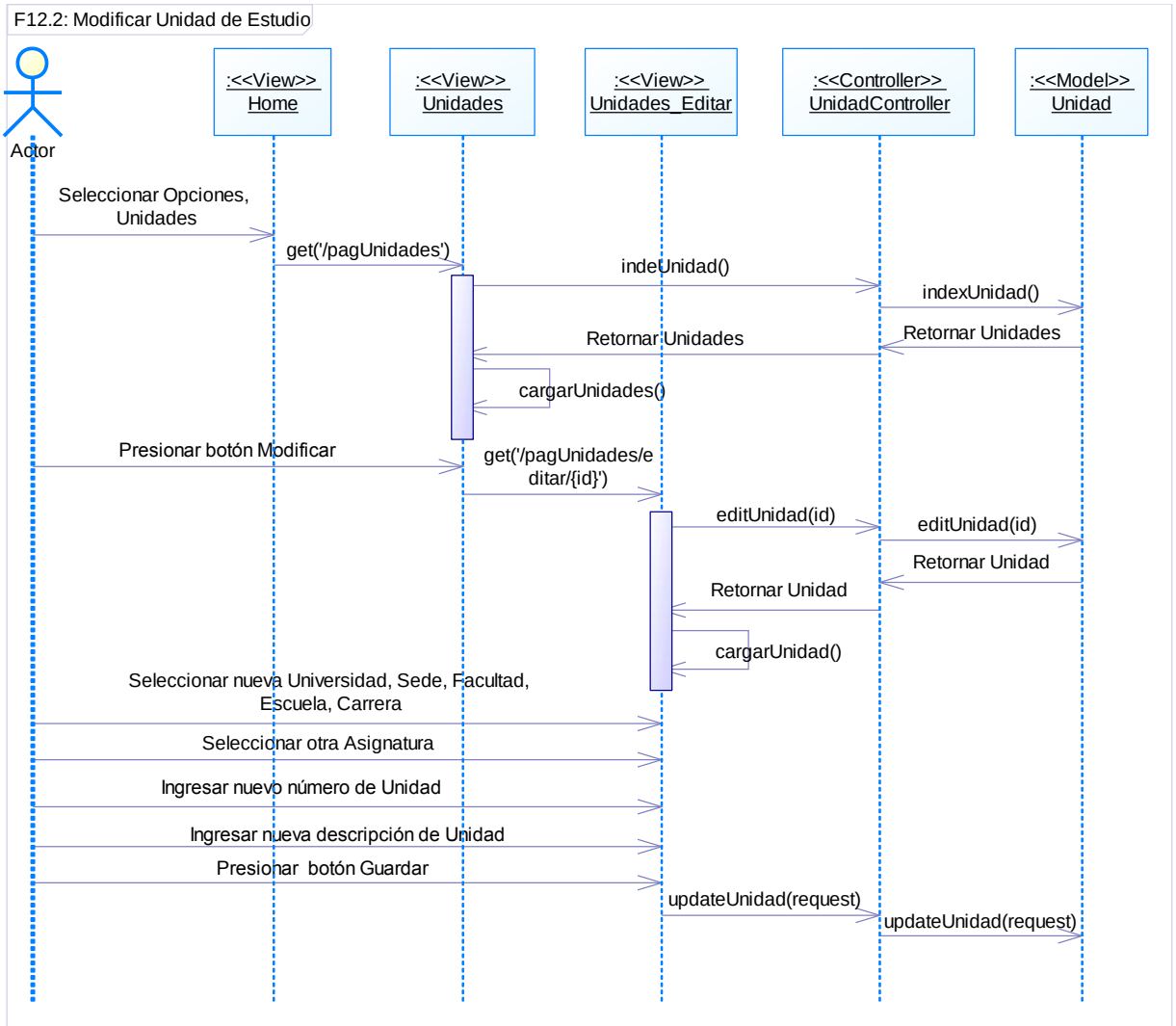


Figura 126. Diagrama Secuencia F12.2: Modificar Unidad de Estudio

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Unidades
2. El sistema despliega la ventana de Unidades
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Unidades ingresados previamente
4. El actor presiona el botón de lápiz (Modificar) en el registro que desee modificar
5. El sistema presenta la ventana Unidades Editar con todos los campos llenos
6. El actor modifica los datos en los campos que desee
7. El actor presiona el botón Guardar
8. El sistema modifica los datos

F12.3: Eliminar Unidad de Estudio

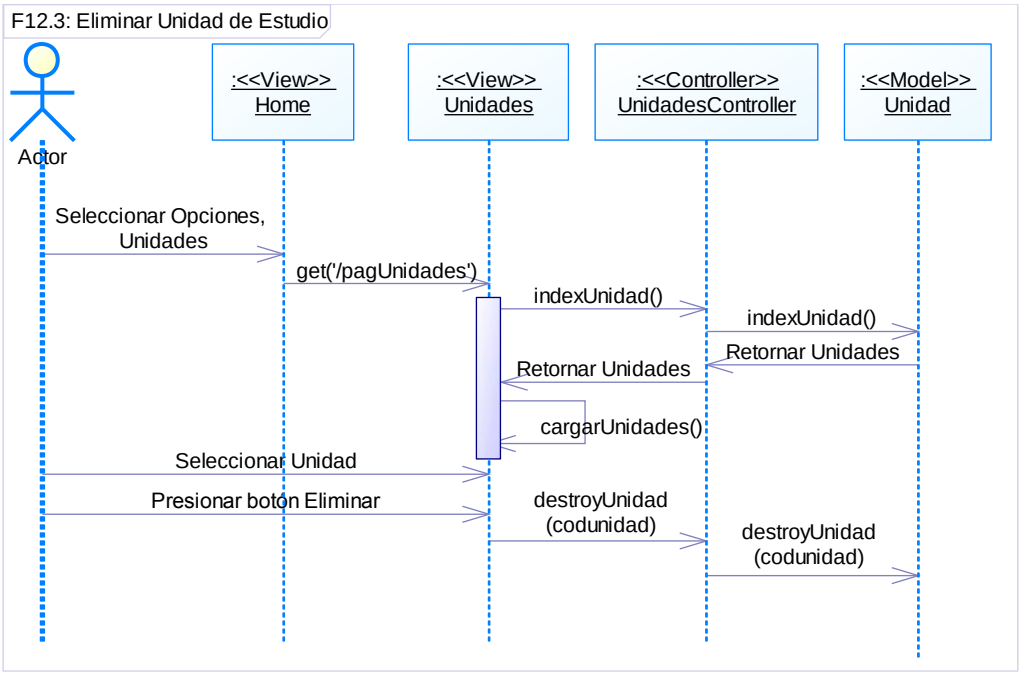


Figura 127. Diagrama Secuencia F12.3: Eliminar Unidad de Estudio

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Unidades
2. El sistema despliega la ventana de Unidades
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Unidades ingresados previamente
4. El actor presiona el botón de basurero (Eliminar) en el registro que desee eliminar
5. El sistema elimina el registro seleccionado

F12.4: Consultar Unidad de Estudio General

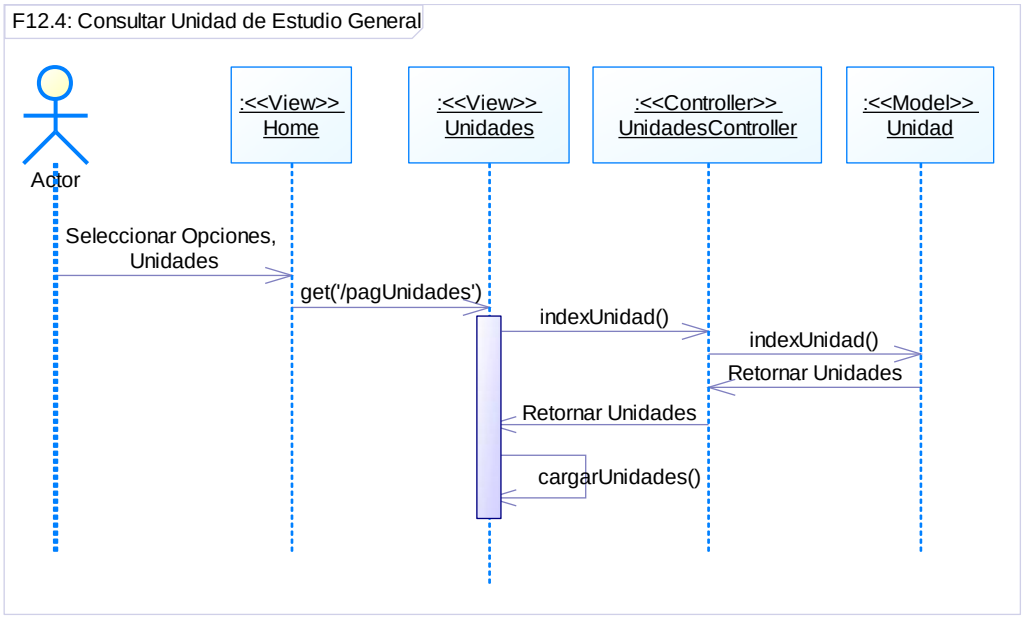


Figura 128. Diagrama Secuencia F12.4: Consultar Unidad de Estudio General

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Unidades
2. El sistema despliega la ventana de Unidades
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Unidades ingresados previamente

F12.4.1: Consultar Unidad de Estudio por Parámetros

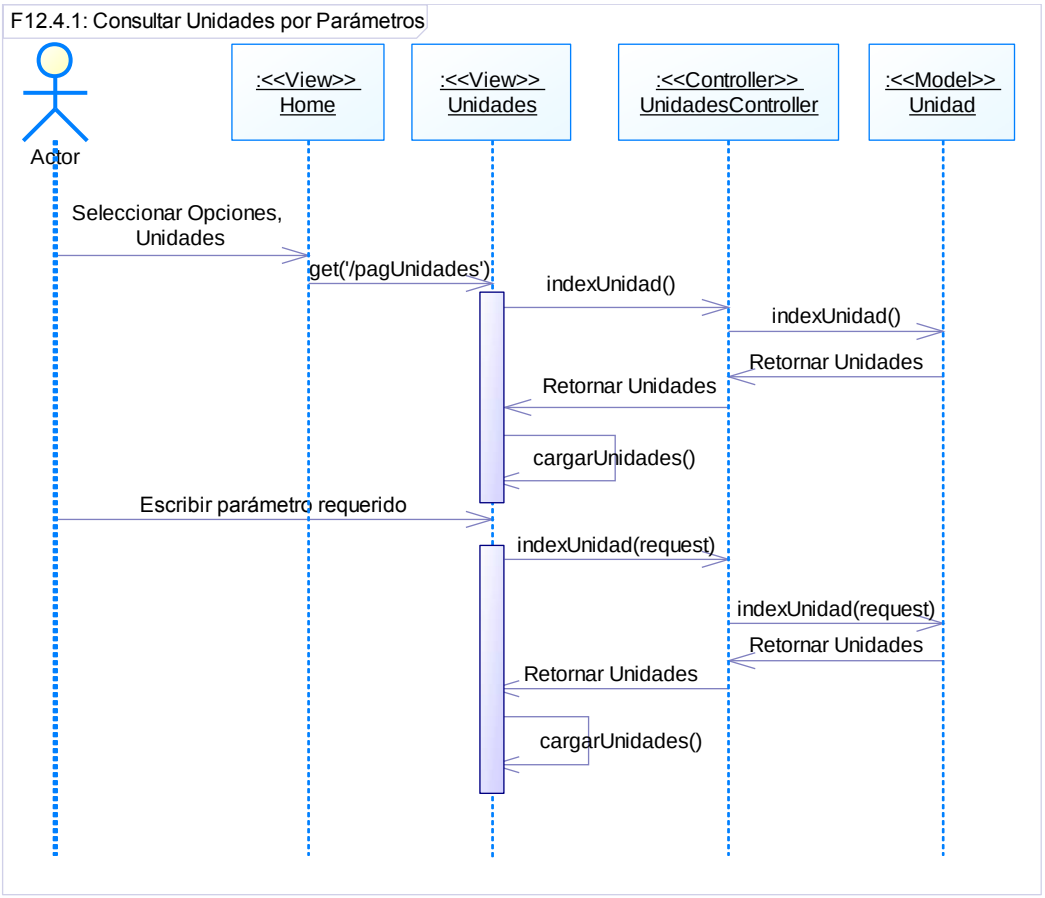


Figura 129. Diagrama Secuencia F12.4.1: Consultar Unidad de Estudio por Parámetros

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Unidades
2. El sistema despliega la ventana de Unidades
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Unidades ingresados previamente
4. El actor escribe la palabra que desea en la barra de búsqueda
5. El sistema presenta dentro de la tabla del punto 3 todos los registros que cumplen con el parámetro ingresado por el actor

F13: Gestionar Tema de Estudio

F13.1: Ingresar Tema de Estudio

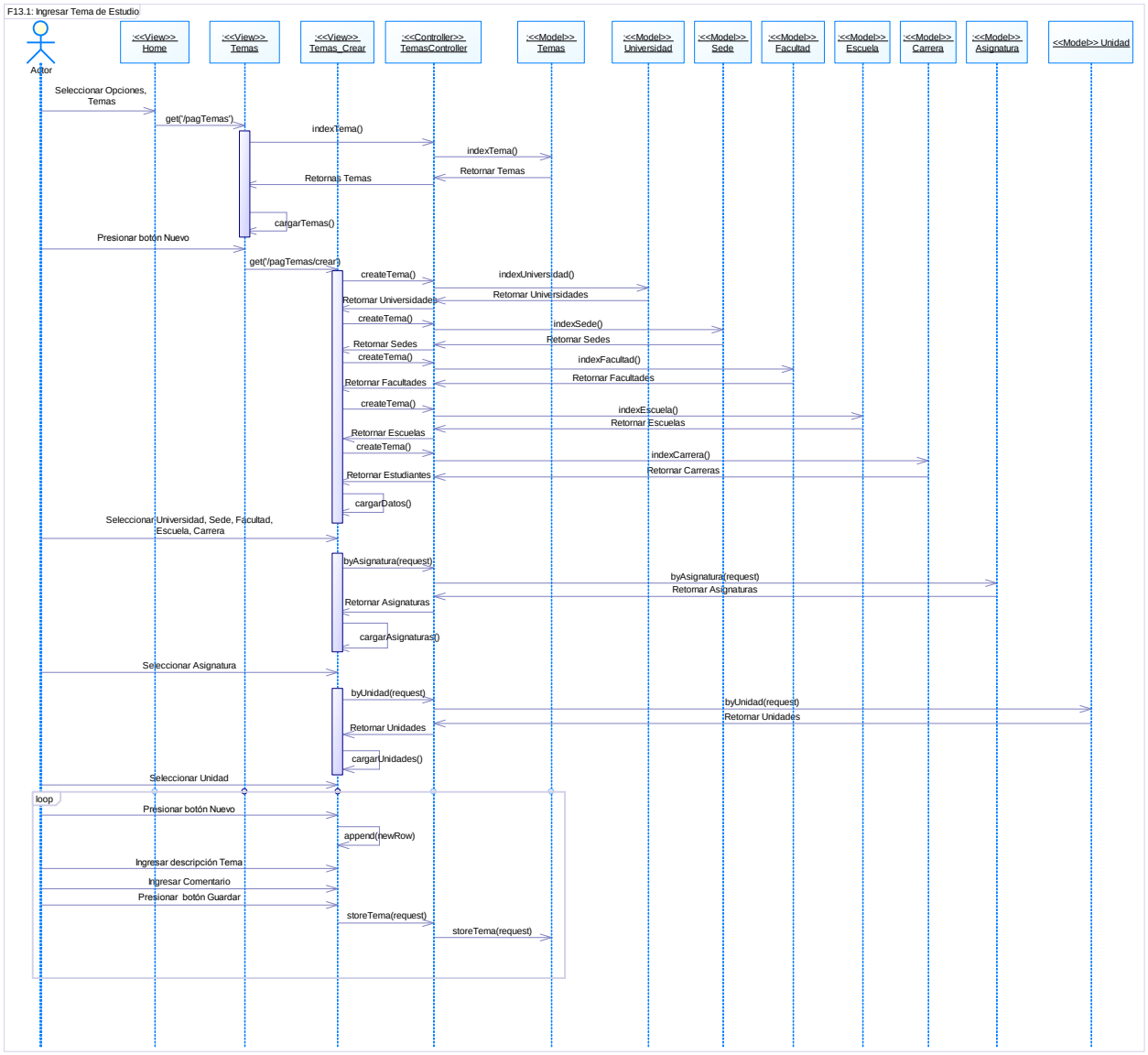


Figura 130. Diagrama Secuencia F13.1: Ingresar Tema de Estudio

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Temas
2. El sistema despliega la ventana de Temas
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Temas ingresados previamente
4. El actor presiona el botón Nuevo
5. El sistema dentro de la ventana Unidades Crear, despliega Universidad, Sede, Facultad, Escuela, Carrera
6. El usuario selecciona la Universidad, Sede, Facultad, Escuela, Carrera que desea
7. El sistema carga las Asignaturas en base a la Universidad, Sede, Facultad, Escuela, Carrera previamente seleccionada
8. El usuario elige la Asignatura que desea
9. El sistema carga las Unidades en base a la Asignatura seleccionada
10. El actor presiona el botón de Nuevo e ingresa la descripción de Tema y un comentario
11. El actor presiona el botón Guardar
12. El sistema almacena los datos

F13.2: Modificar Tema de Estudio

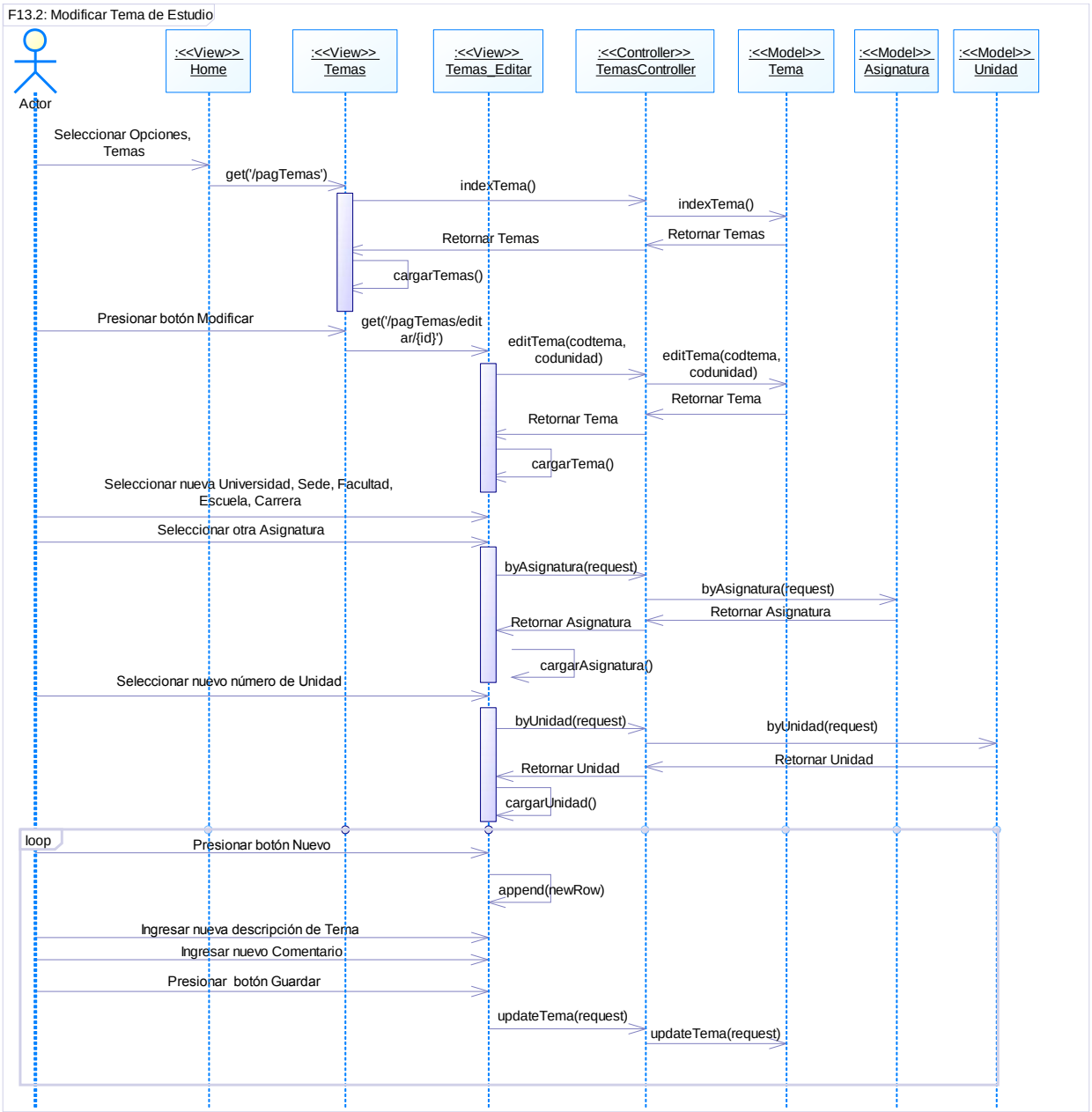


Figura 131. Diagrama Secuencia F13.2: Modificar Tema de Estudio

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Temas
2. El sistema despliega la ventana de Temas
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Temas ingresados previamente
4. El actor presiona el botón de lápiz (Modificar) en el registro que desee modificar
5. El sistema presenta la ventana Temas Editar con todos los campos llenos
6. El actor modifica los datos en los campos que desee
7. El actor presiona el botón Guardar
8. El sistema modifica los datos

F13.3: Eliminar Tema de Estudio

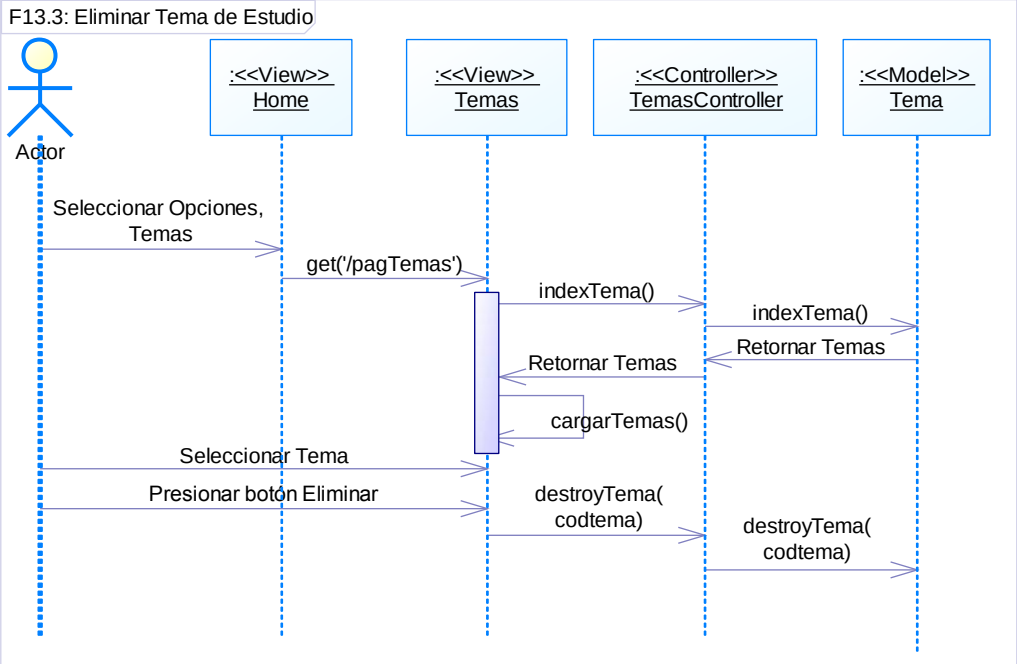


Figura 132. Diagrama Secuencia F13.3: Eliminar Tema de Estudio

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Temas
2. El sistema despliega la ventana de Temas
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Temas ingresados previamente
4. El actor presiona el botón de basurero (Eliminar) en el registro que desee eliminar

5. El sistema elimina el registro seleccionado

F13.4: Consultar Tema de Estudio General

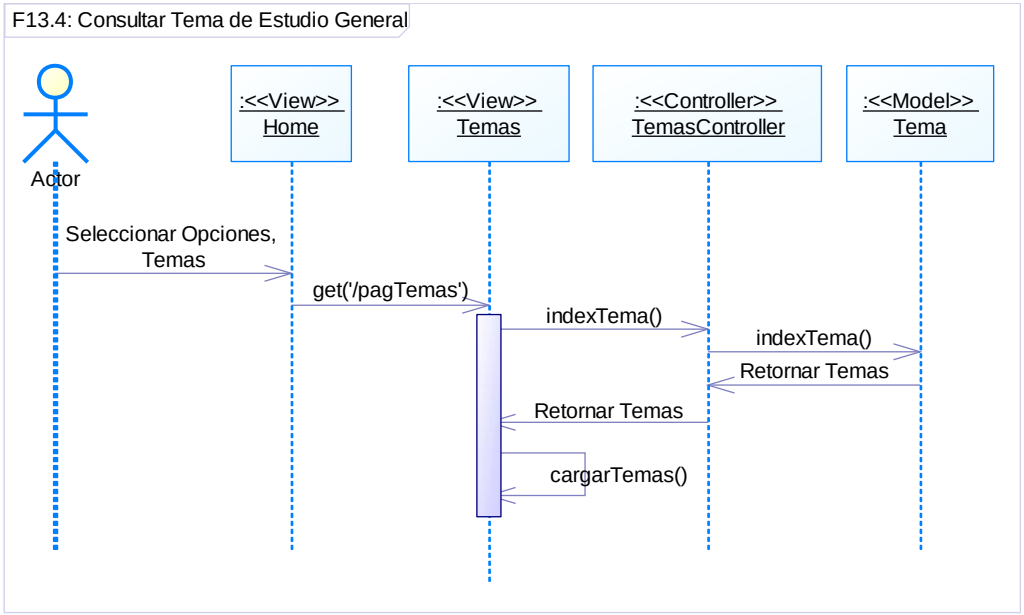


Figura 133. Diagrama Secuencia F13.4: Consultar Tema de Estudio General

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Temas
2. El sistema despliega la ventana de Temas
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Temas ingresados previamente

F13.4.1: Consultar Tema de Estudio por Parámetros

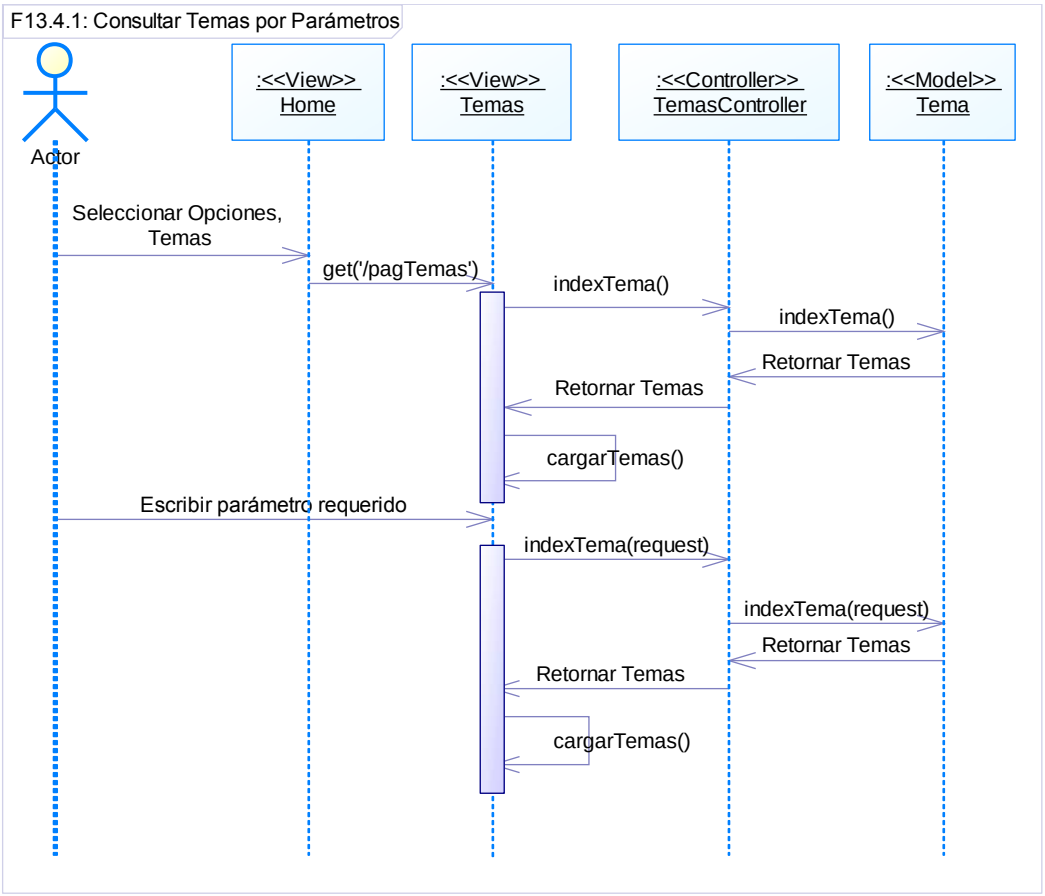


Figura 134. Diagrama Secuencia F13.4.1: Consultar Tema de Estudio por Parámetros

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Temas
2. El sistema despliega la ventana de Temas
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Temas ingresados previamente
4. El actor escribe la palabra que desea en la barra de búsqueda
5. El sistema presenta dentro de la tabla del punto 3 todos los registros que cumplen con el parámetro ingresado por el actor

F14: Gestionar Contenido

F14.1: Ingresar Contenido

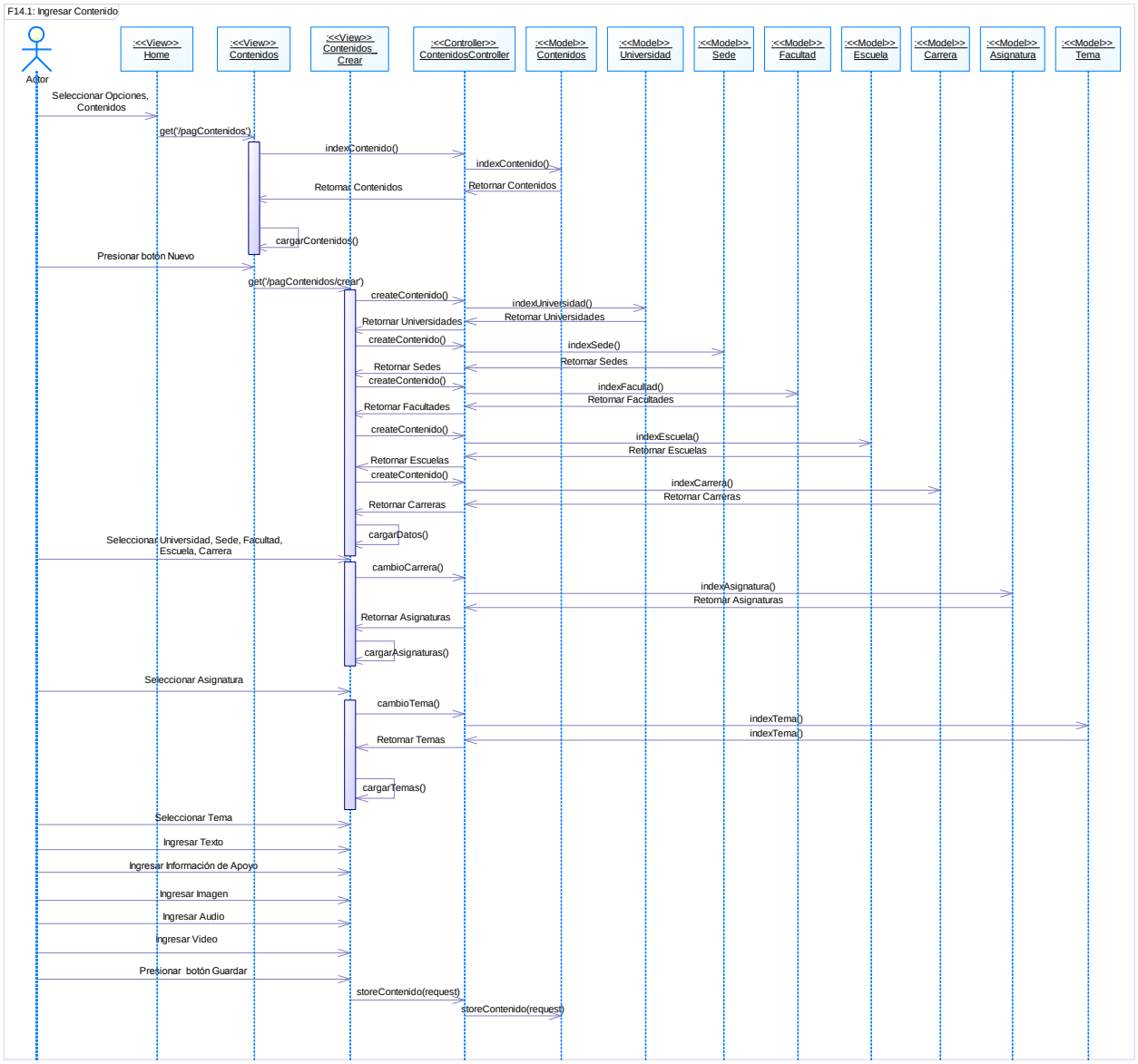


Figura 135. Diagrama Secuencia F14.1: Ingresar Contenido

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Contenidos
2. El sistema despliega la ventana de Contenidos
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Contenidos ingresados previamente
4. El actor presiona el botón Nuevo
5. El sistema dentro de la ventana Unidades Crear, despliega Universidad, Sede, Facultad, Escuela, Carrera
6. El actor selecciona la Universidad, Sede, Facultad, Escuela, Carrera que desea
7. El sistema carga las Asignaturas en base a la Universidad, Sede, Facultad, Escuela, Carrera previamente seleccionada
8. El actor elige la Asignatura que desea
9. El sistema carga los Temas en base a la Asignatura seleccionada
10. El actor selecciona el Tema que desea, ingresa el Texto, Información de Apoyo, una Imagen, un Audio y un Video
11. El actor presiona el botón Guardar
12. El sistema almacena los datos

F14.2: Modificar Contenido

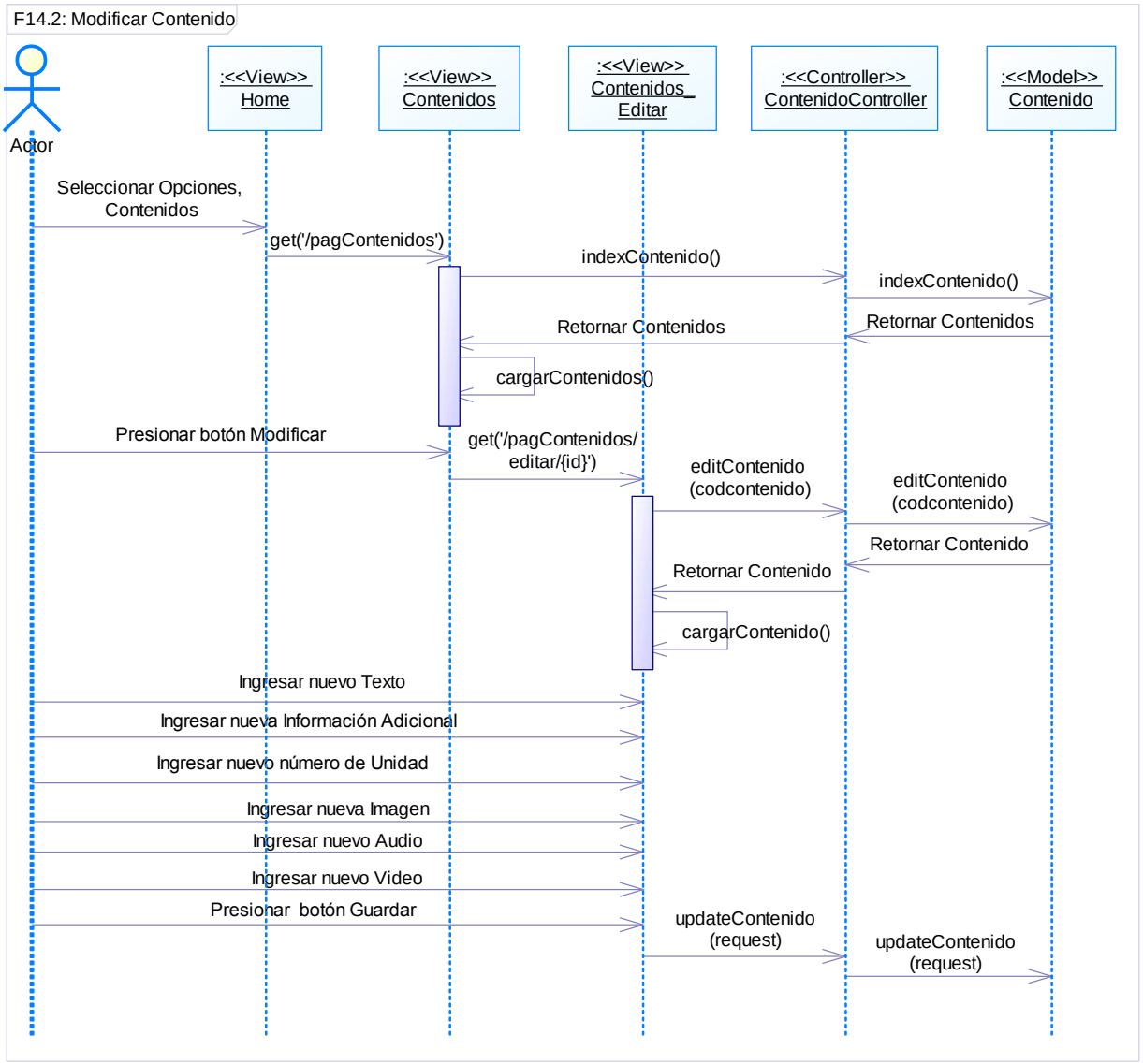


Figura 136. Diagrama Secuencia F14.2: Modificar Contenido

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Contenidos
2. El sistema despliega la ventana de Contenidos

3. El sistema presenta dentro de la ventana una tabla con todos los registros de Contenidos ingresados previamente
4. El actor presiona el botón de lápiz (Modificar) en el registro que desee modificar
5. El sistema presenta la ventana Contenidos Editar con todos los campos llenos
6. El actor modifica los datos en los campos que desee
7. El actor presiona el botón Guardar
8. El sistema modifica los datos

F14.3: Eliminar Contenido

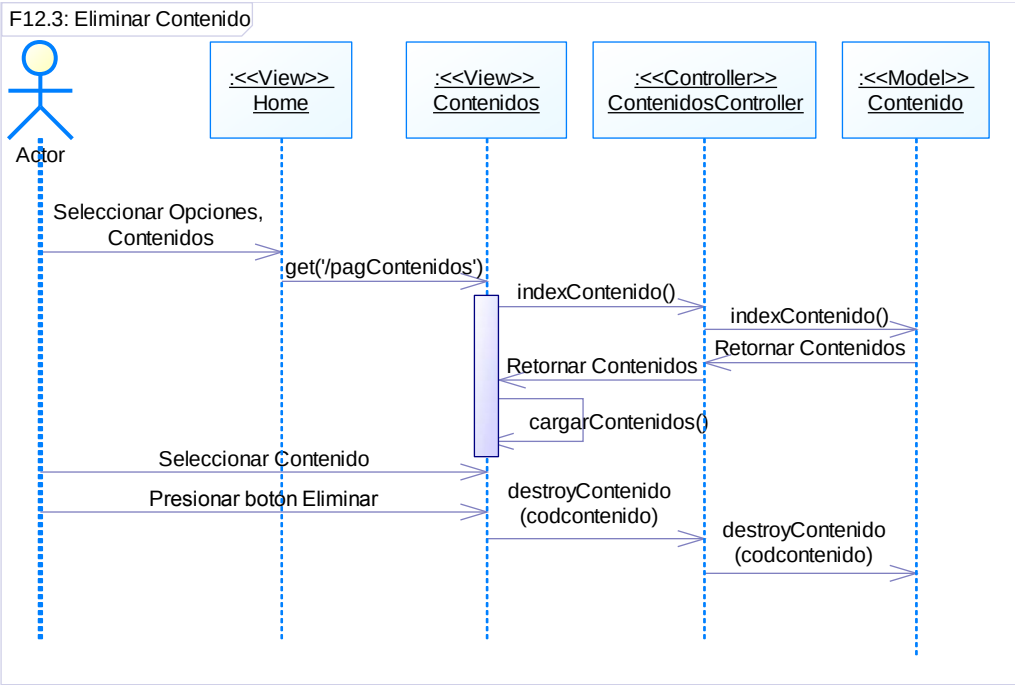


Figura 137. Diagrama Secuencia F14.3: Eliminar Contenido

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Contenidos
2. El sistema despliega la ventana de Contenidos
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Contenidos ingresados previamente
4. El actor presiona el botón de basurero (Eliminar) en el registro que desee eliminar
5. El sistema elimina el registro seleccionado

F14.4: Consultar Contenido General

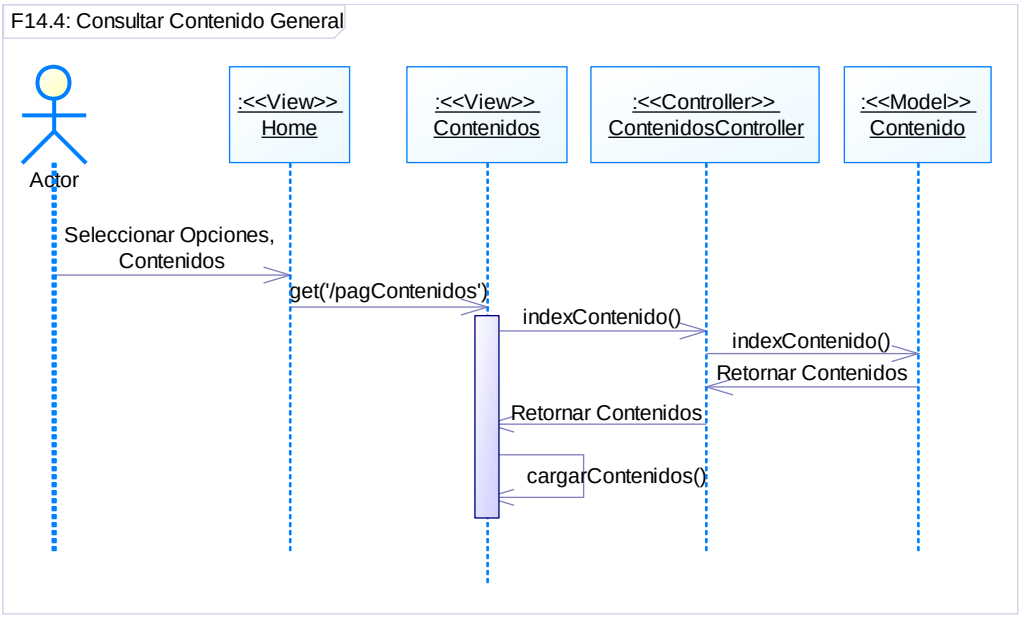


Figura 138. Diagrama Secuencia F14.4: Consultar Contenido General

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Contenidos
2. El sistema despliega la ventana de Contenidos
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Contenidos ingresados previamente

F14.4.1: Consultar Contenido por Parámetros

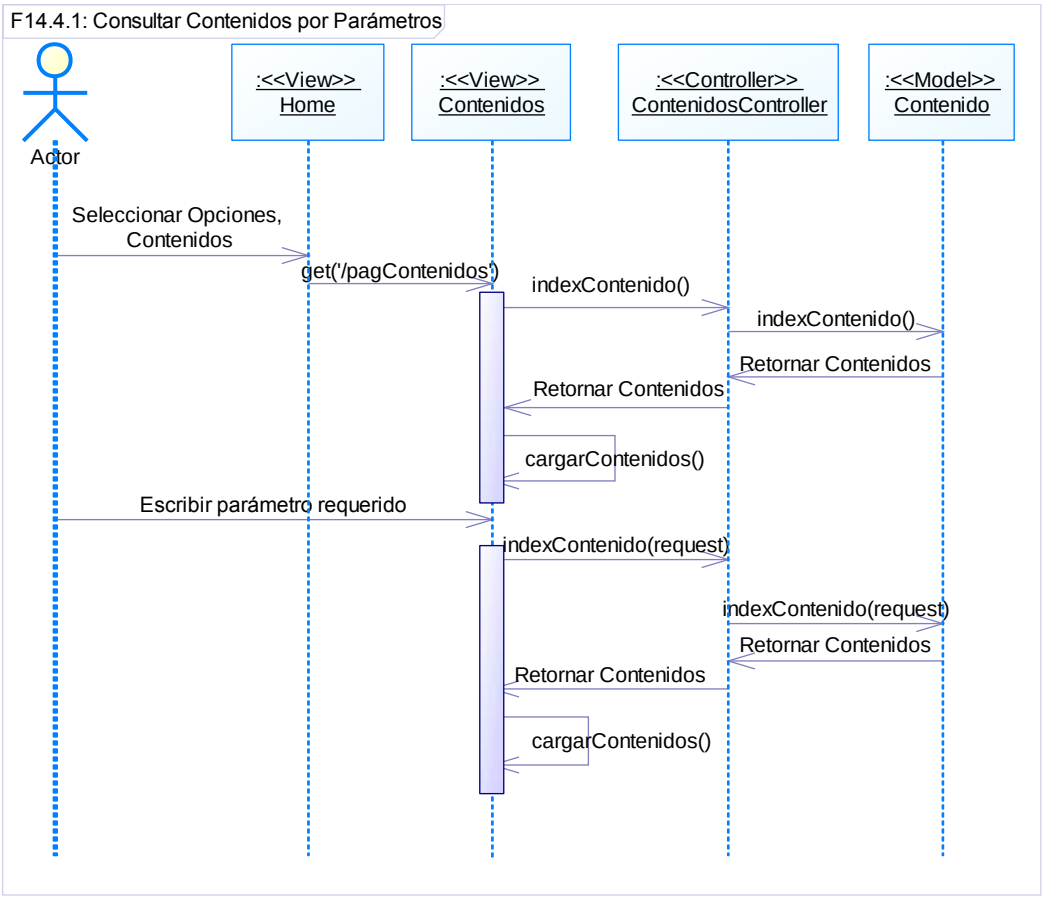


Figura 139. Diagrama Secuencia F14.4.1: Consultar Contenido por Parámetros

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Contenidos
2. El sistema despliega la ventana de Contenidos
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Contenidos ingresados previamente
4. El actor escribe la palabra que desea en la barra de búsqueda
5. El sistema presenta dentro de la tabla del punto 3 todos los registros que cumplen con el parámetro ingresado por el actor

F15: Gestionar Anotación

F15.1: Ingresar o Modificar Anotación

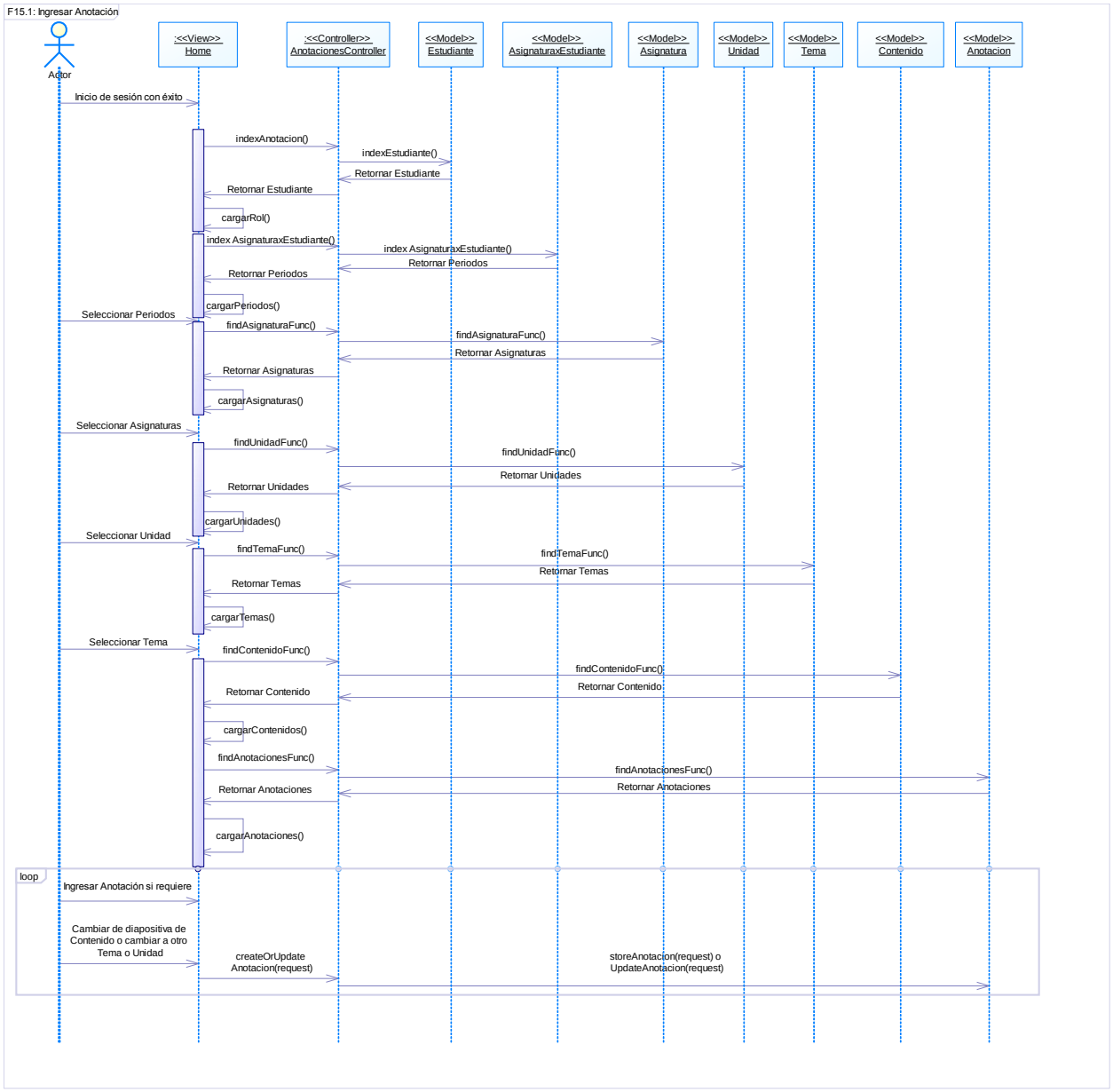


Figura 140. Diagrama Secuencia F15.1: Ingresar o Modificar Anotación

Actores

1. Estudiante

Flujo Principal

1. El sistema despliega la ventana Home
2. El sistema presenta dentro de la ventana una agenda estudiantil y despliega los Periodos en los que el actor ha estado.
3. El actor selecciona el Periodo deseado
4. El sistema carga en base al Periodo elegido las Asignaturas que el actor formó parte
5. El actor selecciona la Asignatura que desea
6. El sistema carga en base a la Asignatura seleccionada las Unidades de Estudio
7. El actor selecciona una Unidad de Estudio
8. El sistema carga los Temas de Estudio en base a la Unidad de Estudio
9. El actor selecciona un Tema de Estudio
10. El sistema carga los Contenidos en base al Tema de Estudio elegido, así como las Anotaciones que el actor ha realizado previamente
11. El actor ingresa una nueva Anotación o continúa ingresando una Anotación
12. El actor cambia de diapositiva, de Tema o de Unidad
13. El sistema almacena los datos

F15.2: Consultar Anotación por Parámetros

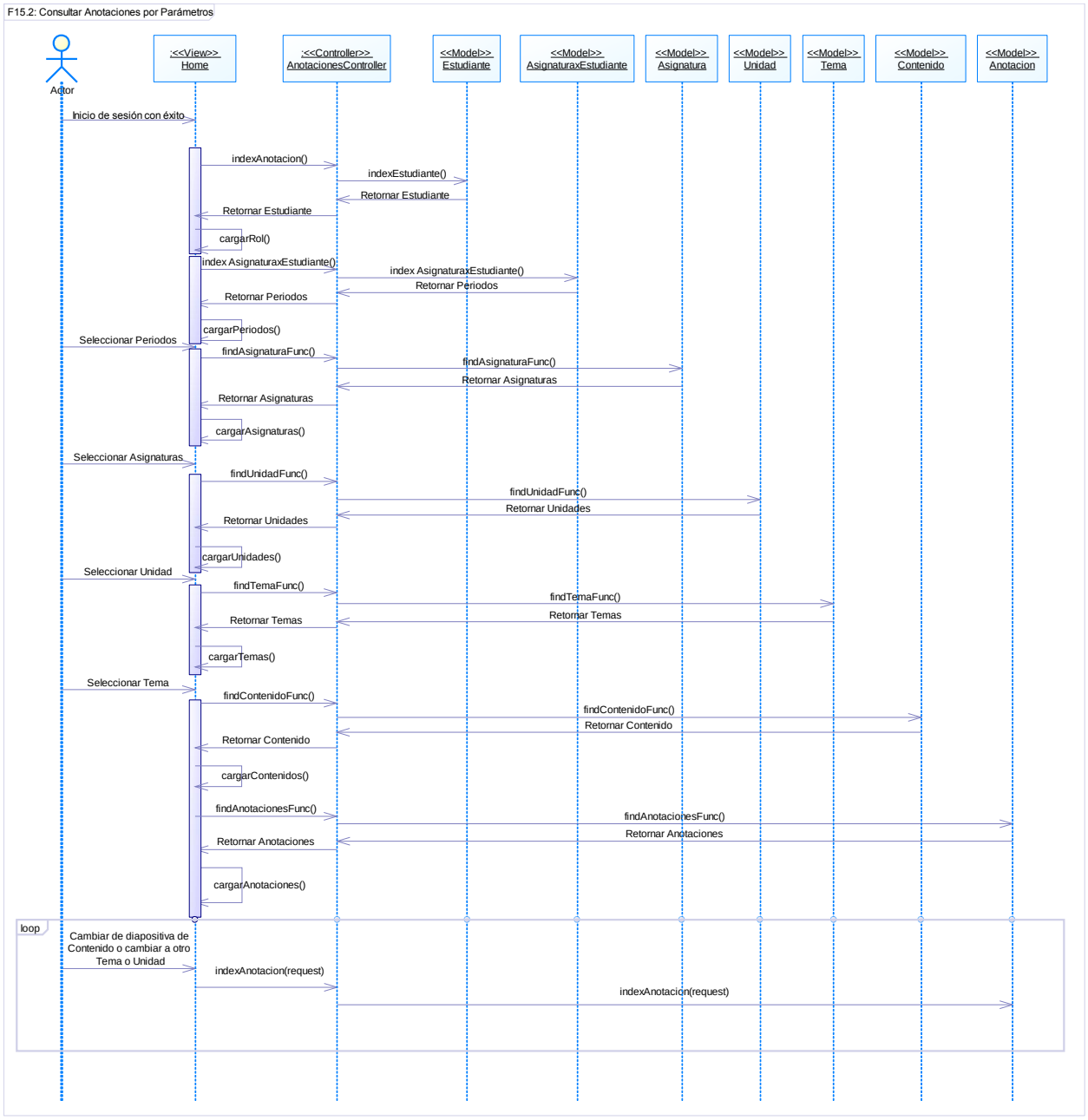


Figura 141. Diagrama Secuencia F15.2: Consultar Anotación por Parámetros

Actores

1. Estudiante

Flujo Principal

1. El sistema despliega la ventana Home
2. El sistema presenta dentro de la ventana una agenda estudiantil y despliega los Periodos en los que el actor ha estado.
3. El actor selecciona el Periodo deseado
4. El sistema carga en base al Periodo elegido las Asignaturas que el actor formó parte
5. El actor selecciona la Asignatura que desea
6. El sistema carga en base a la Asignatura seleccionada las Unidades de Estudio
7. El actor selecciona una Unidad de Estudio
8. El sistema carga los Temas de Estudio en base a la Unidad de Estudio
9. El actor selecciona un Tema de Estudio
10. El sistema carga los Contenidos en base al Tema de Estudio elegido, así como las Anotaciones que el actor ha realizado previamente
11. El actor ingresa una nueva Anotación o continúa ingresando una Anotación
12. El actor cambia de diapositiva, de Tema o de Unidad
13. El sistema carga los datos de Anotaciones para la respectiva diapositiva

F16: Gestionar Evaluación

F16.1: Ingresar Evaluación

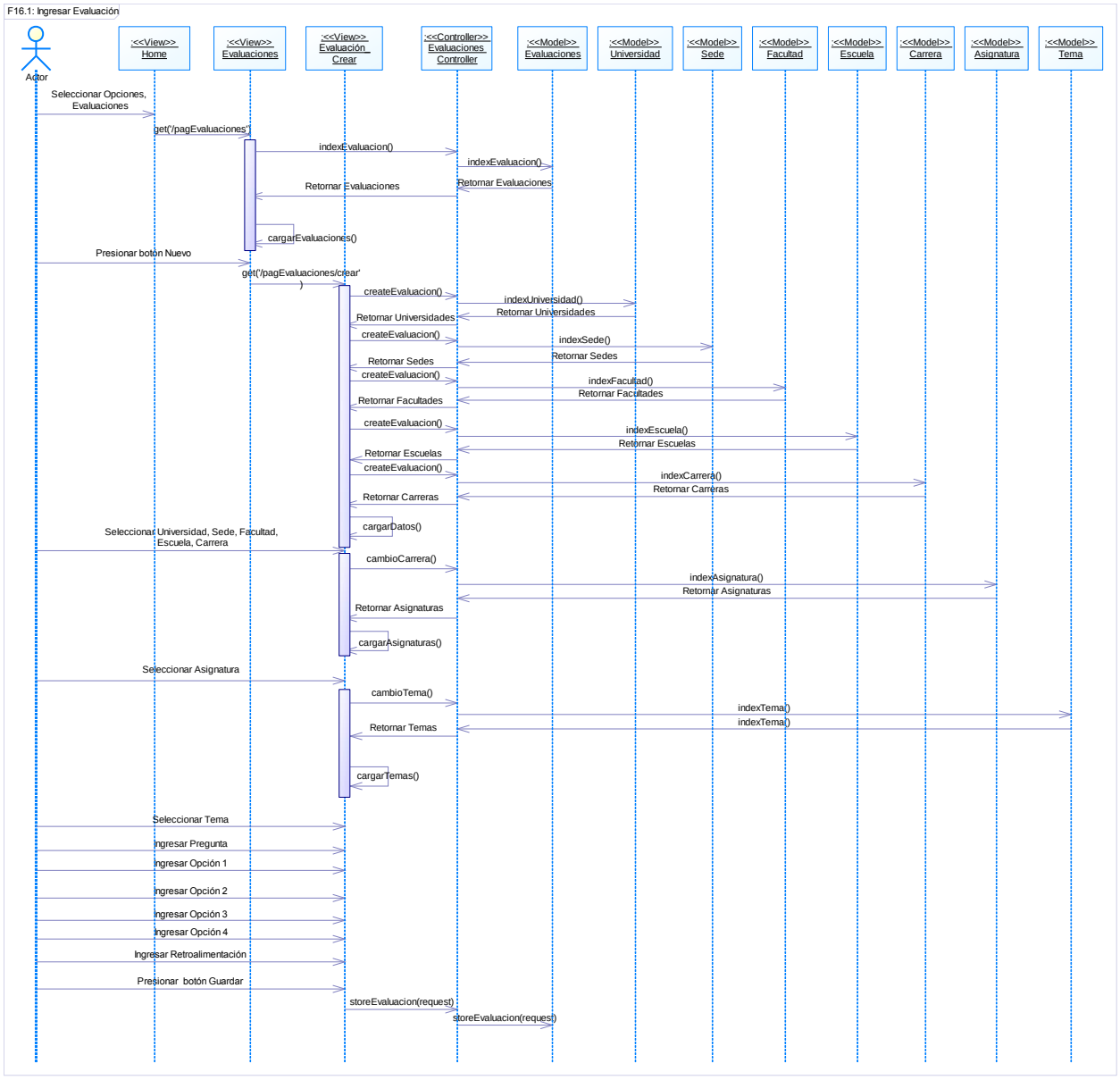


Figura 142. Diagrama Secuencia F16.1: Ingresar Evaluación

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Evaluaciones
2. El sistema despliega la ventana de Evaluaciones
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Evaluaciones ingresados previamente
4. El actor presiona el botón Nuevo
5. El sistema dentro de la ventana Unidades Crear, despliega Universidad, Sede, Facultad, Escuela, Carrera
6. El actor selecciona la Universidad, Sede, Facultad, Escuela, Carrera que desea
7. El sistema carga las Asignaturas en base a la Universidad, Sede, Facultad, Escuela, Carrera previamente seleccionada
8. El actor elige la Asignatura que desea
9. El sistema carga los Temas en base a la Asignatura seleccionada
10. El actor selecciona el Tema que desea, ingresa la Pregunta, 4 Opciones de Respuesta, la respuesta correcta y la Retroalimentación
11. El actor presiona el botón Guardar
12. El sistema almacena los datos

F16.2: Modificar Evaluación

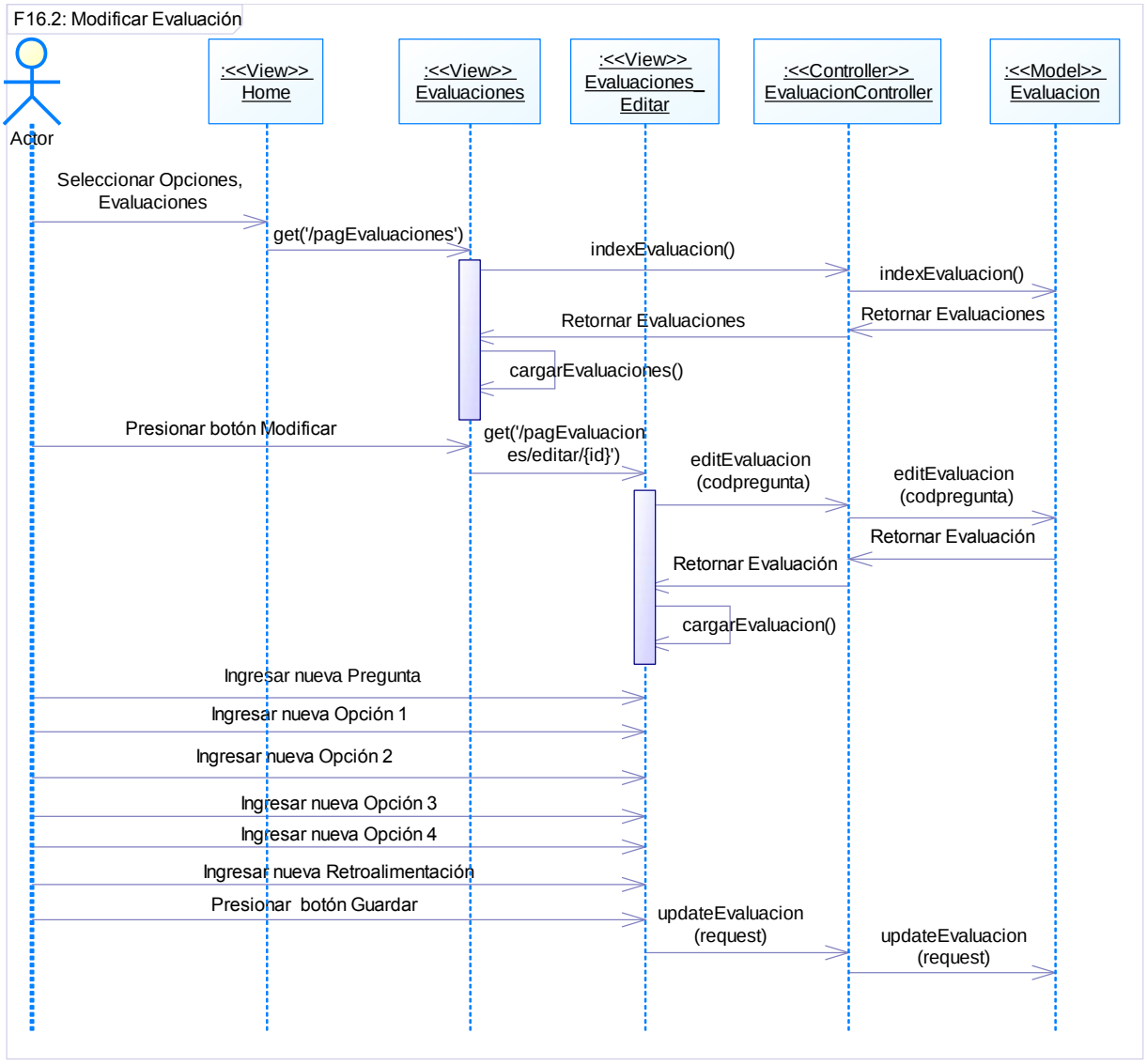


Figura 143. Diagrama Secuencia F16.2: Modificar Evaluación

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Evaluaciones
2. El sistema despliega la ventana de Evaluaciones

3. El sistema presenta dentro de la ventana una tabla con todos los registros de Evaluaciones ingresados previamente
4. El actor presiona el botón de lápiz (Modificar) en el registro que desee modificar
5. El sistema presenta la ventana Evaluaciones Editar con todos los campos llenos
6. El actor modifica los datos en los campos que desee
7. El actor presiona el botón Guardar
8. El sistema modifica los datos

F16.3: Eliminar Evaluación

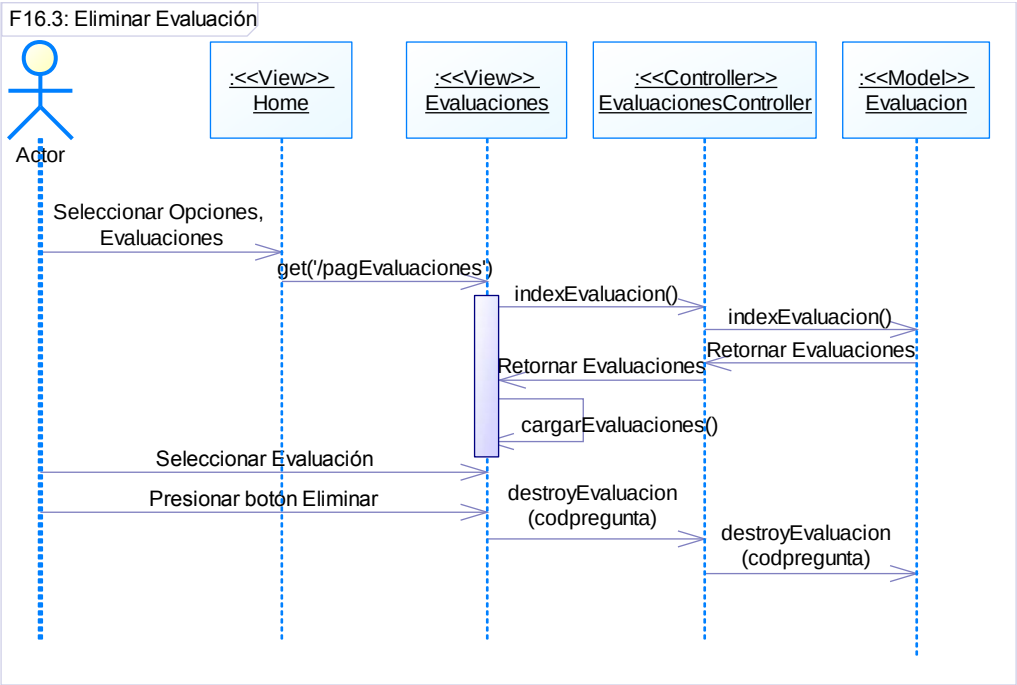


Figura 144. Diagrama Secuencia F16.3: Eliminar Evaluación

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Evaluaciones
2. El sistema despliega la ventana de Evaluaciones
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Evaluaciones ingresados previamente
4. El actor presiona el botón de basurero (Eliminar) en el registro que desee eliminar
5. El sistema elimina el registro seleccionado

F16.4: Consultar Evaluación General

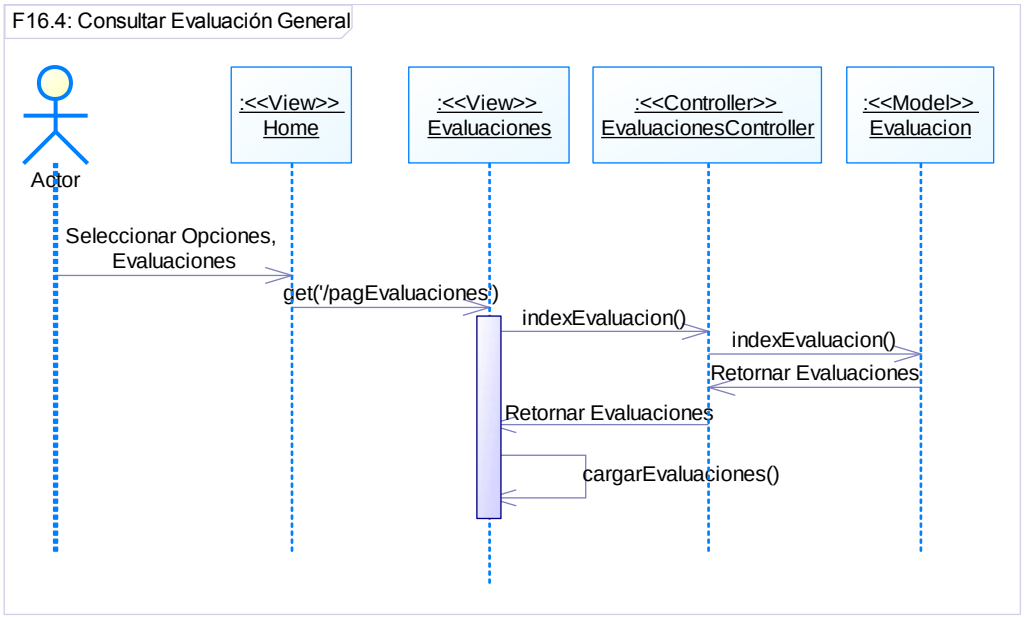


Figura 145. Diagrama Secuencia F16.4: Consultar Evaluación General

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Evaluaciones
2. El sistema despliega la ventana de Evaluaciones
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Evaluaciones ingresados previamente

F16.4.1: Consultar Evaluación por Parámetros

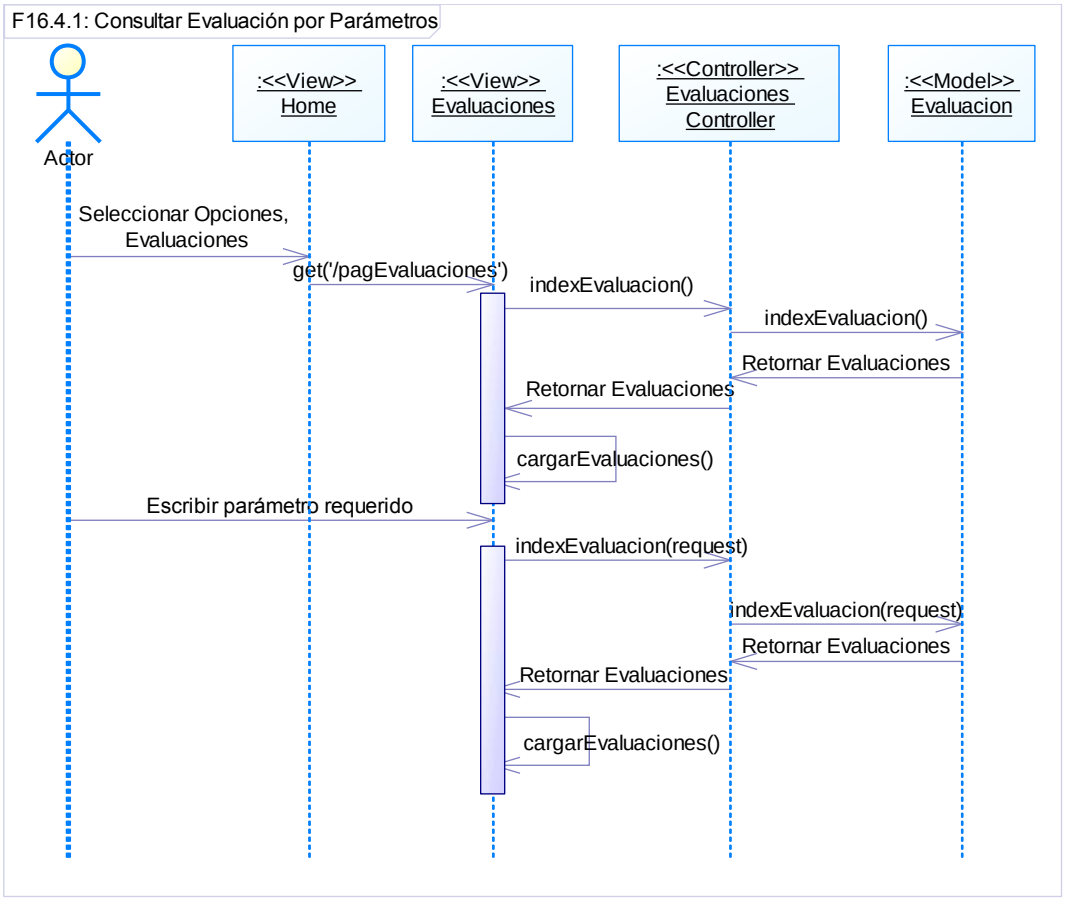


Figura 146. Diagrama Secuencia F16.4.1: Consultar Evaluación por Parámetros

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Evaluaciones
2. El sistema despliega la ventana de Evaluaciones
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Evaluaciones ingresados previamente
4. El actor escribe la palabra que desea en la barra de búsqueda
5. El sistema presenta dentro de la tabla del punto 3 todos los registros que cumplen con el parámetro ingresado por el actor

F17: Gestionar Glosario

F17.1: Ingresar Glosario

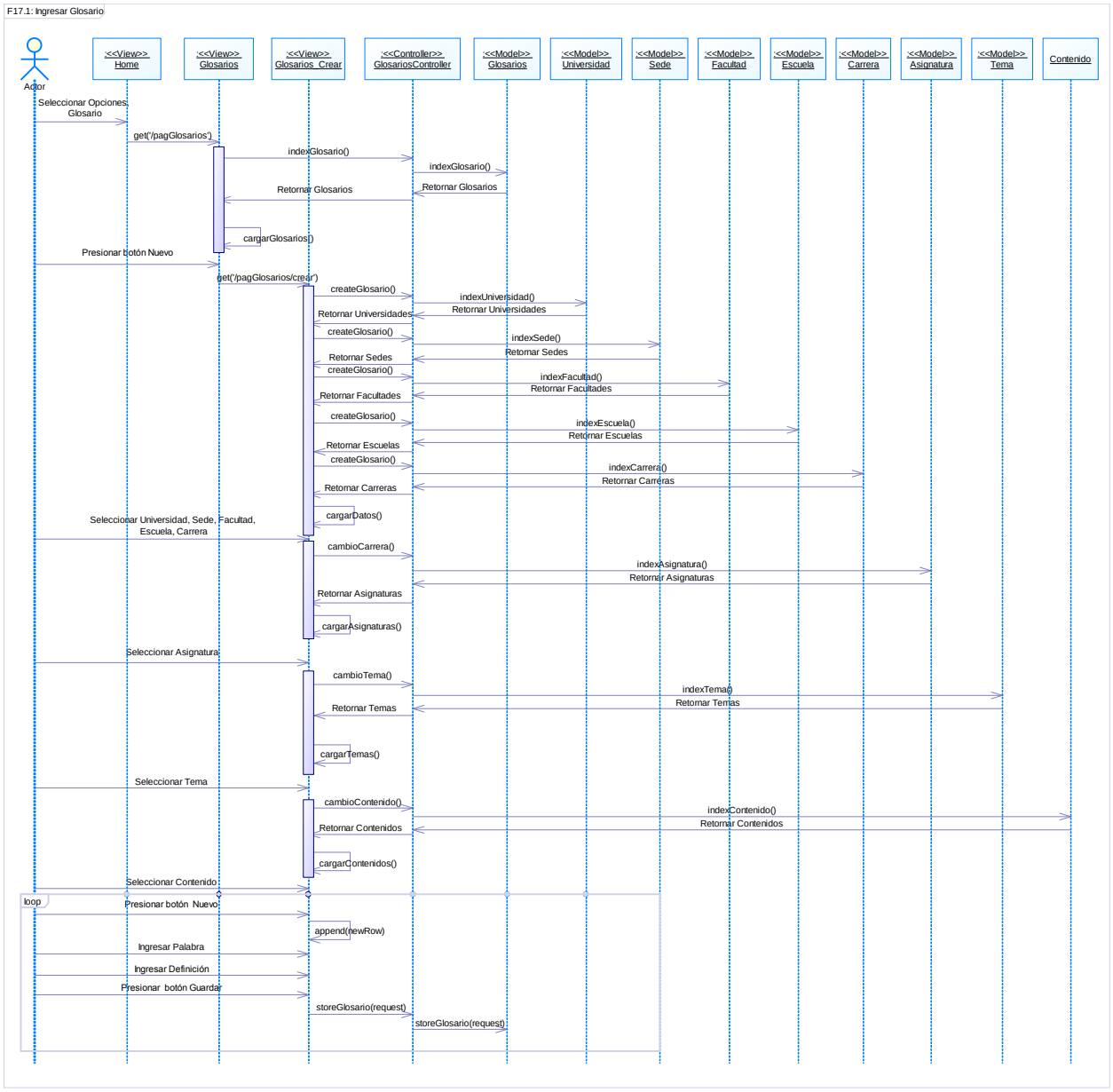


Figura 147. Diagrama Secuencia F17.1: Ingresar Glosario

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Glosarios
2. El sistema despliega la ventana de Glosarios
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Glosarios ingresados previamente
4. El actor presiona el botón Nuevo
5. El sistema dentro de la ventana Unidades Crear, despliega Universidad, Sede, Facultad, Escuela, Carrera
6. El actor selecciona la Universidad, Sede, Facultad, Escuela, Carrera que desea
7. El sistema carga las Asignaturas en base a la Universidad, Sede, Facultad, Escuela, Carrera previamente seleccionada
8. El actor elige la Asignatura que desea
9. El sistema carga los Temas en base a la Asignatura seleccionada
10. El actor selecciona el Tema que desea
11. El sistema carga los Contenidos en base al Tema seleccionado
12. El actor selecciona un Contenido e ingresa la Palabra y la Definición
13. El actor presiona el botón Guardar
14. El sistema almacena los datos

F17.2: Modificar Glosario

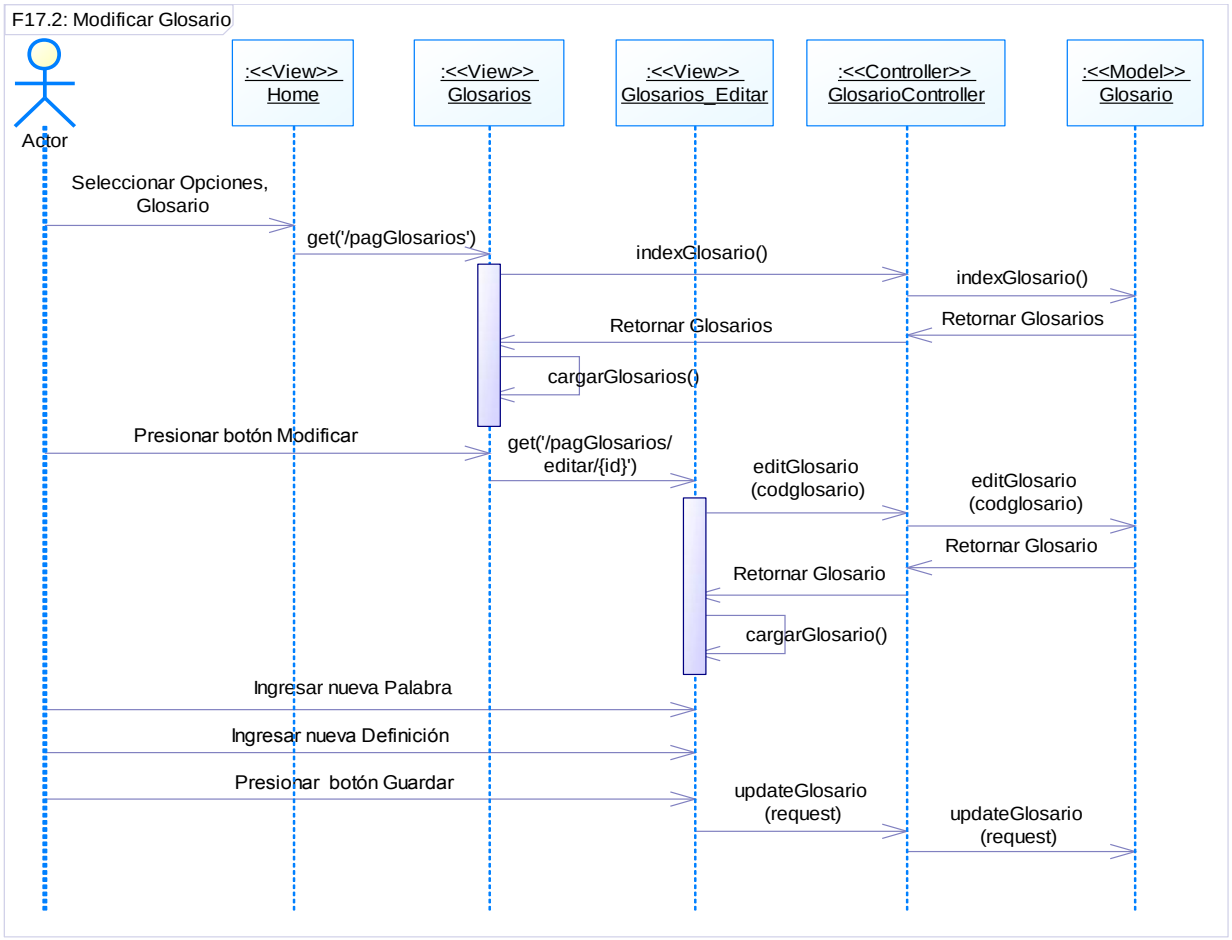


Figura 148. Diagrama Secuencia F17.2: Modificar Glosario

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Glosarios
2. El sistema despliega la ventana de Glosarios
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Glosarios ingresados previamente
4. El actor presiona el botón de lápiz (Modificar) en el registro que desee modificar
5. El sistema presenta la ventana Glosarios Editar con todos los campos llenos
6. El actor modifica los datos en los campos que desee

7. El actor presiona el botón Guardar
8. El sistema modifica los datos

F17.3: Eliminar Glosario

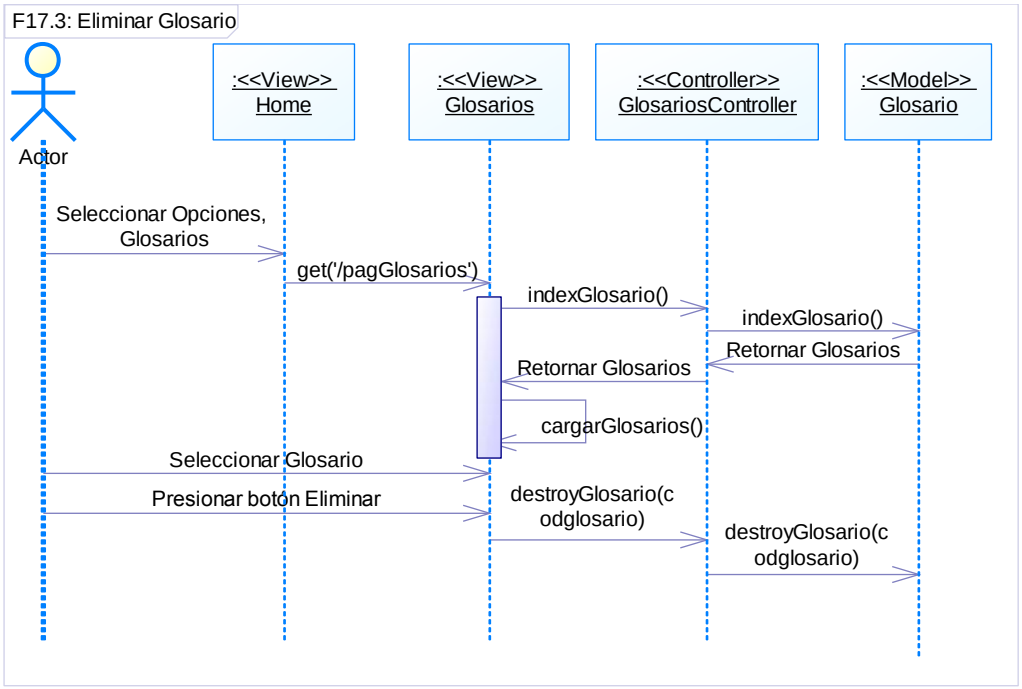


Figura 149. Diagrama Secuencia F17.3: Eliminar Glosario

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Glosarios
2. El sistema despliega la ventana de Glosarios
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Glosarios ingresados previamente
4. El actor presiona el botón de basurero (Eliminar) en el registro que desee eliminar
5. El sistema elimina el registro seleccionado

F17.4: Consultar Glosario General

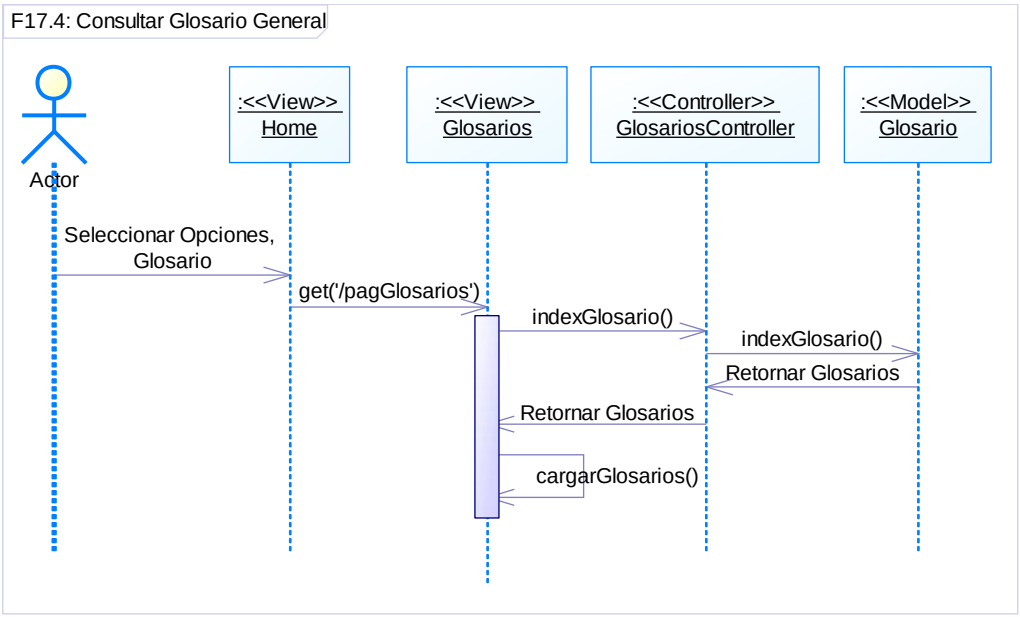


Figura 150. Diagrama Secuencia F17.4: Consultar Glosario General

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Glosarios
2. El sistema despliega la ventana de Glosarios
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Glosarios ingresados previamente

F17.4.1: Consultar Glosario por Parámetros

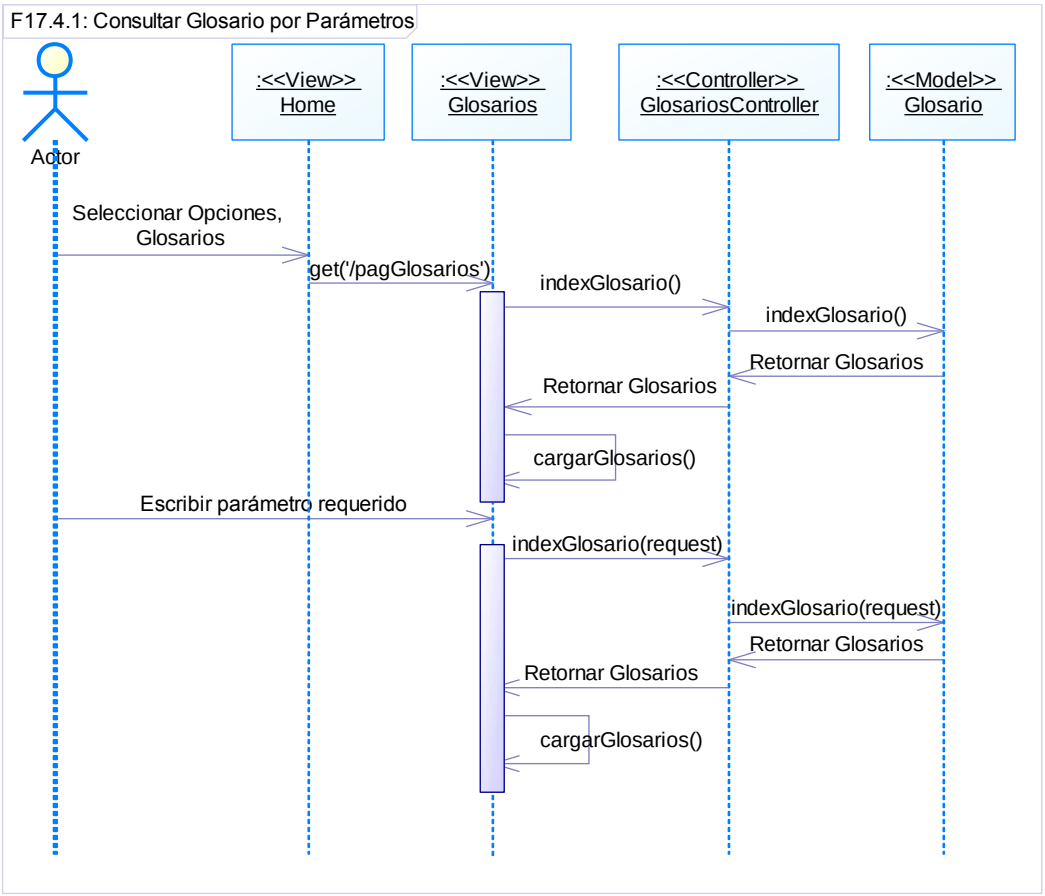


Figura 151. Diagrama Secuencia F17.4.1: Consultar Glosario por Parámetros

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Glosarios
2. El sistema despliega la ventana de Glosarios
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Glosarios ingresados previamente
4. El actor escribe la palabra que desea en la barra de búsqueda
5. El sistema presenta dentro de la tabla del punto 3 todos los registros que cumplen con el parámetro ingresado por el actor

F18: Gestionar Taller

F18.1: Ingresar Taller

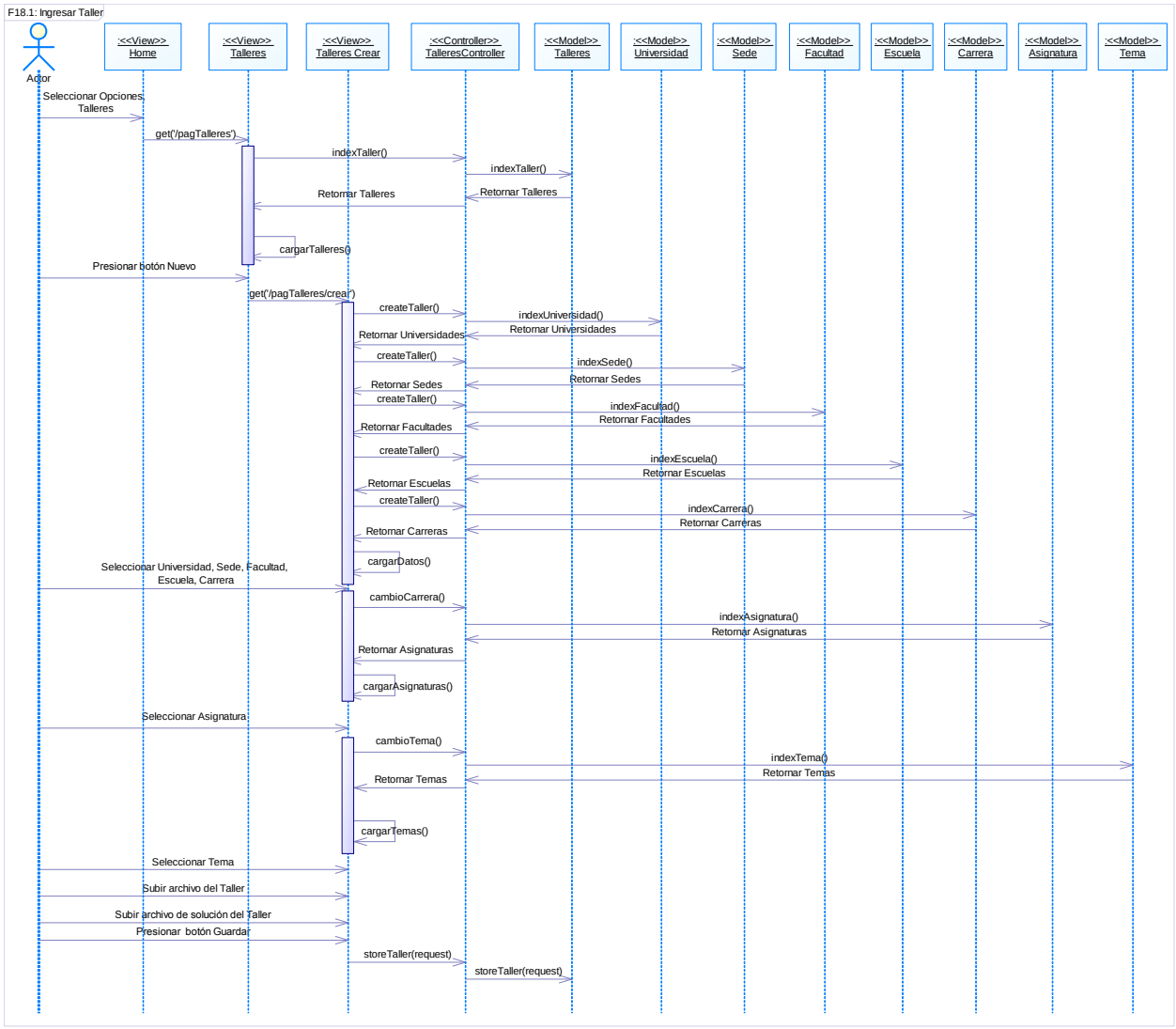


Figura 152. Diagrama Secuencia F18.1: Ingresar Taller

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Talleres
2. El sistema despliega la ventana de Talleres

3. El sistema presenta dentro de la ventana una tabla con todos los registros de Talleres ingresados previamente
4. El actor presiona el botón Nuevo
5. El sistema dentro de la ventana Unidades Crear, despliega Universidad, Sede, Facultad, Escuela, Carrera
6. El actor selecciona la Universidad, Sede, Facultad, Escuela, Carrera que desea
7. El sistema carga las Asignaturas en base a la Universidad, Sede, Facultad, Escuela, Carrera previamente seleccionada
8. El actor elige la Asignatura que desea
9. El sistema carga los Temas en base a la Asignatura seleccionada
10. El actor selecciona el Tema que desea, sube el archivo del Taller y el archivo de solución del Taller.
11. El actor presiona el botón Guardar
12. El sistema almacena los datos

F18.2: Modificar Taller

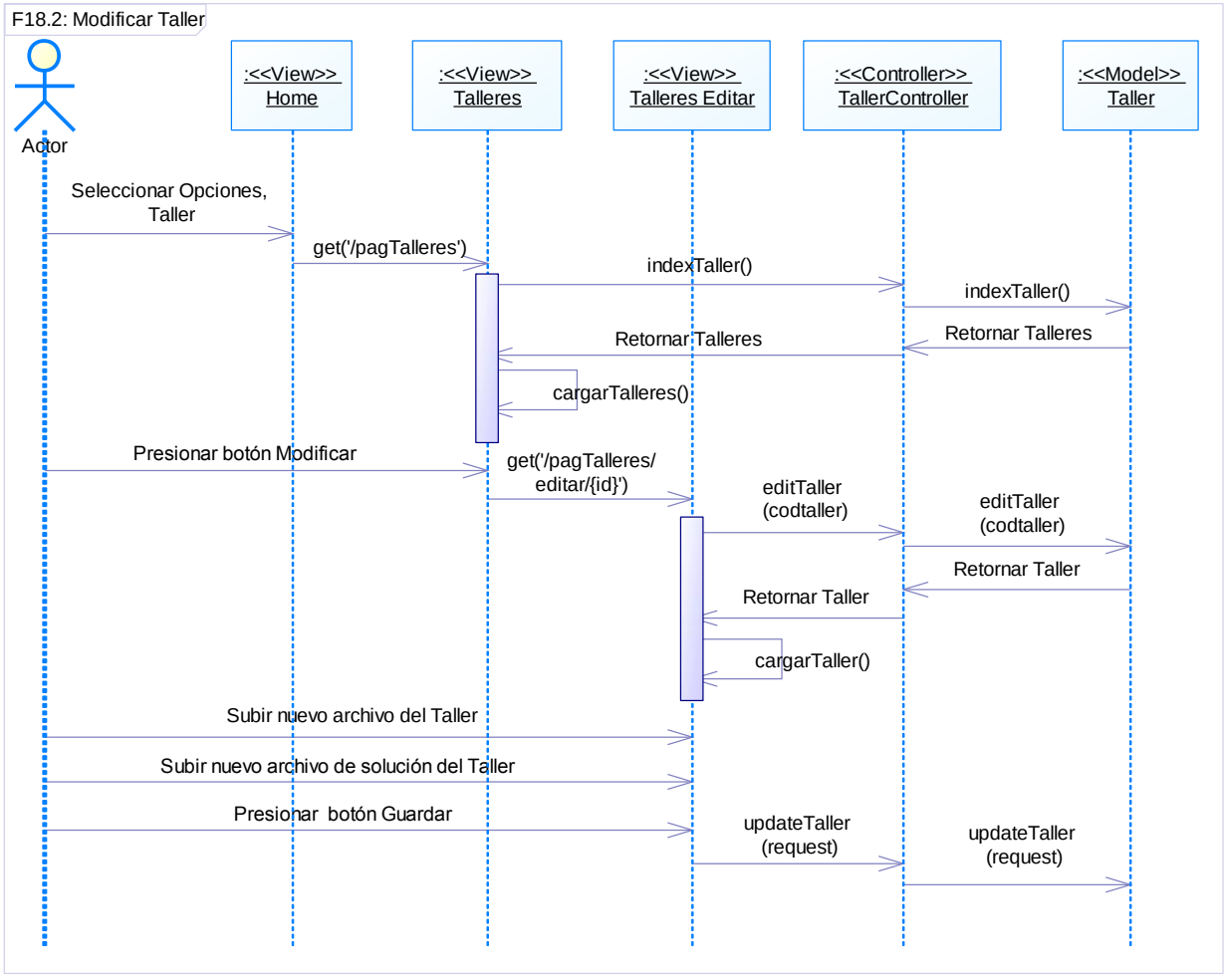


Figura 153. Diagrama Secuencia F18.2: Modificar Taller

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Talleres
2. El sistema despliega la ventana de Talleres
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Talleres ingresados previamente
4. El actor presiona el botón de lápiz (Modificar) en el registro que desee modificar
5. El sistema presenta la ventana Talleres Editar con todos los campos llenos
6. El actor modifica los datos en los campos que desee
7. El actor presiona el botón Guardar
8. El sistema modifica los datos

F18.3: Eliminar Taller

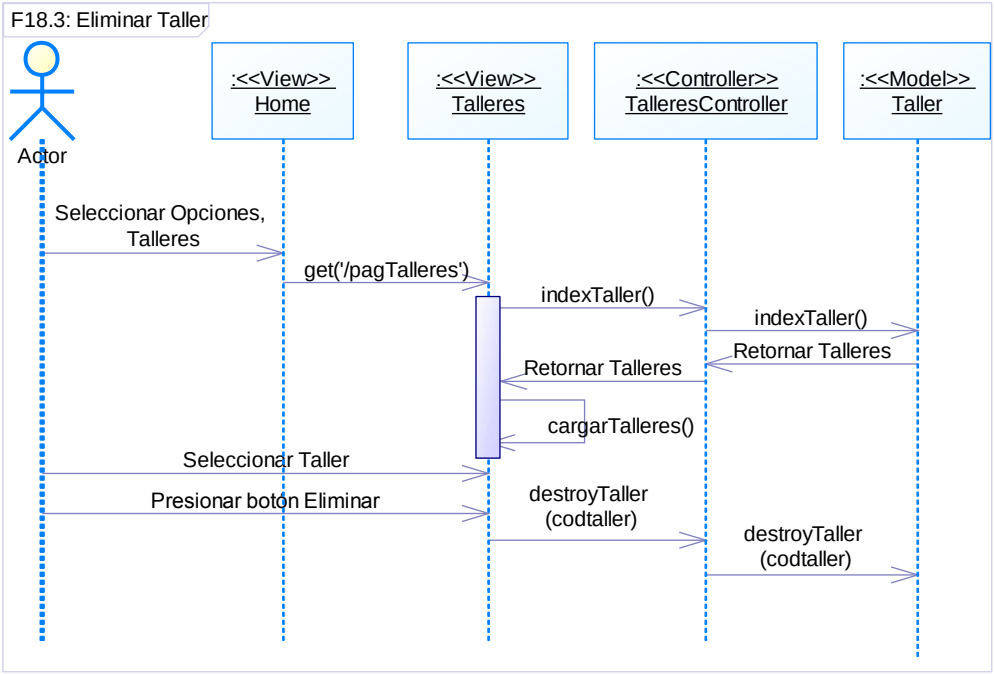


Figura 154. Diagrama Secuencia F18.3: Eliminar Taller

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Talleres
2. El sistema despliega la ventana de Talleres
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Talleres ingresados previamente
4. El actor presiona el botón de basurero (Eliminar) en el registro que desee eliminar
5. El sistema elimina el registro seleccionado

F18.4: Consultar Taller General

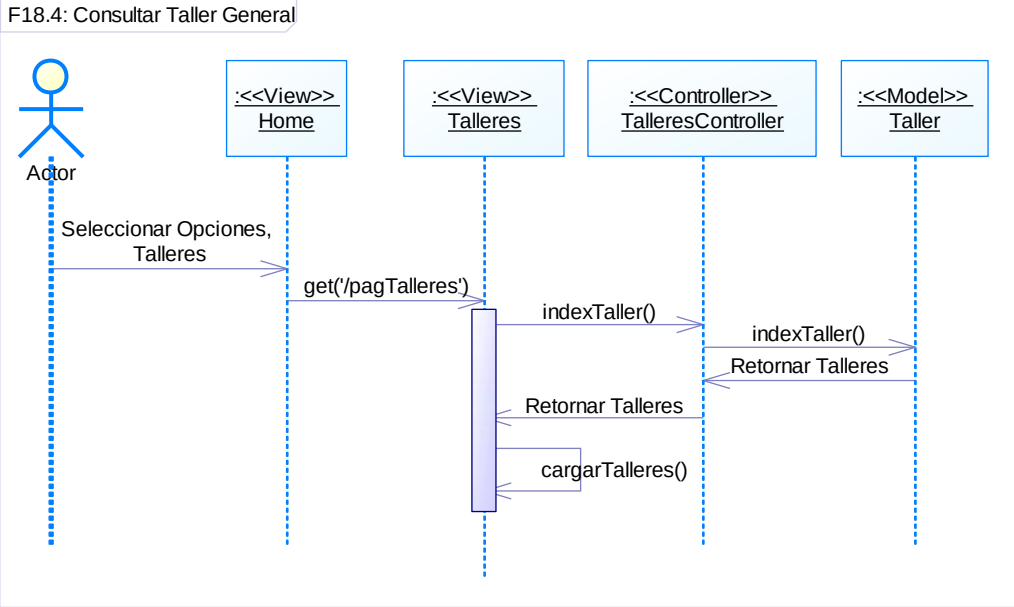


Figura 155. Diagrama Secuencia F18.4: Consultar Taller General

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Talleres
2. El sistema despliega la ventana de Talleres

3. El sistema presenta dentro de la ventana una tabla con todos los registros de Talleres ingresados previamente

F18.4.1: Consultar Taller por Parámetros

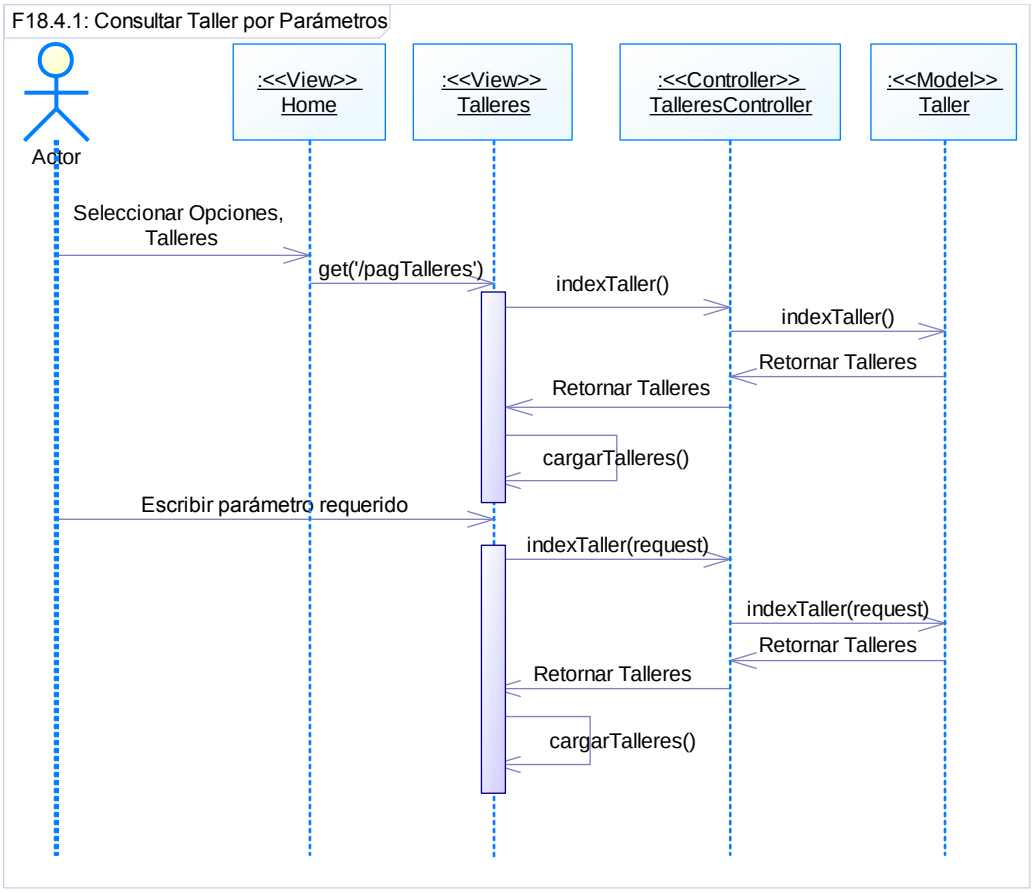


Figura 156. Diagrama Secuencia F18.4.1: Consultar Taller por Parámetros

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Talleres
2. El sistema despliega la ventana de Talleres
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Talleres ingresados previamente
4. El actor escribe la palabra que desea en la barra de búsqueda

5. El sistema presenta dentro de la tabla del punto 3 todos los registros que cumplen con el parámetro ingresado por el actor

F19: Gestionar Asignación de Profesores por Asignatura

F19.1: Ingresar Asignación de Profesores por Asignatura

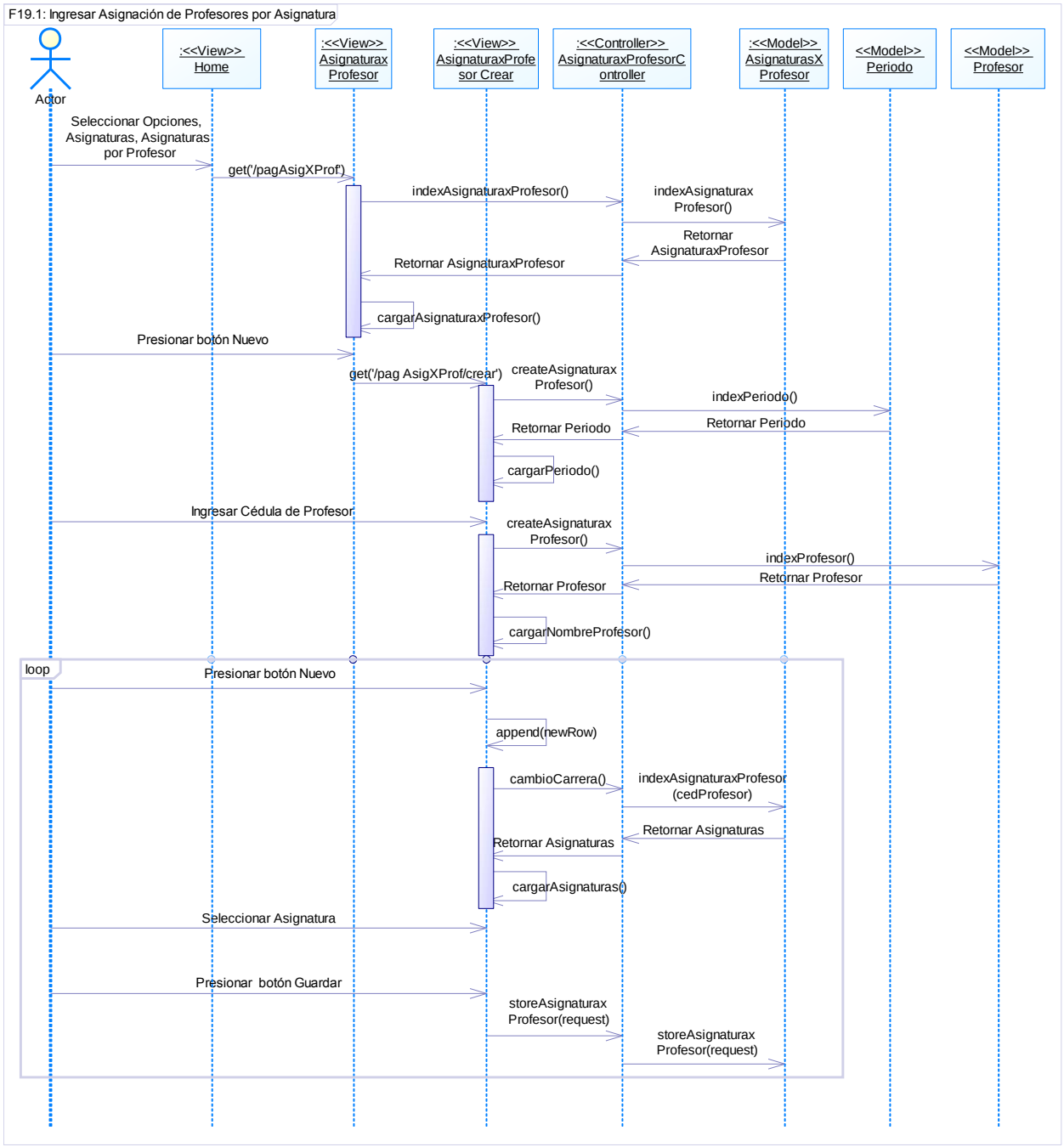


Figura 157. Diagrama Secuencia F19.1: Ingresar Asignación de Profesores por Asignatura

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Asignaturas y en Asignaturas por Profesor
2. El sistema despliega la ventana de AsignaturaxProfesor
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Asignaturas por Profesor ingresados previamente
4. El actor presiona el botón Nuevo
5. El sistema dentro de la ventana AsignaturaxProfesor Crear, despliega el Periodo actual
6. El actor ingresa la cédula del Profesor
7. El sistema carga el nombre del Profesor
8. El actor presiona el botón de Nuevo
9. El sistema agrega una fila y despliega las Asignaturas que el Profesor puede dictar
10. El actor elige la Asignatura que desea y presiona el botón Guardar
11. El sistema almacena los datos

F19.2: Modificar Asignación de Profesores por Asignatura

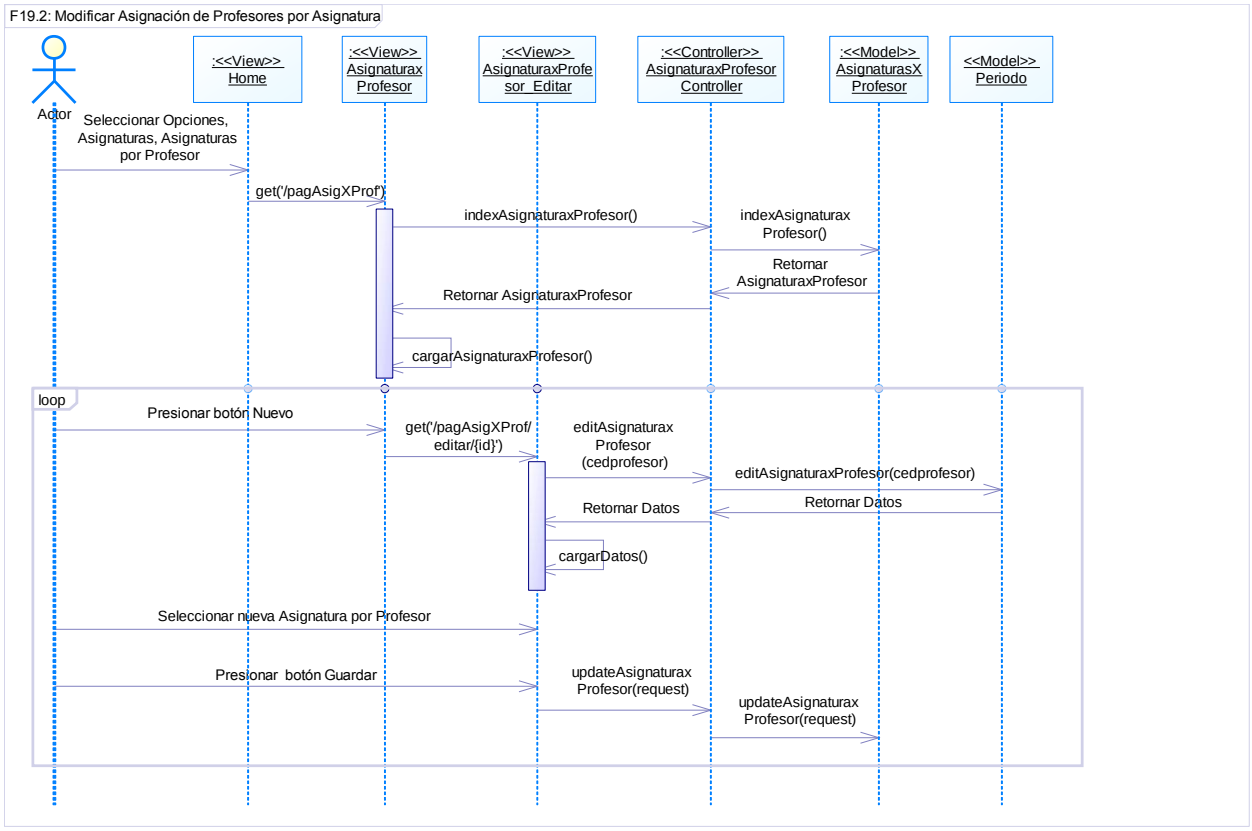


Figura 158. Diagrama Secuencia F19.2: Modificar Asignación de Profesores por Asignatura

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Asignaturas y en Asignaturas por Profesor
2. El sistema despliega la ventana de AsignaturaxProfesor
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Asignaturas por Profesor ingresados previamente
9. El actor presiona el botón de lápiz (Modificar) en el registro que desee modificar
10. El sistema dentro de la ventana AsignaturaxProfesor Editar, despliega todos los campos llenos y las Asignaturas que dicta
4. El actor modifica las Asignaturas que el Profesor dicta y presiona el botón Guardar

5. El sistema modifica los datos

F19.3: Eliminar Asignación de Profesores por Asignatura

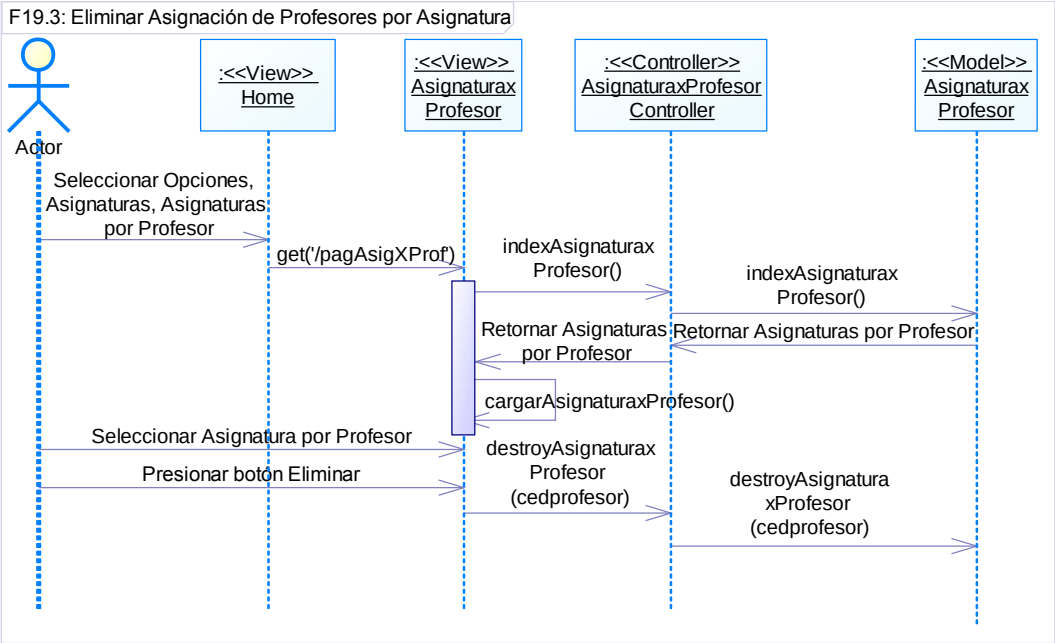


Figura 159. Diagrama Secuencia F19.3: Eliminar Asignación de Profesores por Asignatura

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Asignaturas y en Asignaturas por Profesor
2. El sistema despliega la ventana de AsignaturaxProfesor
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Asignaturas por Profesor ingresados previamente
4. El actor presiona el botón de basurero (Eliminar) en el registro que desee eliminar
5. El sistema elimina el registro seleccionado

F19.4: Consultar Asignación de Profesores por Asignatura General

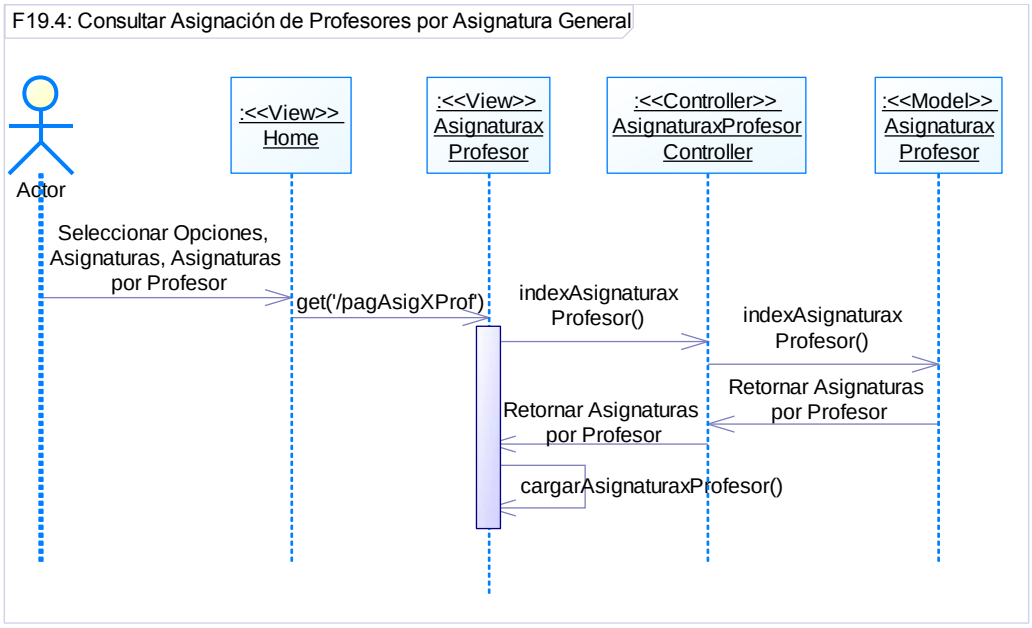


Figura 160. Diagrama Secuencia F19.4: Consultar Asignación de Profesores por Asignatura General

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Asignaturas y en Asignaturas por Profesor
2. El sistema despliega la ventana de AsignaturaxProfesor
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Asignaturas por Profesor ingresados previamente

F19.4.1: Consultar Asignación de Profesores por Asignatura por Parámetros

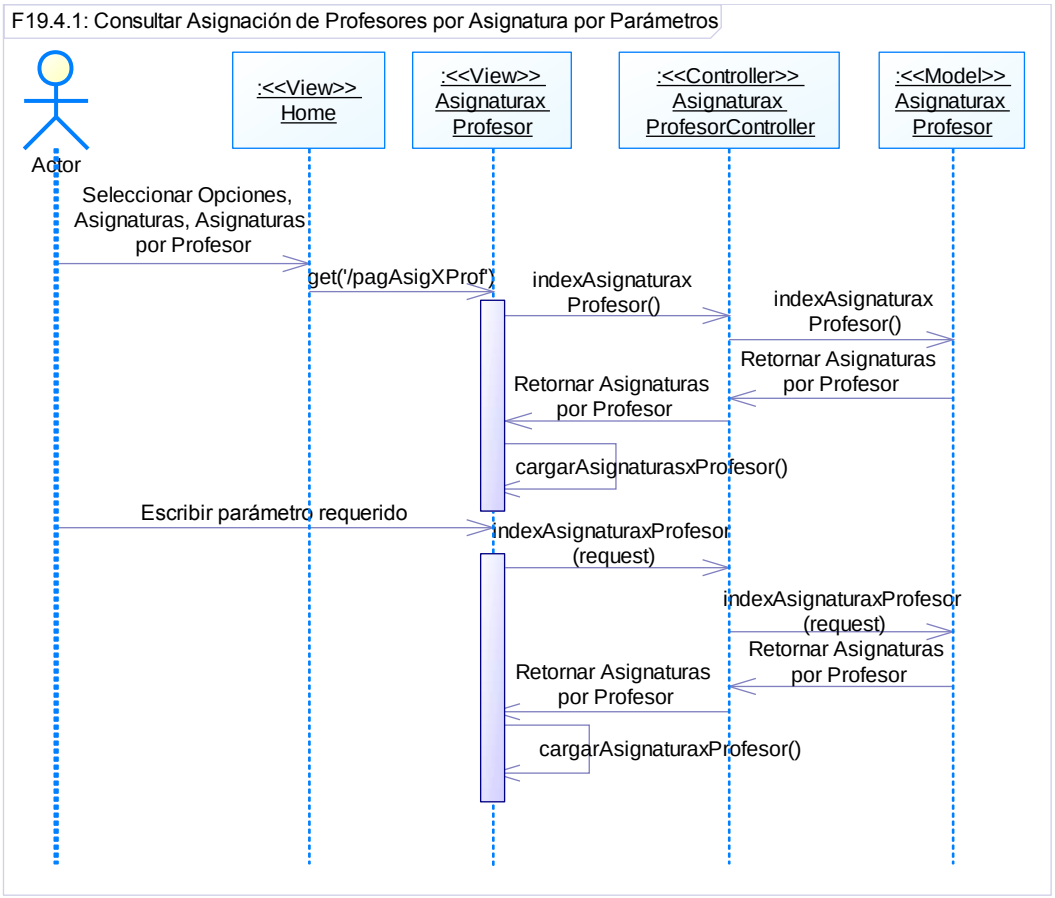


Figura 161. Diagrama Secuencia F19.4.1: Consultar Asignación de Profesores por Asignatura por Parámetros

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Asignaturas y en Asignaturas por Profesor
2. El sistema despliega la ventana de AsignaturaxProfesor
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Asignaturas por Profesor ingresados previamente
4. El actor escribe la palabra que desea en la barra de búsqueda
5. El sistema presenta dentro de la tabla del punto 3 todos los registros que cumplen con el parámetro ingresado por el actor

F20: Gestionar Asignación de Estudiantes por Asignatura

F20.1: Ingresar Asignación de Estudiantes por Asignatura

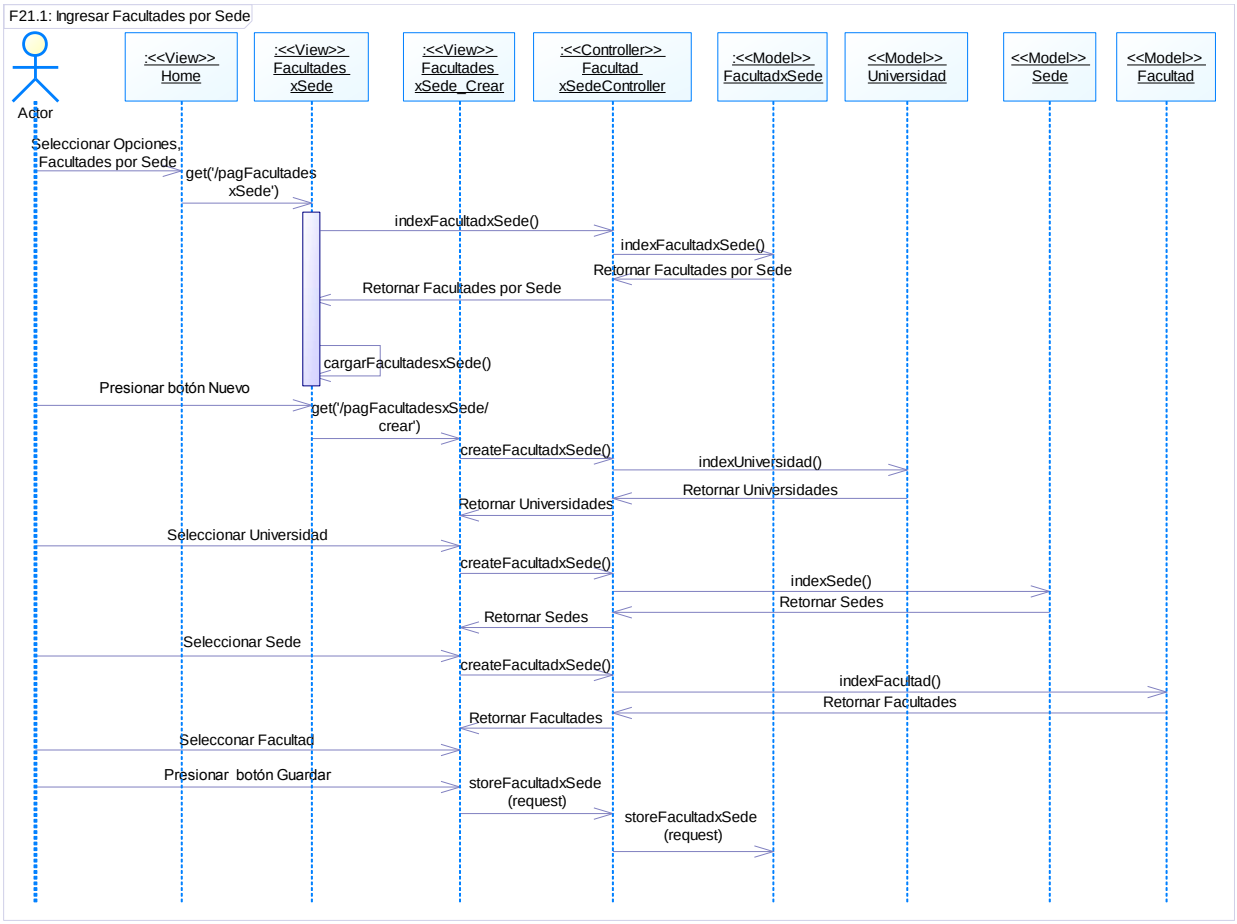


Figura 162. Diagrama Secuencia F20.1: Ingresar Asignación de Estudiantes por Asignatura

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Asignaturas y en Asignaturas por Profesor
2. El sistema despliega la ventana de AsignaturaxProfesor
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Asignaturas por Profesor ingresados previamente
4. El actor presiona el botón Nuevo

5. El sistema dentro de la ventana AsignaturaxProfesor Crear, despliega el Periodo actual
6. El actor ingresa la cédula del Profesor
7. El sistema carga el nombre del Profesor
8. El actor presiona el botón de Nuevo
9. El sistema agrega una fila y despliega las Asignaturas que el Profesor puede dictar
10. El actor elige la Asignatura que desea y presiona el botón Guardar
11. El sistema almacena los datos

F20.2: Modificar Asignación de Estudiantes por Asignatura

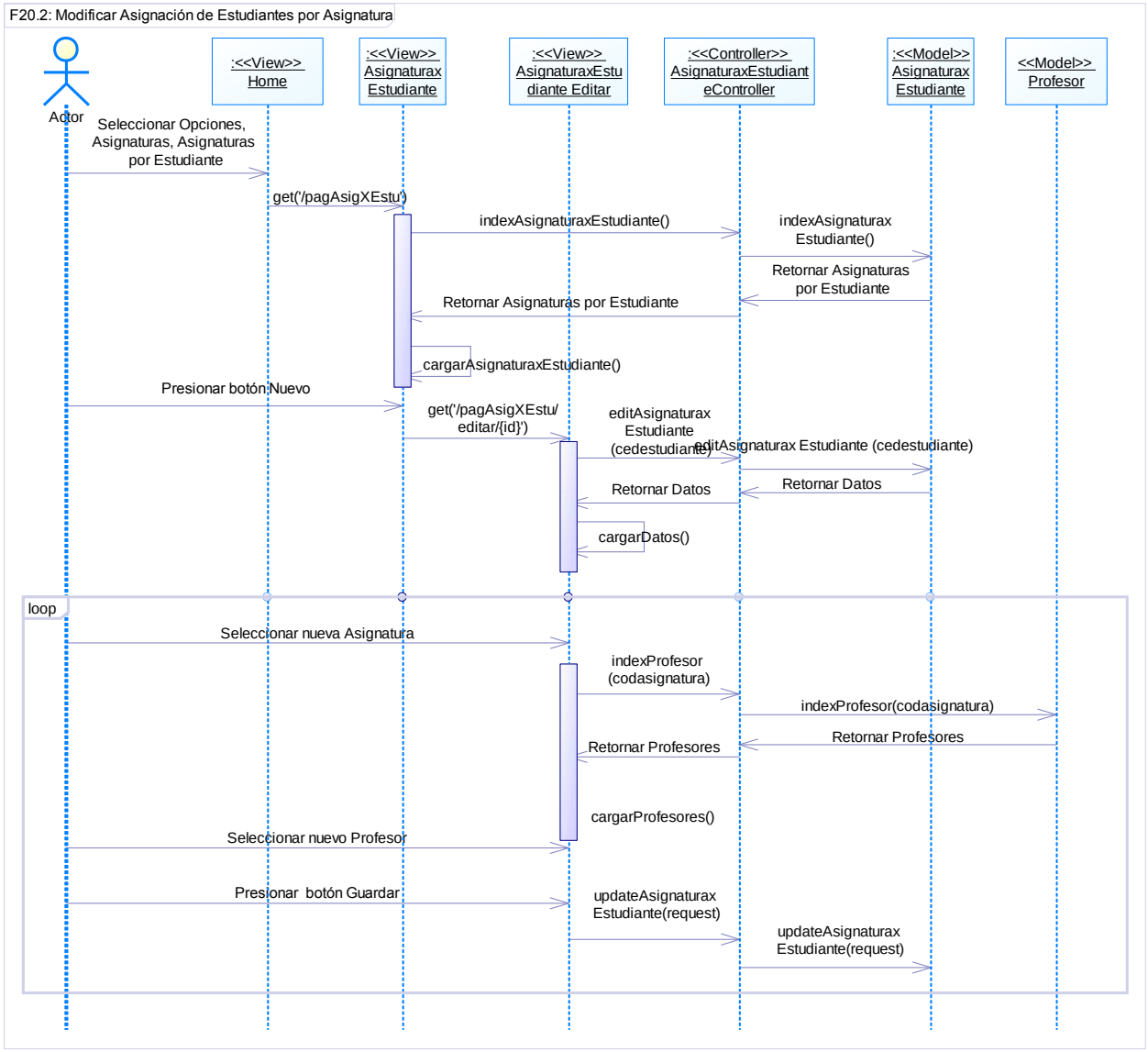


Figura 163. Diagrama Secuencia F20.2: Modificar Asignación de Estudiantes por Asignatura

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Asignaturas y en Asignaturas por Estudiante
2. El sistema despliega la ventana de AsignaturaxEstudiante
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Asignaturas por Estudiante ingresados previamente
4. El actor presiona el botón de lápiz (Modificar) en el registro que desee modificar
5. El sistema dentro de la ventana AsignaturaxEstudiante Editar, despliega todos los campos llenos y las Asignaturas que recibe el Estudiante en el Periodo activo
6. El actor modifica las Asignaturas que el Estudiante recibe o el Profesor y presiona el botón Guardar
7. El sistema modifica los datos

F20.3: Eliminar Asignación de Estudiantes por Asignatura

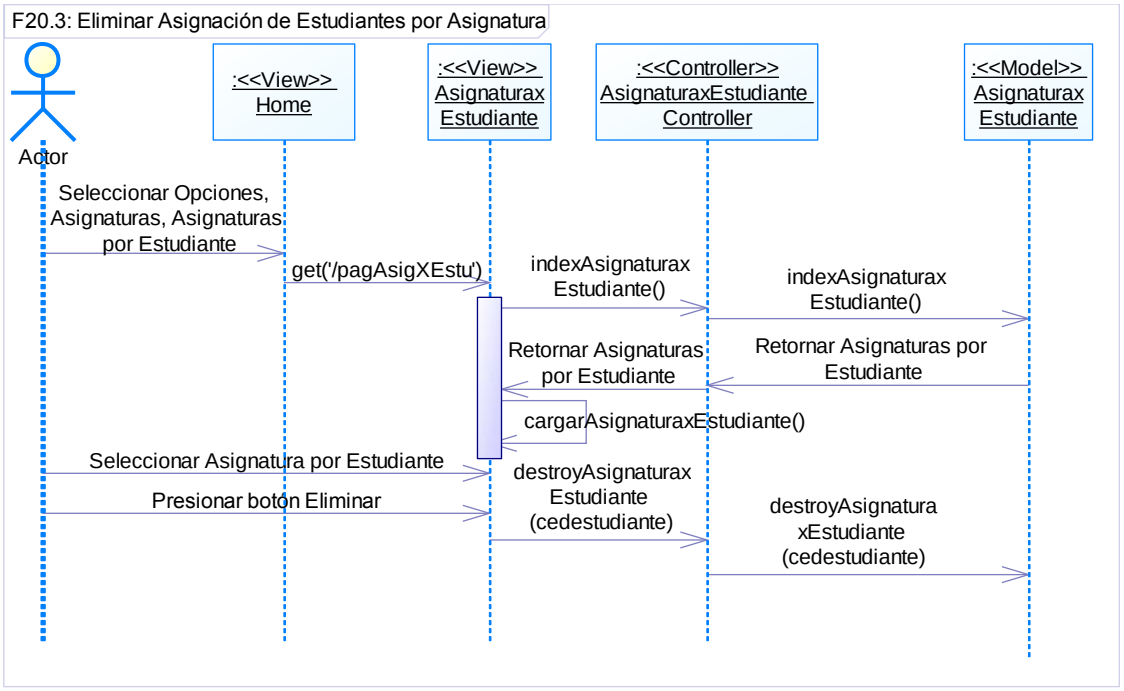


Figura 164. Diagrama Secuencia F20.3: Eliminar Asignación de Estudiantes por Asignatura

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Asignaturas y en Asignaturas por Estudiante
2. El sistema despliega la ventana de AsignaturaxEstudiante
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Asignaturas por Estudiante ingresados previamente
4. El actor presiona el botón de basurero (Eliminar) en el registro que desee eliminar
5. El sistema elimina el registro seleccionado

F20.4: Consultar Asignación de Estudiantes por Asignatura General

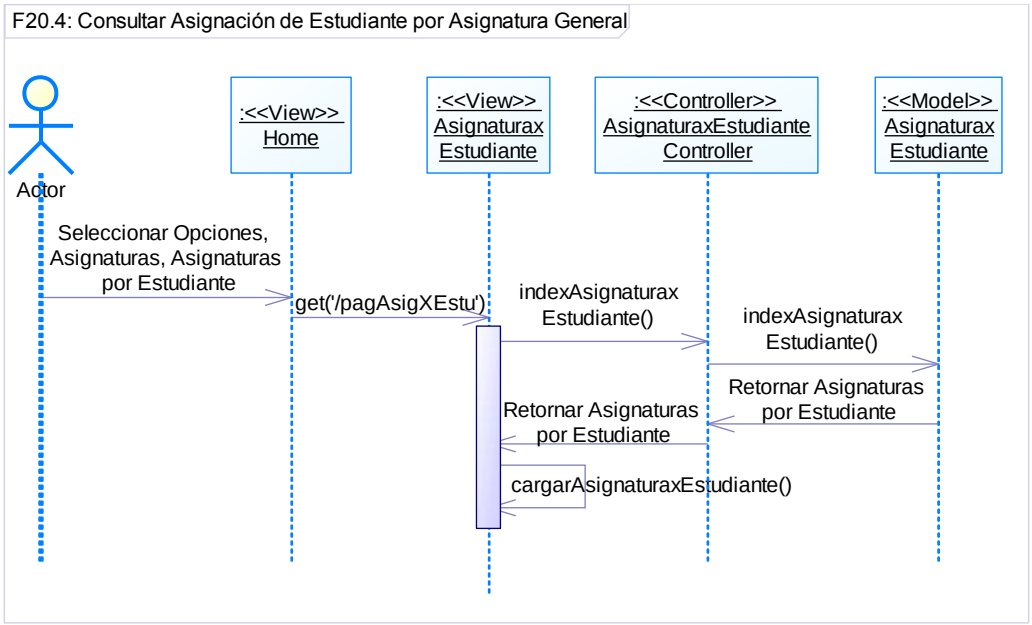


Figura 165. Diagrama Secuencia F20.4: Consultar Asignación de Estudiantes por Asignatura General

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Asignaturas y en Asignaturas por Estudiante
2. El sistema despliega la ventana de AsignaturaxEstudiante

3. El sistema presenta dentro de la ventana una tabla con todos los registros de Asignaturas por Estudiante ingresados previamente

F20.4.1: Consultar Asignación de Estudiantes por Asignatura por Parámetros

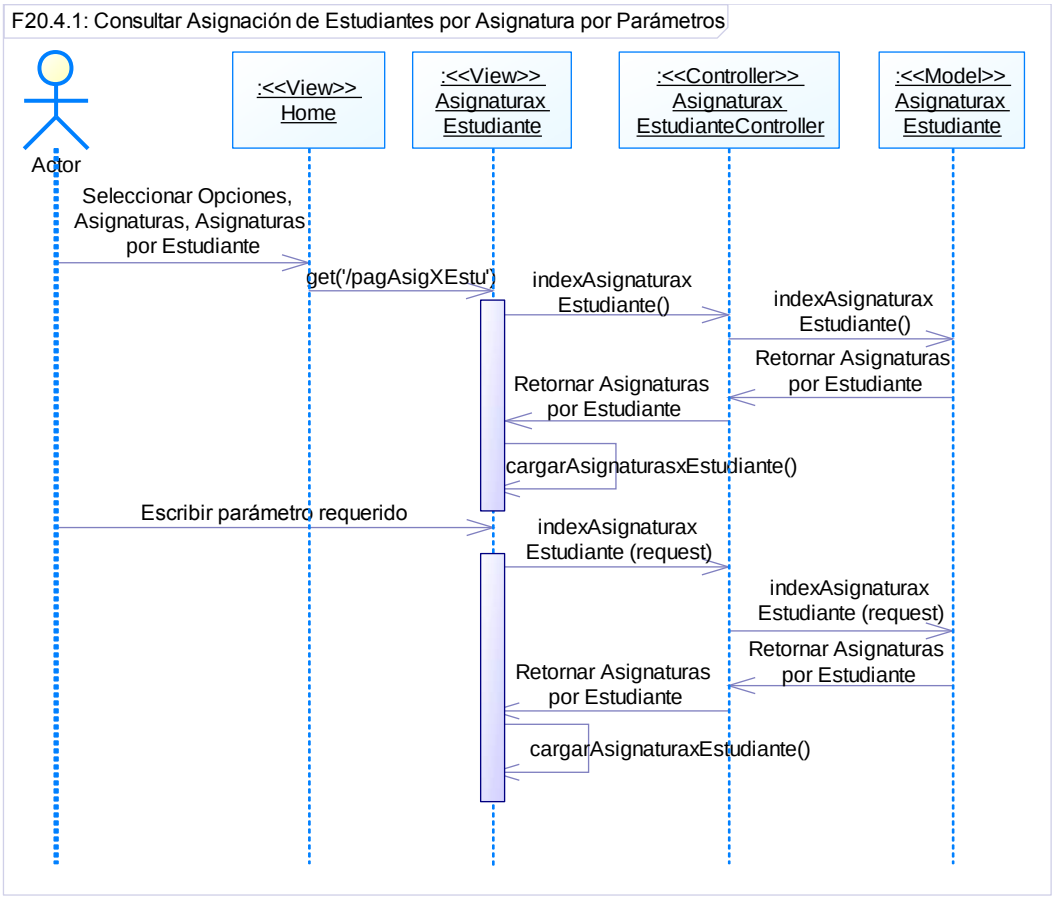


Figura 166. Diagrama Secuencia F20.4.1: Consultar Asignación de Estudiantes por Asignatura por Parámetros

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Asignaturas y en Asignaturas por Estudiante
2. El sistema despliega la ventana de AsignaturaxEstudiante
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Asignaturas por Estudiante ingresados previamente
4. El actor escribe la palabra que desea en la barra de búsqueda

5. El sistema presenta dentro de la tabla del punto 3 todos los registros que cumplen con el parámetro ingresado por el actor

F21: Gestionar Asignación de Facultades por Sede

F21.1: Ingresar Asignación de Facultades por Sede

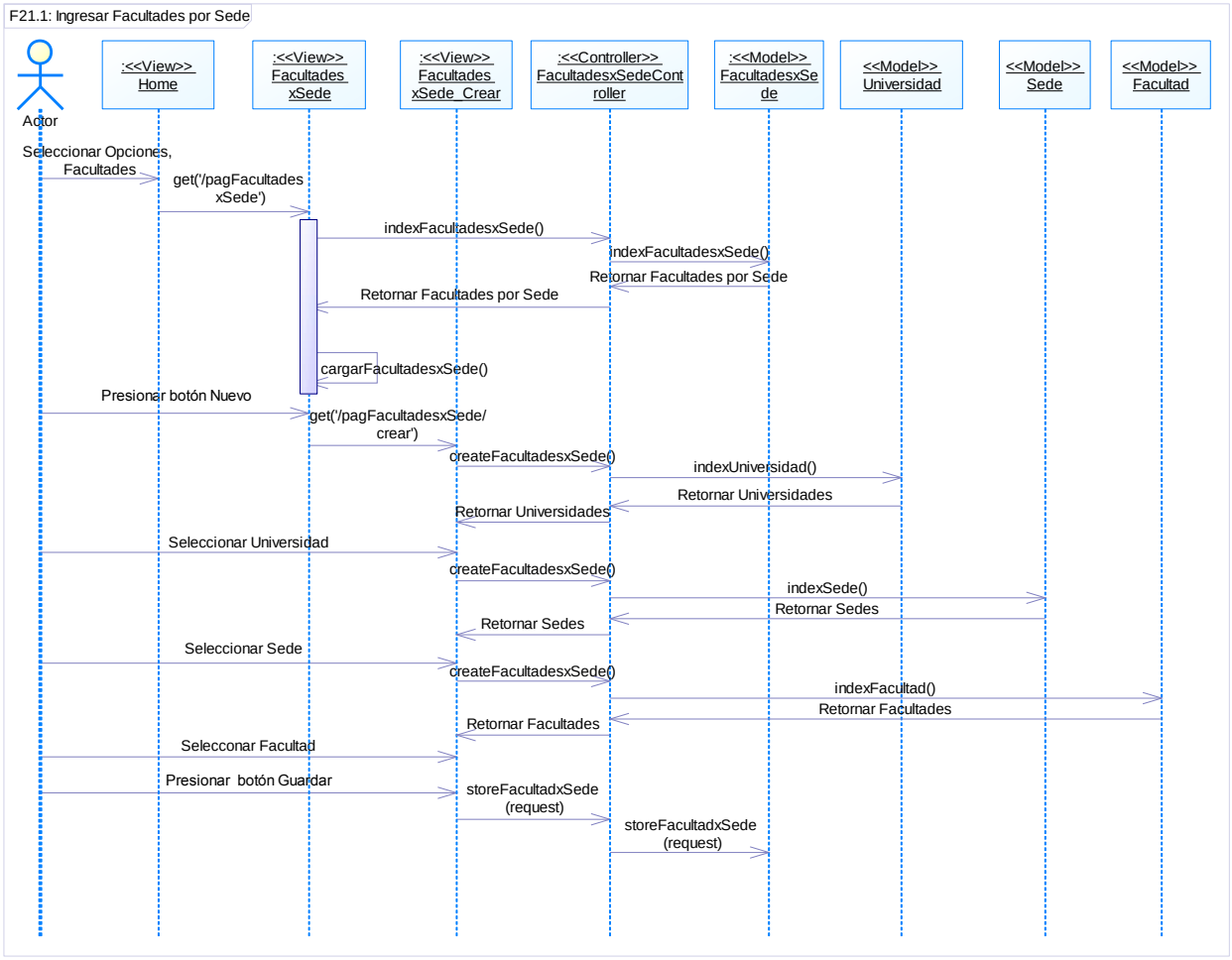


Figura 167. Diagrama Secuencia F21.1: Ingresar Asignación de Facultades por Sede

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Facultades por Sede
2. El sistema despliega la ventana de FacultadesxSede

3. El sistema presenta dentro de la ventana una tabla con todos los registros de Facultades por Sede ingresados previamente
4. El actor presiona el botón Nuevo
5. El sistema dentro de la ventana FacultadesxSede Crear, despliega Universidades
6. El actor selecciona una Universidad
7. El sistema despliega Sedes
8. El actor selecciona una Sede
9. El sistema despliega una Facultad
10. El actor selecciona una Facultad y presiona el botón Guardar
11. El sistema almacena los datos

F21.2: Eliminar Asignación de Facultades por Sede

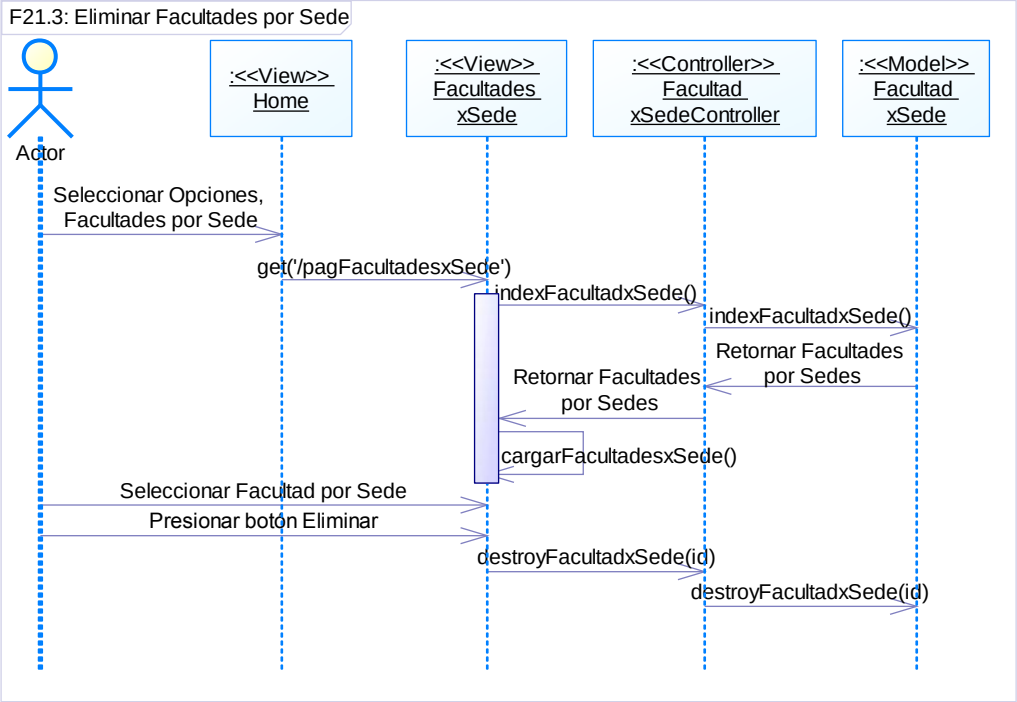


Figura 168. Diagrama Secuencia F21.2: Eliminar Asignación de Facultades por Sede

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Facultades por Sede
2. El sistema despliega la ventana de FacultadesxSede

3. El sistema presenta dentro de la ventana una tabla con todos los registros de Facultades por Sede ingresados previamente
4. El actor presiona el botón de basurero (Eliminar) en el registro que desee eliminar
5. El sistema elimina el registro seleccionado

F21.3: Consultar Asignación de Facultades por Sede General

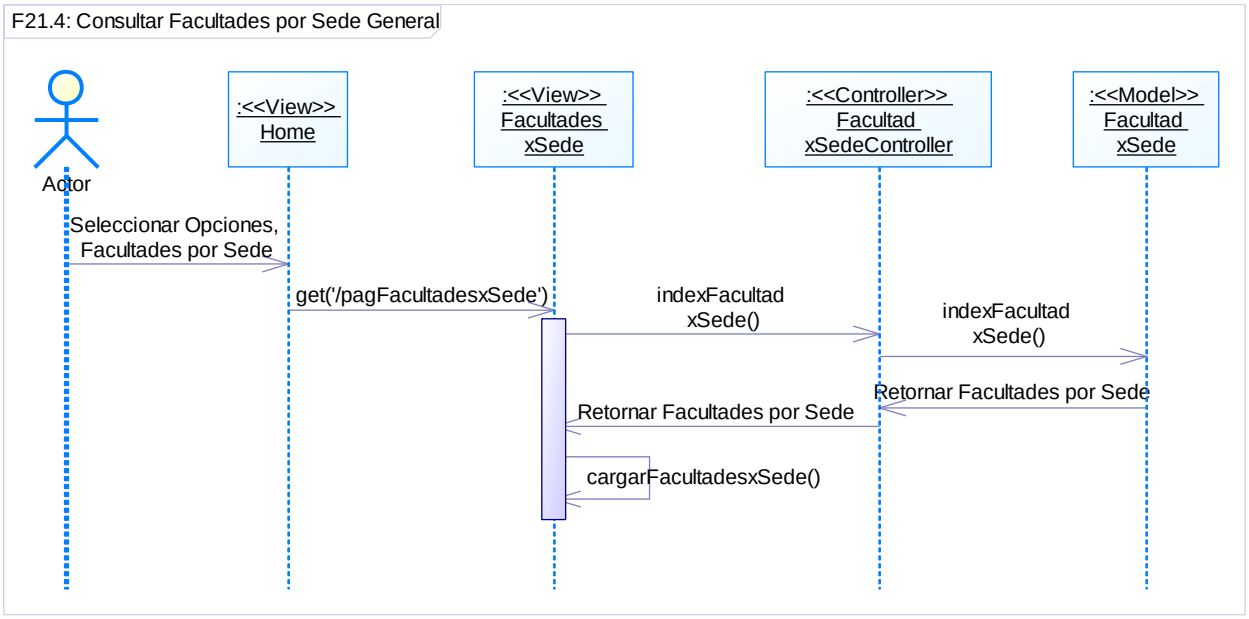


Figura 169. Diagrama Secuencia F21.3: Consultar Asignación de Facultades por Sede General

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Facultades por Sede
2. El sistema despliega la ventana de FacultadesxSede
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Facultades por Sede ingresados previamente

F21.3.1: Consultar Asignación de Facultades por Sede por Parámetros

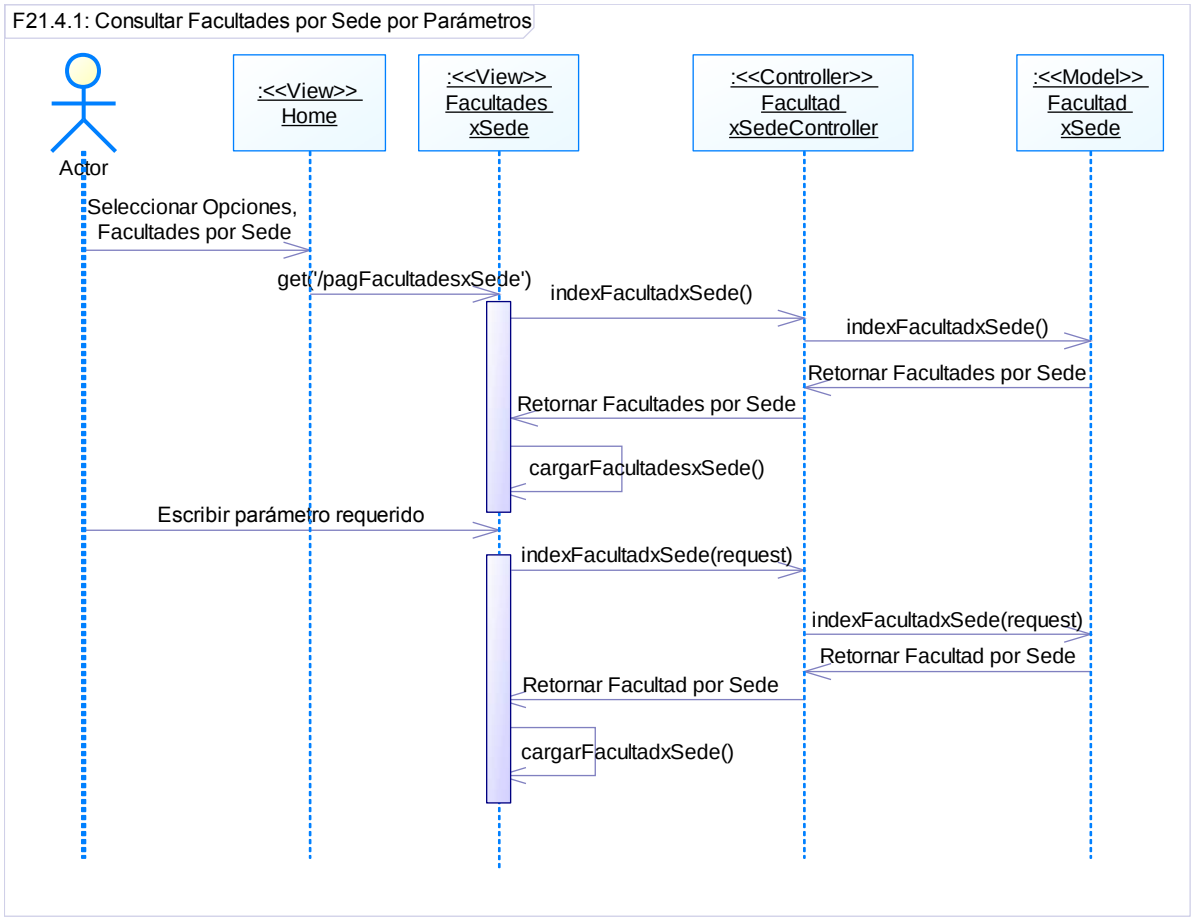


Figura 170. Diagrama Secuencia F21.3.1: Consultar Asignación de Facultades por Sede por Parámetros

Actores

1. Administrador

Flujo Principal

1. El actor selecciona Opciones, Facultades por Sede
2. El sistema despliega la ventana de FacultadesxSede
3. El sistema presenta dentro de la ventana una tabla con todos los registros de Facultades por Sede ingresados previamente
4. El actor escribe la palabra que desea en la barra de búsqueda
5. El sistema presenta dentro de la tabla del punto 3 todos los registros que cumplen con el parámetro ingresado por el actor

F22: Gestionar Estudiante Excel

F22.1: Ingresar Estudiante

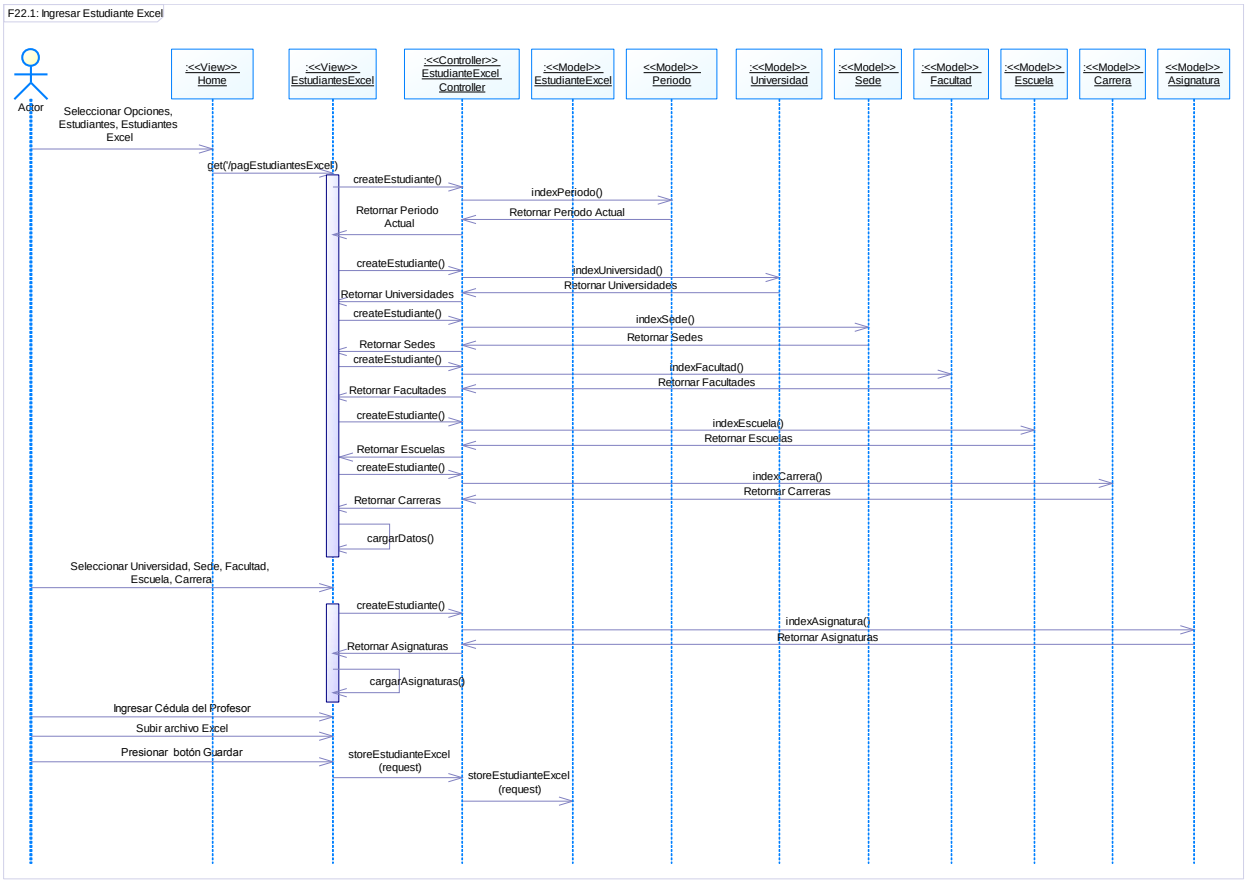


Figura 171. Diagrama Secuencia F22.1: Ingresar Estudiante

Actores

1. Profesor

Flujo Principal

1. El actor selecciona Opciones, Estudiantes y en Estudiantes Excel
2. El sistema despliega la ventana de EstudiantesExcel y dentro de la ventana presenta el Periodo actual y Universidad, Sede, Facultad, Escuela, Carrera
3. El actor selecciona la Universidad, Sede, Facultad, Escuela, Carrera
4. El sistema carga las Asignaturas en base a lo seleccionado anteriormente por el actor

5. El actor selecciona la Asignatura, ingresa la cédula del Profesor, sube el archivo de Excel con los datos de los Estudiantes y presiona el botón Guardar
6. El sistema almacena los datos

Capítulo 3

Implementación y Pruebas

Implementación

Código

En el documento solo se resaltan pequeños fragmentos de código, con las partes más relevantes del sistema, que resumen el comportamiento de múltiples módulos. El resto del código se encontrará en el CD entregable, dentro de la carpeta Implementación.

F3: Módulo Universidades

La Gestión de Universidades en el sistema permite manejar diversos establecimientos educativos que albergan Profesores y Estudiantes. Por lo tanto, existe una estructura académica definida llamada Syllabus que permite manejar el Contenido de diversas Asignaturas repartidas a distintos Profesores; en las cuales un Estudiante se inscribe.

Controlador de Universidades:

El Controlador de Universidades es el lugar que permite definir los métodos en la capa lógica (DP) que utilizan las peticiones del archivo routes.php→web.php, para

realizar acciones dentro de una vista con respecto al ingreso, edición, borrado y consulta de información con respecto a Universidades.

Index:

Método que permite cargar información de Universidades en la tabla tipo CRUD. Además de enviar un arreglo de Universidades para cargar información en <input> y <label> tags.

```
public function index(Request $request)
{
 if($request->get('universidad')!= ""){
    $universidades = DB::table('universidades')->where('universidades.descuniversidad','LIKE','%'.$request->get('universidad')
    return view('universidades',compact('universidades'));
 }
 else{
    $universidades = DB::table('universidades')->paginate();
    return view('universidades',compact('universidades'));
 }
}
```

Figura 172. Método Index Universidades

Create:

Método que permite abrir y redirigir a la Vista Ingresar Universidades.

```
public function create()
{
    return view('universidades_crear');
}
```

Figura 173. Método Create Universidades

Store:

Método que permite almacenar un nuevo registro de Universidades dentro de la base de datos del sistema.

```
public function store(Request $request)
{
    $ultimo = Universidades::all()->sortBy('coduniversidad')->last();
    $cod = substr($ultimo->coduniversidad,4,6);
    if(intval($cod)<10)
    {
        $cod1 = substr($ultimo->coduniversidad,0,5);
        $cod2 = substr($ultimo->coduniversidad,5,6);
        $codfin = $cod1.strval(intval($cod2)+1);

    }
    else
    {
        $cod1 = substr($ultimo->coduniversidad,0,4);
        $cod2 = substr($ultimo->coduniversidad,4,6);
        $codfin = $cod1.strval(intval($cod2)+1);
    }


       Universidades::create([
       'coduniversidad'=>$codfin,
       'descuniversidad'=>$request->input('descuniversidad'),
       'categuniversidad'=>$request->input('categuniversidad'),
       'dir1universidad'=>$request->input('dir1universidad'),
       'dir2universidad'=>$request->input('dir2universidad'),
       'numdiruniversidad'=>$request->input('numdiruniversidad'),
       'tipouniversidad'=>$request->input('tipouniversidad'),
       'rectuniversidad'=>$request->input('rectuniversidad'),
       'viserecuniversidad'=>$request->input('viserecuniversidad'),
       'secregenuniversidad'=>$request->input('secregenuniversidad'),
       'rucuniversidad'=>$request->input('rucuniversidad')

      ]);
      return redirect('pagUniversidades');


  }
```

Figura 174. Método Store Universidades

Destroy:

Método que permite eliminar un registro existente de la tabla Universidades en la base de datos del sistema, en base a un código de Universidad recibido por la vista.

```
public function destroy($coduniversidad)
{
    $codigo = universidades::where('coduniversidad', $coduniversidad)->first();

    if ($codigo != null) {
        $codigo->delete();
        return back()->with(['message'=> 'Successfully deleted!!']);
    }
    return back()->with(['message'=> 'Wrong ID!!']);
}
```

Figura 175. Método Destroy Universidades

Edit:

Método que permite abrir y redirigir a la Vista Editar Universidades, que además de enviar el código del registro de Universidades, va a permitir cargar información en los <input> y <label> tags.

```
public function edit($id)
{
    $codigo = universidades::where('coduniversidad', $id)->first();
    return view('universidades_editar', compact('codigo'));
}
```

Figura 176. Método Edit Universidades

Update:

Método que permite actualizar un registro existente de la tabla Universidades en la base de datos del sistema. Si se actualiza imprime un mensaje de registro actualizado de forma exitosa.

```php
public function update(Request $request, $id)
{
    $codigo = universidades::find($id);
    $codigo->descuniversidad = $request->input('descuniversidad');
    $codigo->categuniversidad = $request->input('categuniversidad');
    $codigo->dir1universidad = $request->input('dir1universidad');
    $codigo->dir2universidad = $request->input('dir2universidad');
    $codigo->numdiruniversidad = $request->input('numdiruniversidad');
    $codigo->tipouniversidad = $request->input('tipouniversidad');
    $codigo->rectuniversidad = $request->input('rectuniversidad');
    $codigo->viserecuniversidad = $request->input('viserecuniversidad');
    $codigo->secregenuniversidad = $request->input('secregenuniversidad');
    $codigo->rucuniversidad = $request->input('rucuniversidad');
    $codigo->save();

    return redirect()->action('UniversidadesController@index', ['success' => 'Universidad actualizada']);
}
```

Figura 177. Método Update Universidades

Modelo de Universidades:

El Modelo de Universidades permite enlazar la capa de manejo de datos (MD) para recibir y enviar información con respecto a la tabla de la base de datos llamada Universidades que se encuentra en PostgreSQL; la base de datos total del sistema se encuentra albergada en un gestor para el manejo de base de datos, que en este caso es pgAdmin 4.

El Modelo de Universidades se lo utiliza dentro del Controlador de Universidades para llamar al ORM por defecto de Laravel llamado Eloquent, que interpreta procesos de bases de datos escritos con instrucciones php; los cuales posteriormente son traducidos a sentencias SQL que realizan operaciones dentro de la tabla Universidades en la base de datos.

Es importante recordar que, para construir el Modelo de Universidades, se debe tomar en cuenta el nombre de los atributos como se encuentran en la tabla Universidades

de la base de datos y que se debe especificar la/las claves primarias que identifican a la tabla Universidades

```
class Universidades extends Model
{
    protected $table = 'universidades';
    protected $primaryKey = 'coduniversidad';
    public $timestamps = false;
    protected $fillable = ['coduniversidad','descuniversidad','categuniversidad','dir1universidad','dir2universidad',
    'numdiruniversidad','tipouniversidad','rectuniversidad','viserecuniviersidad','secregenuniversidad','rucuniversidad'];
    public $incrementing = false;
}
```

Figura 178. Universidades Model

Rutas de Universidad:

En el archivo de Rutas de Universidades se encuentran las direcciones que permiten conectar la capa lógica (DP) con las vistas (GUI) de las Universidades dentro del sistema.

```
//UNIVERSIDADES
Route::get('/pagUniversidades','UniversidadesController@index');
Route::get('/pagUniversidades/crear','UniversidadesController@create');
Route::post('/pagUniversidades/crear','UniversidadesController@store');
Route::get('/pagUniversidades/editar/{coduniversidad}','UniversidadesController@edit');
Route::post('/pagUniversidades/editar/{coduniversidad}','UniversidadesController@update');
Route::post('/pagUniversidades/{coduniversidad}','UniversidadesController@destroy');
```

Figura 179. Universidades Route Path

Vista Principal de Universidades:

La Vista Principal de Universidades permite mostrar todos los datos que se encuentran almacenados en la tabla Universidades de la Base de Datos del sistema. Los datos del registro se reflejan en una tabla tipo CRUD que permite seleccionar el tipo de operación que se quiere realizar con respecto a los datos (insertar, borrar y actualizar). Las operaciones se ven reflejadas en forma de botones.

```
<div class="row"style="    margin-bottom: 1%;  margin-top: 1%; margin-left: 1%">
    <div class="col-sm-3">
    <form  action="/pagUniversidades" method="GET" class="navbar-form navbar-left" role="search">
        <div class= "input-group">
            <input type="text" class="form-control"name="universidad" id="universidad"placeholder="Buscar Universidad"/>
            <span class="input-group-btn">
                <button type="submit" class="btn btn-info">
                    Buscar
                </button>
            </span>
        </div>
    </form>
```

Figura 180. Universidades Blade 1

Como se puede observar, es necesario crear una tabla que sirva como almacén para los registros recuperados de la tabla Universidades que se encuentra en la base de datos del sistema. Sin embargo, se debe usar un tag <form> con método get para poder recuperar la información en un solo sentido y se debe especificar el url exacto de la ruta definida en el archivo web.php.

```
    <td><a href="pagUniversidades/editar/{{$universidad->coduniversidad}}"><button class='btn btn-warning-glyphicon glyphicon-pencil'></button></a></td>
    <td>{{$universidad->descuniversidad}}</td>
    <td>{{$universidad->categuniversidad}}</td>
    <td>{{$universidad->dir1universidad}}</td>
    <td>{{$universidad->dir2universidad}}</td>
    <td>{{$universidad->numdiruniversidad}}</td>
    <td>{{$universidad->tipouniversidad}}</td>
    <td>{{$universidad->rectuniversidad}}</td>
    <td>{{$universidad->viserecuniversidad}}</td>
    <td>{{$universidad->secregenuniversidad}}</td>
    <td>{{$universidad->rucuniversidad}}</td>
</tr>
@endforeach
```

Figura 181. Universidades Blade 2

Es importante definir las columnas <td> en base al nombre declarado en la función Index del controlador. Ya que caso contrario no se va a poder recuperar los datos y cargarlos en las celdas de la tabla.

```
<script>
    $(document).ready(function () {
        $('#myTable').DataTable({
            "language": {
                "url": "/json/Spanish.json"
            }
        });
    });
    function ConfirmDelete()
    {

    var x = confirm("¿Esta seguro que desea eliminar esta universidad?");
    if (x)
        return true;
    else
        return false;
    }
</script>
```

Figura 182. Universidades Blade 3

La librería Data Tables se utiliza para poder cargar de forma dinámica y con posibilidad de obtener un campo de búsqueda por columna y coincidencia los datos de una consulta SQL en la base de datos. Aquí se considera el tipo de notación JSON para indexar campos y así realizar consultas que reordenen la tabla por id.

Vista Ingresar Universidades:

La Vista Ingresar Universidades permite ingresar un nuevo registro en la tabla Universidades de la Base de Datos del sistema.

```
<form method="post" action = "/pagUniversidades/crear">
    {{ csrf_field() }}
    <div class="form-group row">
        <label for="sedeNombre" class="col-sm-2 col-form-label">Descripción<span style="color:red;font-weight:bold">*</span></labe
        <div class="col-sm-5">
        <input type="text" class="form-control" name="descuniversidad" id="descuniversidad" minlength="3"required>
        </div>
    </div>
    <div class="form-group row">
        <label for="sedeNombre" class="col-sm-2 col-form-label">Categoria<span style="color:red;font-weight:bold">*</span></label>
        <div class="col-sm-1">
            <select name="categuniversidad" class="form-control" type="text">
                <option value=""selected></option>
                <option value="A">A</option>
                <option value="B">B</option>
                <option value="C">C</option>
                <option value="D">D</option>
            </select>
        </div>
    </div>
```

Figura 183. Universidades Create Blade 1

Es necesario colocar la instrucción {{csrf_field()}} que permite denegar la falsificación de petición en sitios cruzados (se origina cuando un sitio realiza peticiones en nombre de un usuario sobre nuestro sitio). Al denegar la petición permite acceder a las rutas declaradas en el web.php con post y get para ingresar, recuperar y alterar información dentro de un Blade al usar los tags <input> , <select>, etc.

```
<div class="form-group row">
    <label for="sedeNombre" class="col-sm-2 col-form-label">Tipo<span style="color:red;font-weight:bold">*</span></l
    <div class="col-sm-1">
        <select name="tipouniversidad" class="form-control" type="text">
            <option value=""selected></option>
            <option value="PR">PR</option>
            <option value="PU">PU</option>
        </select>
    </div>
</div>
<div class="form-group row">
    <label for="sedeNombre" class="col-sm-2 col-form-label">Rector</label>
    <div class="col-sm-3">
        <input type="text" class="form-control" name="rectuniversidad" id="rectuniversidad">
    </div>
</div>
<div class="form-group row">
    <label for="sedeNombre" class="col-sm-2 col-form-label">Vicerrector</label>
    <div class="col-sm-3">
        <input type="text" class="form-control" name="viserecuniversidad" id="viserecuniversidad">
    </div>
</div>
<div class="form-group row">
    <label for="sedeNombre" class="col-sm-2 col-form-label">Secretario General</label>
    <div class="col-sm-3">
        <input type="text" class="form-control" name="secregenuniversidad" id="secregenuniversidad">
    </div>
</div>
<div class="form-group row">
    <label for="sedeNombre" class="col-sm-2 col-form-label">Ruc</label>
    <div class="col-sm-3">
        <input type="number" class="form-control" name="rucuniversidad" id="rucuniversidad" maxlength="13">
    </div>
</div>
```

Figura 184. Universidades Create Blade 2

Finalmente, es importante definir el nombre de cada tag <input> y <select> en base al nombre declarado en la función Index del controlador. Ya que caso contrario no se va a poder recuperar los datos y posteriormente ingresar el registro.

Vista Modificar Universidades:

La Vista Modificar Universidades permite modificar un registro previamente ingresado dentro de la tabla Universidades de la Base de Datos del sistema.

```
{{ csrf_field() }}
<div class="form-group row">
    <label for="sedeNombre" class="col-sm-2 col-form-label">Descripción</label>
    <div class="col-sm-5">
        <input type="text" class="form-control" name="descuniversidad" value="{{$codigo->descuniversidad}}">
    </div>
</div>
```

Figura 185. Universidades Edit Blade 1

Como se pudo observar, es necesario colocar la instrucción {{ csrf_field()}} que permite denegar la falsificación de petición en sitios cruzados (se origina cuando un sitio realiza peticiones en nombre de un usuario sobre nuestro sitio). Al denegar la petición permite acceder a las rutas declaradas en el web.php con post y get para ingresar, recuperar y alterar información dentro de un Blade al usar los tags <input> , <select>, etc.

```
<div class="form-group row">
    <label for="sedeNombre" class="col-sm-2 col-form-label"># Direccion</label>
    <div class="col-sm-3">
        <input type="text" class="form-control" name="numdiruniversidad" id="numdiruniversidad" value="{{$codigo->numdiruniversidad}}" >
    </div>
</div>

<div class="form-group row">
    <label for="sedeNombre" class="col-sm-2 col-form-label">Tipo</label>
    <div class="col-sm-1">
        <select type="text" class="form-control" name="tipouniversidad" id="tipouniversidad">
            <option value="PR"<?php if ($codigo->tipouniversidad == 'PR') echo ' selected="selected"'; ?>>PR</option>
            <option value="PU"<?php if ($codigo->tipouniversidad == 'PU') echo ' selected="selected"'; ?>>PU</option>
        </select>
    </div>
</div>
```

Figura 186. Universidades Edit Blade 2

Es importante definir el nombre de cada tag <input> y <select> en base al nombre declarado en la función Index del controlador. Ya que caso contrario no se va a poder recuperar los datos para cargar los valores del registro previamente ingresado en la tabla CRUD y posteriormente actualizar el registro. No se debe olvidar que la/las claves primarias se mantienen fijas hasta acabar la edición de los campos seleccionables o editables.

Vista Eliminar Universidades:

La Vista Eliminar Universidades permite eliminar un registro previamente ingresado dentro de la tabla Universidades de la Base de Datos del sistema. Esta opción se encuentra disponible directamente desde la tabla CRUD en la Vista Principal de Universidades.

```
<form action="{{action('UniversidadesController@destroy', $universidad->coduniversidad)}}" method="POST" onsubmit="return ConfirmDelete()"
    @csrf
    <input name="_method" type="hidden" value="POST">
    <td><button class='btn btn-danger-glyphicon glyphicon glyphicon-trash'></button></td>
</form>
```

Figura 187. Universidades Blade – Eliminar Registro

Es importante recordar que para borrar un registro se necesita confirmar el proceso de borrado. Para esto se utiliza una propiedad del tag <form> llamada onsubmit, que permite confirmar el proceso de borrado; que por cierto en el caso de Universidades no es lógico, pero impide borrar en cascada si existe clave foránea de la tabla Universidades en otro módulo.

Interfaces Universidades:

Las Interfaces de Universidades se encuentran separadas por cada funcionalidad descrita en las Vistas de Universidades.

Interface Principal de Universidades (Tabla CRUD):

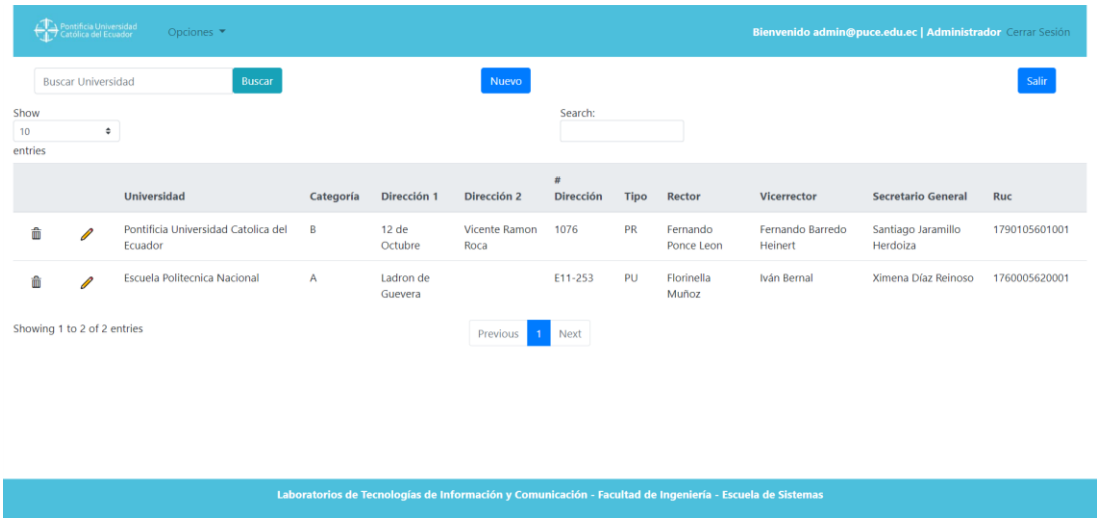


Figura 188. Página de Universidades Principal

Como se puede observar, la Interface Principal de Universidades tiene un diseño de tabla Responsivo que permite manejar las 4 operaciones básicas que componen a un CRUD(crear, borrar, actualizar y consultar). Sin embargo, no se debe olvidar que para alterar e ingresar un nuevo registro se accede a la Interface de Ingreso Universidades y a la Interface Edición Universidades, mediante el Controlador de Universidades.

Interface Ingreso Universidades:

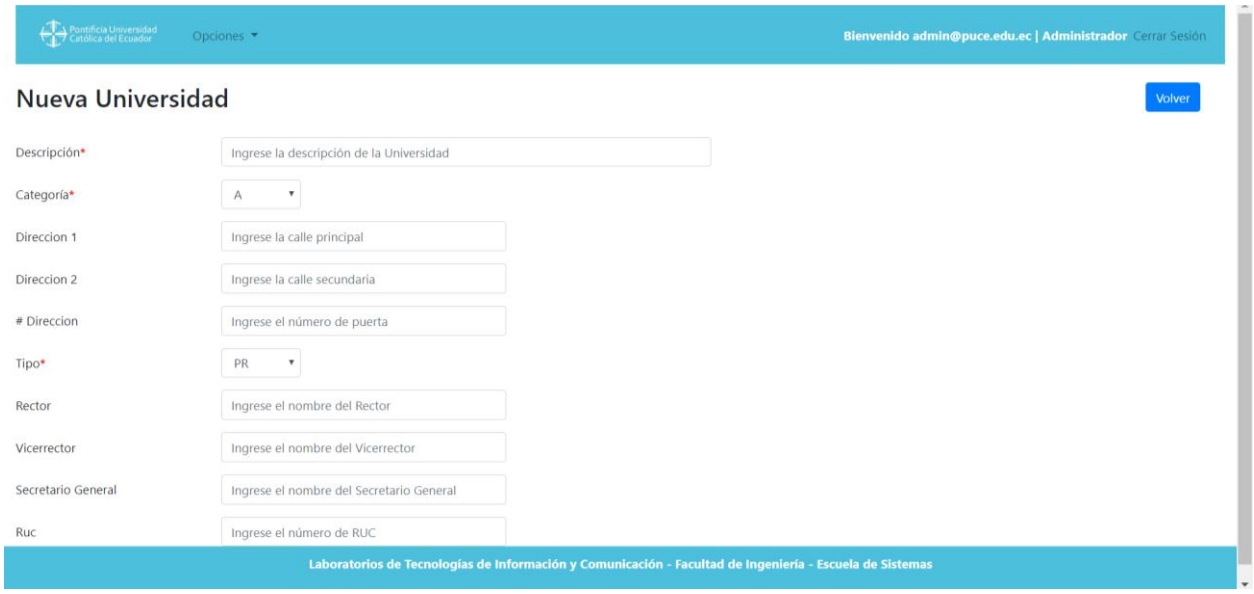


Figura 189. Página de Universidades Crear

Como se puede observar, en la Interface de Ingreso de Universidades se colocan los datos para agregar un nuevo registro en la tabla de Universidades. Sin embargo, se debe tomar en cuenta que los datos obligatorios tienen un asterisco de color rojo, es decir, se debe llenar los campos de forma definitiva. Al presionar guardar se agrega un nuevo registro y por detrás se autogenera un código de Universidad con formato y se coloca en la tabla CRUD.

Interface Edición Universidades:

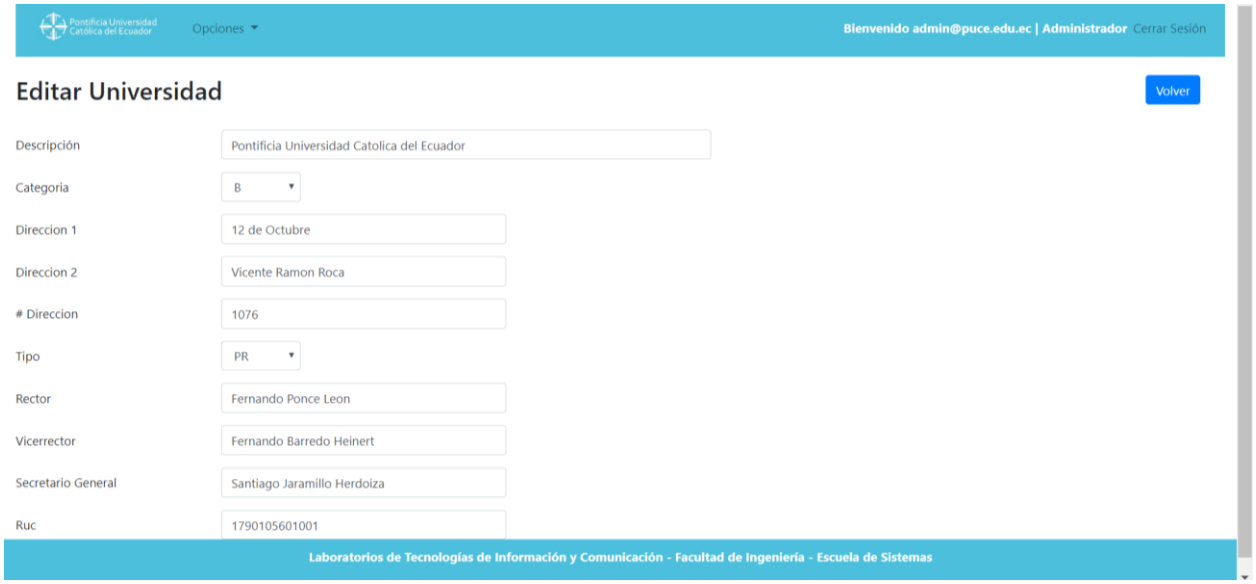


Figura 190. Página de Universidades Editar

Como se puede observar, en la Interface de Edición de Universidades se colocan los datos recuperados de un registro previamente ingresados, para editar y actualizar un registro ya existente de la tabla Universidades. Al presionar guardar se modifica el registro ya en base a un código de Universidad con formato no visible y se coloca en la tabla CRUD.

Requisitos, Relaciones, Procesos de Grabado, Validaciones y Estructuras que se Deben Tomar en Cuenta para la Construcción de Interfaces:

Sistema: Agenda Electrónica

Tabla: UNIVERSIDADES

Relaciones: uni_tiene_sed

Requisitos Previos: Ninguno

Proceso de grabado: Tabla Universidades

Validaciones:

El código de Universidades no se muestra por pantalla. Se lo almacena en la base de datos en base al siguiente formato: 'UNI'+'-'+'número auto incremental' de 2 dígitos, por ejemplo, UNI-00,UNI-01…UNI-NN.

Verificar si código de Universidad ingresado previamente ya existe, en caso de que exista se llenan los demás campos.

Se debe ingresar la categoría en función del CES, descripción, dirección 1, dirección 2, # dirección validando estándar del Municipio, el tipo (PR=Privada, PU=Pública) validando que solo se ingresen 2 caracteres, ingresar nombre del rector, vicerrector, del secretario general y validar que RUC tenga máximo 13 caracteres.

Para borrar una Universidad, debe validar que no exista una Sede relacionada a dicha Universidad.

Botones:

Agrega una nueva fila para ingresar un nuevo registro.

Para confirmar y guardar en la base de datos se presiona el botón de guardar ya sea que se esté ingresando o modificando.



El botón borrar permite eliminar un registro ingresado previamente de acuerdo con las validaciones definidas. Antes de eliminar se muestra un mensaje de confirmación si es que de verdad desea eliminar ese registro.

El botón salir permite regresar a la pantalla principal.



La barra de búsqueda despliega el registro que se encuentra realizando una consulta en base a cualquier atributo de la tabla.

F9: Módulo Gestionar Estudiante

La Gestión de Estudiantes en el sistema permite manejar diversos Estudiantes que conforman una Universidad. El propósito de cada Estudiante es acceder a la Agenda Electrónica creada por uno o varios Profesores con Contenido interactivo, que le permita tomar anotaciones relevantes de cada tema que compone a una materia; para así mejorar su aprendizaje significativo y garantizar un posterior acceso a su material digital de ser necesario.

Controlador de Estudiantes:

El Controlador de Estudiantes es el lugar que permite definir los métodos en la capa lógica (DP) que utilizan las peticiones del archivo routes.php→web.php, para realizar acciones dentro de una vista con respecto al ingreso, edición, borrado y consulta de información con respecto a Estudiantes.

Debido a que el ingreso de Estudiantes se lo puede realizar a través de un archivo de Excel que tenga la lista de Estudiantes pertenecientes a una Asignatura, o de forma manual utilizando formularios; existen 2 métodos de ingreso (Store y EstudianteImport) y 2 métodos de carga de datos (Create y CreateExcel) como se puede observar a continuación:

Index:

Método que permite cargar información de Estudiantes en la tabla tipo CRUD. Además de enviar un arreglo de Estudiantes para cargar información en <input> y <label> tags.

```php
public function index(Request $request)
{
    if($request->get('cedula')!= ""){
        $estudiantes = DB::table('estudiantes')->join('carreras','carreras.codcarrera','=','estudiantes.codcarrera')->where('estudiantes.cedestudiante','LIKE','%'.$request->get('cedula').'%')->paginate(10);
        return view('estudiantes',compact('estudiantes'));
    }
    else{
        $estudiantes = DB::table('estudiantes')->join('carreras','carreras.codcarrera','=','estudiantes.codcarrera')->paginate(10);
        return view('estudiantes',compact('estudiantes'));
    }
}
```

Figura 191. Método Index Estudiantes

EstudianteImport:

```php
public function estudianteImport(Request $request)
{
    try {
    if($request->hasFile('file')){
        $path = $request->file('file')->getRealPath();
        $data = Excel::load($path, function($reader){})->get();
            if (!empty($data) && $data->count())
            {
                $cedprofesor = $request->get('cedprofesor');
                $codperiodo = $request->get('codperiodo');
                $codcarrera = $request->get('unies');

                $codasignatura = $request->get('asignatura');



                foreach($data as $key => $value)
                {
                    $buscarProf = Profesor::where('cedprofesor', $value->cedula)->first();
                    $buscarEst = Estudiante::where('cedestudiante', $value->cedula)->first();
                    if(is_null($buscarProf) && is_null($buscarEst))
                    {
                        $newmail = $value->correo;
                        $existe = User::where('email', $newmail)->first();
                        if (is_null($existe))
                        {
                            //estudiante
                            $cedestudiante = $value->cedula;
                            $nomestudiante=$value->nombre;
                            $apeestudiante=$value->apellido;
                            $correestudiante  = $value->correo;

                            $data2 = array('cedestudiante'=>$cedestudiante,'codcarrera'=>$codcarrera,'nomestudiante'=>$nomestudiante,'apeestudiante'=>$apeestudiante,'correestudiante'=>$correestudiante);
                            DB::table('estudiantes')->insert($data2);
```

Figura 192. Método EstudianteImport Estudiantes 1

La primera parte del método permite enlazar el documento de Excel recuperado de la vista. Posteriormente, comprueba si los datos de las celdas del documento de Excel no se encuentran vacías; de no estar vacías, recupera los datos que se encuentran dentro de las celdas usando un bucle y compara que no se ingrese un registro repetido.

Finalmente, almacena los datos de Estudiantes recuperados en variables y compone un arreglo llamado $data2. Al finalizar, se ejecuta una instrucción SQL de insertar registro en la tabla de Estudiantes.

```
                    $data3 = array('cedestudiante'=>$cedestudiante,'codasignatura'=>$codasignatura,'cedprofesor'=>$cedprofesor,'codperiodo'=>$codperiodo);
                    DB::table('asignaturasxestudiantes')->insert($data3);
                    //usuario
                    $testmail = $value->correo;
                    User::create([
                        'name' => Str::substr($testmail, 0, strrpos($testmail, '@')),
                        'email' => $value->correo,
                        'password' => bcrypt((String)($value->cedula)),
                        'remember_token' => Str::random(5),
                    ]);
                    $id_usuario= User::where('email', $value->correo)->first();
                    $id_usuarioC = $id_usuario->id;
                    //rol
                    DB::table('role_user')->insert([
                        'role_id'=> 3,
                        'user_id'=> (int)$id_usuarioC,
                    ]);
                }
            }
        }
    }
    } catch (\Exception $exception) {
        return back()->withError($exception->getMessage());
    }
    return redirect('pagEstudiantes-excel/create')->with('success', 'Se añadio correctamente');

}
```

Figura 193. Método EstudianteImport Estudiantes 2

La segunda parte del método almacena los datos de Asignaturas con respecto a cada Estudiante recuperados en variables y compone un arreglo llamado $data3. Al finalizar, se ejecuta una instrucción SQL de insertar registro en la tabla de AsignaturasxEstudiantes.

Create:

Método que permite abrir y redirigir a la Vista Ingresar Estudiantes. Además de enviar un arreglo de Carreras para cargar información en <input> y <label> tags.

```
public function create()
{
    $carreras = DB::select('SELECT carreras.codcarrera, carreras.desccarrera, escuelas.descescuela, facultades.descfacultad, sedes.descsede, universidades.descuniversidad
                FROM carreras natural join escuelas natural join facultades natural join facultadesxsedes natural join sedes natural join universidades;');
    return view('estudiantes_crear', compact('carreras'));
}
```

Figura 194. Método Create Estudiantes

CreateExcel:

Método que permite abrir y redirigir a la Vista Ingresar EstudiantesExcel. Además de enviar un arreglo de períodos, Escuelas, Carreras y Asignaturas para cargar información en <input> y <label> tags.

```
public function create()
{
    try {
    $carreras = DB::select('SELECT carreras.codcarrera, carreras.desccarrera, escuelas.descescuela, facultades.descfacultad, sedes.descsede, universidades.descuniversidad
                FROM carreras natural join escuelas natural join facultades natural join facultadesxsedes natural join sedes natural join universidades;');
    $datos = DB::table('escuelas')
    ->join('facultadesxsedes', 'escuelas.codfacultad', '=', 'facultadesxsedes.codfacultad')
    ->join('sedes', 'facultadesxsedes.codsede', '=', 'sedes.codsede')
    ->join('universidades', 'sedes.coduniversidad', '=', 'universidades.coduniversidad')
    ->join('carreras', 'escuelas.codescuela', '=', 'carreras.codescuela')
    ->join('asignaturas', 'carreras.codcarrera', '=', 'asignaturas.codcarrera')
    ->join('periodos', 'periodos.codsede', '=', 'sedes.codsede')
    ->where('periodos.estperiodo', '=', 'A')
    ->get();

    $asignaturas = DB::table('asignaturas')->get();
    $periodos = DB::table('periodos')->where('periodos.estperiodo', '=', 'A')->get();
    } catch (\Exception $exception) {
        return back()->withError($exception->getMessage());
    }
    return view('EstudiantesExcel',['periodos'=>$periodos,'datos'=>$datos,'carreras'=>$carreras,'asignaturas'=>$asignaturas]);
}
```

Figura 195. Método Create Estudiantes Excel

Store:

Método que permite almacenar un nuevo registro de Estudiantes dentro de la base de datos del sistema mediante el uso de un formulario dentro de la vista Crear Estudiantes.

```php
public function store(Request $request)
{
    $buscarProf = Profesor::where('cedprofesor', $request->input('cedestudiante'))->first();
    $buscarEst = Estudiante::where('cedestudiante', $request->input('cedestudiante'))->first();
    if(is_null($buscarProf) && is_null($buscarEst))
    {
        $newmail = $request->input('correestudiante');
        $existe = User::where('email', $newmail)->first();
        if (is_null($existe))
        {
            //estudiante
            $estudiante = new Estudiante();
            $estudiante->cedestudiante = $request->input('cedestudiante');
            $estudiante->codcarrera = $request->input('codcarrera');
            $estudiante->nomestudiante = $request->input('nomestudiante');
            $estudiante->apeestudiante = $request->input('apeestudiante');
            $estudiante->correestudiante = $request->input('correestudiante');
            $estudiante->save();
            //usuario
            $testmail=$request->input('correestudiante');
            User::create([
                'name' => Str::substr($testmail, 0, strrpos($testmail, '@')),
                'email' => $request->input('correestudiante'),
                'password' => bcrypt((String)($request->input('cedestudiante'))),
                'remember_token' => Str::random(5),
            ]);
            $id_usuario= User::where('email', $request->input('correestudiante'))->first();
            $id_usuarioC = $id_usuario->id;
            //rol
            DB::table('role_user')->insert([
                'role_id'=> 3,
                'user_id'=> (int)$id_usuarioC,
            ]);
            return redirect('pagEstudiantes');
        }
        else
        {
            return redirect()->back() ->with(['error' => 'No se puede ingresar el estudiante. El correo que esta ingresando ya lo tiene registrado otro usuario']);
        }
```

Figura 196. Método Store Estudiantes

Este método particular permite no solo controlar el ingreso de los Estudiantes en la base de datos del sistema, sino que también permite manejar el ingreso de nuevos usuarios con rol de Estudiante dentro de la tabla Usuarios y Roles en la base de datos del sistema.

Para esto primero verifica que el correo del Usuario en general no exista dentro del sistema, y posteriormente se compara la existencia de las cédulas de identidad para evitar que se ingrese un usuario duplicado. Finalmente, se ingresa un Estudiante y Usuario dentro de sus respectivas tablas y se manda un mensaje a consola que dice registro ingresado correctamente.

Destroy:

Método que permite eliminar un registro existente de la tabla Estudiantes y Usuarios en la base de datos del sistema. Para poder eliminar se recibe la cédula de identidad del Estudiante y si se elimina exitosamente se manda un mensaje a consola que diga registro borrado exitosamente. Caso contrario y si no se puede eliminar debido a que se irrumpe con la integridad referencial de las tablas y sus registros, se manda un mensaje que dice que no se puede eliminar Estudiante.

```
public function destroy($id)
{
    $asigxest = DB::table('asignaturasxestudiantes')->where('cedestudiante', '=', $id)->get();
    if($asigxest->count()==0)
    {
        //estudiante
        $estudiante = Estudiante::find($id);
        $estudiante->delete();
        //usuario
        $usuarioEst= User::where('email', $estudiante->correestudiante)->first();
        $usuarioEst->delete();
        return redirect()->action('EstudianteController@index', ['success' => 'Estudiante eliminado']);
    }
    else{
        return redirect()->action('EstudianteController@index', ['success' => 'Estudiante no se pudo eliminar']);
    }

}
```

Figura 197. Método Destroy Estudiantes

Edit:

Método que permite abrir y redirigir a la Vista Editar Estudiantes, para esto se recibe la cédula del Estudiante, que en base a una consulta SQL, permita recuperar el objeto Estudiante que posteriormente se va a editar de la base de datos.

```php
public function edit($id)
{
    $estudiante = Estudiante::where('cedestudiante', $id)->first();
    return view('estudiantes_editar', compact('estudiante'));
}
```

Figura 198. Método Edit Estudiantes

Update:

Método que permite actualizar un registro existente de la tabla Estudiantes y Usuarios en la base de datos del sistema, para esto se utiliza la cédula de identidad del Estudiante recibida a través de la vista Editar Estudiantes. Si se ingresa un correo ya existente por otro Usuario, se envía un mensaje que dice que no se puede actualizar Estudiante.

```php
public function update(Request $request, $id)
{
    $estudiante = Estudiante::find($id);
    $oldmail = $estudiante->correestudiante;
    $usuarioEst= User::where('email', $oldmail)->first();
    $newmail = $request->input('correestudiante');
    $existe = User::where('email', $newmail)->first();
    if ( is_null($existe) || $existe->email==$oldmail)
    {
        //estudiante table
        $estudiante->nomestudiante = $request->input('nomestudiante');
        $estudiante->apeestudiante = $request->input('apeestudiante');
        $estudiante->correestudiante = $request->input('correestudiante');
        $estudiante->save();
        ///user table
        $usuarioEst->name = Str::substr($newmail, 0, strrpos($newmail, '@'));
        $usuarioEst->email = $newmail;
        $usuarioEst->remember_token = Str::random(5);
        $usuarioEst->save();
        return redirect()->action('EstudianteController@index', ['success' => 'Estudiante actualizado']);
    }
    else
    {
        return redirect()->back()->with(['error' => 'No se puede actualizar el estudiante. El correo que esta ingresando ya lo tiene registrado otro estudiante']);
    }
}
```

Figura 199. Método Update Estudiantes

Modelo de Estudiantes:

El Modelo de Estudiantes permite enlazar la capa de manejo de datos (MD) para recibir y enviar información con respecto a la tabla de la base de datos llamada Estudiantes

que se encuentra en PostgreSQL; la base de datos total del sistema se encuentra albergada en un gestor para el manejo de base de datos, que en este caso es pgAdmin 4.

El Modelo de Estudiantes se lo utiliza dentro del Controlador de Estudiantes y Estudiantes Excel para llamar al ORM por defecto de Laravel llamado Eloquent, que interpreta procesos de bases de datos escritos con instrucciones php; los cuales posteriormente son traducidos a sentencias SQL que realizan operaciones dentro de la tabla Estudiantes en la base de datos.

Es importante recordar que, para construir el Modelo de Estudiantes, se debe tomar en cuenta el nombre de los atributos como se encuentran en la tabla Estudiantes de la base de datos y la clave primaria que identifica a la tabla Estudiantes.

```php
class Estudiante extends Model
{
    protected $table = 'estudiantes';
    protected $primaryKey = 'cedestudiante';
    public $timestamps = false;
    protected $fillable = ['codcarrera','cedestudiante','nomestudiante','apeestudiante','correestudiante'];
}
```

Figura 200. Estudiantes Model

Rutas de Estudiante:

En el archivo de Rutas de Estudiantes se encuentran las direcciones que permiten conectar la capa lógica (DP) con las vistas (GUI) de las Estudiantes dentro del sistema. Cabe recalcar, que existen 2 grupos de rutas para manejar los Estudiantes en Formularios y los Estudiantes en Excel.

```
//ESTUDIANTES
Route::get('/pagEstudiantes','EstudianteController@index');
Route::get('/pagEstudiantes/crear','EstudianteController@create');
Route::post('/pagEstudiantes/crear','EstudianteController@store');
Route::get('/pagEstudiantes/editar/{id}','EstudianteController@edit');
Route::post('/pagEstudiantes/editar/{id}','EstudianteController@update');
Route::post('/pagEstudiantes/{id}','EstudianteController@destroy');
```

Figura 201. Estudiantes Formularios Route Path

```
//ESTUDIANTES EXCEL
Route::view('/pagEstudiantesExcel','estudiantesExcel');
Route::post('import', 'EstudianteExcelController@estudianteImport')->name('estudiante.import');
Route::resource('pagEstudiantes-excel','EstudianteExcelController');
```

Figura 202. Estudiantes en Excel Route Path

Vista Principal de Estudiantes:

La Vista Principal de Estudiantes permite mostrar todos los datos que se encuentran almacenados en la tabla Estudiantes de la Base de Datos del sistema. Los datos del registro se reflejan en una tabla tipo CRUD que permite seleccionar el tipo de operación que se quiere realizar con respecto a los datos (insertar, borrar y actualizar). Las operaciones se ven reflejadas en forma de botones.

```
        <H1>ESTUDIANTES</H1>
    </div>
    <div class="col-md-2"></div>
    <div class="col-md-6">
        <a href="pagEstudiantes/crear"><button type="submit" class="btn btn-primary">Nuevo Estudiante</button></a>
    </div>
    <div class = "col-md-1">
    <a href="/"><button type="submit" class="btn btn-primary">Salir</button></a>
    </div>
</div>
<div class="table-container">
    <table  id="myTable" class="table table-hover "  style="width:100%;margin-top:10px">
    <thead class="container">
        <tr>
            <th></th>
            <th ></th>
            <th >Cédula</th>
            <th >Nombre</th>
            <th >Apellido</th>
            <th >e-mail</th>
            <th>Carrera</th>

        </tr>
    </thead>
```

Figura 203. Estudiantes Blade 1

Como se puede observar, es necesario crear una tabla que sirva como almacén para los registros recuperados de la tabla Estudiantes que se encuentra en la base de datos del sistema. Sin embargo, se debe usar un tag <form> con método get para poder recuperar la información en un solo sentido y se debe especificar el url exacto de la ruta definida en el archivo web.php.

```
    <label name="estudianteci">{{ $estudiante->cedestudiante }}</label>
    </td>

    <td >
        <label name="estudiantenombre">{{ $estudiante->nomestudiante }}</label>
    </td>
    <td >
        <label name="estudianteapellido">{{ $estudiante->apeestudiante }}</label>
    </td>

    <td >
        <label name="estudianteemail">{{ $estudiante->correestudiante }}</label>
    </td>
    <td >
        <label name="estudianteescuela">{{ $estudiante->desccarrera }}</label>
    </td>
</tr>
```

Figura 204. Estudiantes Blade 2

Es importante definir las columnas <td> en base al nombre declarado en la función Index del controlador. Ya que caso contrario no se va a poder recuperar los datos y cargarlos en las celdas de la tabla.

```
$(document).ready(function () {
    $('#myTable').DataTable({
        "paging": false,
        "info": false,
        "autoWidth": true,
        "searching": true, // Search box and search function will be actived
        "dom": '<"top"f>rt<"bottom"ilp><"clear">',
        "language": {
            "zeroRecords": "No existe registros",
            "infoEmpty": "No se econtró ningún registro",
            "sSearch": "Buscar:    "
            }
    });

});
function ConfirmDelete()
  {
  var x = confirm("¿Esta seguro que desea eliminar este estudiante?");
  if (x)
    return true;
  else
    return false;
  }
```

Figura 205. Estudiantes Blade 3

La librería Data Tables se utiliza para poder cargar de forma dinámica y con posibilidad de obtener un campo de búsqueda por columna y coincidencia los datos de una consulta SQL en la base de datos. Aquí se considera el tipo de notación JSON para indexar campos y así realizar consultas que reordenen la tabla por id.

Vista Ingresar Estudiantes:

La Vista Ingresar Estudiantes permite ingresar un nuevo registro en la tabla Estudiantes de la Base de Datos del sistema.

```
<div style="    margin-left: 5%">

    <div class="row">
        <div class="col-sm-11">
            <h2>Nuevo Estudiante</h2>
        </div>
        <div class="col-sm-1">
            <a href="/pagEstudiantes" class="btn btn-primary"> Volver</a>
        </div>
    </div>

    <br>


    <form method="post" action = "/pagEstudiantes/crear">
        {{ csrf_field() }}
        <div class="form-group row">
        <label for="sedeNombre" class="col-sm-2 col-form-label">Universidad - Carrera</label>
        <div  style="width: 38%;">
        <select class="form-control" id="codcarrera" name="codcarrera">
                @foreach ($carreras as $carrera)
+                   <option value="{{$carrera->codcarrera}}">{{$carrera->descuniversidad}} -  {{$carrera->descsede}} -  {{$carrera->descfacultad}} -  {{$carrera->descescuela}}
                    -  {{$carrera->desccarrera}}</option>
+               @endforeach
            </select>
        </div>
```

Figura 206. Estudiantes Create Blade 1

Es necesario colocar la instrucción {{ csrf_field()}} que permite denegar la falsificación de petición en sitios cruzados (se origina cuando un sitio realiza peticiones en nombre de un usuario sobre nuestro sitio). Al denegar la petición permite acceder a las rutas declaradas en el web.php con post y get para ingresar, recuperar y alterar información dentro de un Blade al usar los tags <input> , <select>, etc.

```
</div>
<div class="form-group row">
    <label for="cedestudiante" class="col-sm-2 col-form-label">Cédula</label>
    <div class="col-sm-2">
        <input type="text" class="form-control" name="cedestudiante" id="cedestudiante" maxlength="10"required placeholder="Ingrese su Cédula">
    </div>
</div>
<div class="form-group row">
    <label for="nomestudiante" class="col-sm-2 col-form-label">Nombres</label>
    <div class="col-sm-2">
        <input type="text" class="form-control" name="nomestudiante" id="nomestudiante" required placeholder="Ingrese sus Nombres">
    </div>
</div>
<div class="form-group row">
    <label for="apeestudiante" class="col-sm-2 col-form-label">Apellidos</label>
    <div class="col-sm-2">
        <input type="text" class="form-control" name="apeestudiante" id="apeestudiante" required placeholder="Ingrese sus Apellidos">
    </div>
</div>

<div class="form-group row">
    <label for="correestudiante" class="col-sm-2 col-form-label">Correo</label>
    <div class="col-sm-2">
        <input type="email" pattern=".+@puce.edu.ec"   class="form-control" name="correestudiante" id="correestudiante" placeholder="Correo de la Universidad">
    </div>
</div>
```

Figura 207. Estudiantes Create Blade 2

Es importante definir el nombre de cada tag <input> y <select> en base al nombre declarado en la función Index del controlador. Ya que caso contrario no se va a poder recuperar los datos y posteriormente ingresar el registro.

Vista Ingresar Estudiantes Excel:

La Vista Ingresar Estudiantes Excel permite ingresar registros en la tabla Estudiantes de la Base de Datos del sistema. Sin embargo, requiere de un archivo Excel con la lista de Estudiantes que se encuentran matriculados a una Asignatura por parte del Profesor.

```
<form   method="post" action="{{route('estudiante.import')}}" class="form-horizontal" data-toggle="validator" enctype="multipart/form-data">
@csrf
    <div class="form-horizontal">


    @foreach($datos as $dato)
    <input type="hidden" id="codcarrera" class="form-control" name="codcarrera" value="{{$dato->codcarrera}}">
    <input type="hidden" id="codasignatura" class="form-control" name="codasignatura" value="{{$dato->codasignatura}}">
    @endforeach
```

Figura 208. Estudiantes Create Excel Blade 1

Como se pudo observar, es necesario colocar la instrucción @csrf que permite denegar la falsificación de petición en sitios cruzados (se origina cuando un sitio realiza peticiones en nombre de un usuario sobre nuestro sitio). Al denegar la petición permite acceder a las rutas declaradas en el web.php con post y get para ingresar, recuperar y alterar información dentro de un Blade al usar los tags <input> , <select>, etc.

```
    <select class="form-control" id="unies" name="unies">
    <option>Seleccionar Carrera</option>
@foreach ($carreras as $carrera)
<option value="{{$carrera->codcarrera}}" > {{$carrera->descuniversidad}} - {{$carrera->descsede}} - {{$carrera->descfacultad}} - {{$carrera->descescuela}} - {{$carrera->desccarrera}}</option>
@endforeach
    </select>
        </div>
</div>

<div class="form-group row">
    <label class="control-label col-sm-3" for="asignatura">Asignatura</label>
    <div class="col-sm-10">
    <select class="form-control" id="asignatura" name="asignatura">

    </select>
        </div>
</div>

<div class="form-group row">
    <label class="control-label col-sm-3" for="cedprofesor">Cedula del profesor</label>
    <div class="col-sm-10">
        <input type="text" id="cedprofesor" class="form-control" name="cedprofesor">
        </div>
</div>

<div class="form-group row">
    <label for="file" class="col-md-3 control-label">Archivo</label>
    <div class="col-md-10">
        <input type="file" id="file" name="file" class="form-control" autofocus required>
    <span class="help-block with-errors"></span>
    </div>
</div>
```

Figura 209. Estudiantes Create Excel Blade 2

Finalmente, es importante definir el nombre de cada tag <input> y <select> en base al nombre declarado en la función Index del controlador. Ya que caso contrario no se va a poder recuperar los datos y posteriormente ingresar los registros. Cabe recalcar, que el archivo es cargado usando un <input type=’file’> que permite recuperar el archivo y cargarlo en un componente tipo xls que será enviado al controlador al presionar el botón guardar.

Vista Modificar Estudiantes:

La Vista Modificar Estudiantes permite modificar un registro previamente ingresado dentro de la tabla Estudiantes de la Base de Datos del sistema.

```
<div class="row">
    <div class="col-sm-11">
        <h2>Editar Provincia</h2>
    </div>
    <div class="col-sm-1">
        <a href="/pagEstudiantes" class="btn btn-primary"> Volver</a>
    </div>
</div>
```

Figura 210. Estudiantes Edit Blade 1

Es necesario colocar la instrucción {{ csrf_field()}} que permite denegar la falsificación de petición en sitios cruzados (se origina cuando un sitio realiza peticiones en nombre de un usuario sobre nuestro sitio). Al denegar la petición permite acceder a las rutas declaradas en el web.php con post y get para ingresar, recuperar y alterar información dentro de un Blade al usar los tags <input> , <select>, etc.

```
<form method="post" action = "">
    {{ csrf_field() }}
    <div class="form-group row">
        <label for="cedestudiante" class="col-sm-2 col-form-label">Cédula</label>
        <div class="col-sm-2">
            <input type="text" class="form-control" name="cedestudiante" id="cedestudiante" maxlength="10" value="{{$estudiante->cedestudiante}}" disabled>
        </div>
    </div>
    <div class="form-group row">
        <label for="nomestudiante" class="col-sm-2 col-form-label">Nombres</label>
        <div class="col-sm-2">
            <input type="text" class="form-control" name="nomestudiante" id="nomestudiante" required value="{{$estudiante->nomestudiante}}">
        </div>
    </div>
    <div class="form-group row">
        <label for="apeestudiante" class="col-sm-2 col-form-label">Apellidos</label>
        <div class="col-sm-2">
            <input type="text" class="form-control" name="apeestudiante" id="apeestudiante" required value="{{$estudiante->apeestudiante }}">
        </div>
    </div>

    <div class="form-group row">
        <label for="correestudiante" class="col-sm-2 col-form-label">Correo</label>
        <div class="col-sm-2">
            <input type="email" pattern=".+@puce.edu.ec"   class="form-control" name="correestudiante" id="correestudiante" value="{{$estudiante->correestudiante}}">
        </div>
    </div>
    <div class="col-md-4">
        <button type="submit" class="btn btn-primary">Guardar</button>
    </div>
</form>
```

Figura 211. Estudiantes Edit Blade 2

Es importante definir el nombre de cada tag <input> y <select> en base al nombre declarado en la función Index del controlador. Ya que caso contrario no se va a poder recuperar los datos para cargar los valores del registro previamente ingresado en la tabla CRUD y posteriormente actualizar el registro. No se debe olvidar que la/las claves primarias se mantienen fijas hasta acabar la edición de los campos seleccionables o editables.

Vista Eliminar Estudiantes:

La Vista Eliminar Estudiantes permite eliminar un registro previamente ingresado dentro de la tabla Estudiantes de la Base de Datos del sistema. Esta opción se encuentra disponible directamente desde la tabla CRUD en la Vista Principal de Estudiantes.

```
<form action="{{action('EstudianteController@destroy', $estudiante->cedestudiante)}}" method="POST" onsubmit="return ConfirmDelete()"
    @csrf
    <input name="_method" type="hidden" value="POST">
    <td><button class='btn btn-danger-glyphicon glyphicon glyphicon-trash'></button></td>
</form>
```

Figura 212. Estudiantes Blade – Eliminar Registro

Es importante recordar que para borrar un registro se necesita confirmar el proceso de borrado. Para esto se utiliza una propiedad del tag <form> llamada onsubmit, que permite confirmar el proceso de borrado; que por cierto en el caso de Estudiantes no es lógico, pero impide borrar en cascada si existe clave foránea de la tabla Estudiantes en otro módulo.

Interfaces Estudiantes:

Las Interfaces de Estudiantes se encuentran separadas por cada funcionalidad descrita en las Vistas de Estudiantes.

Interface Principal de Estudiantes (Tabla CRUD):

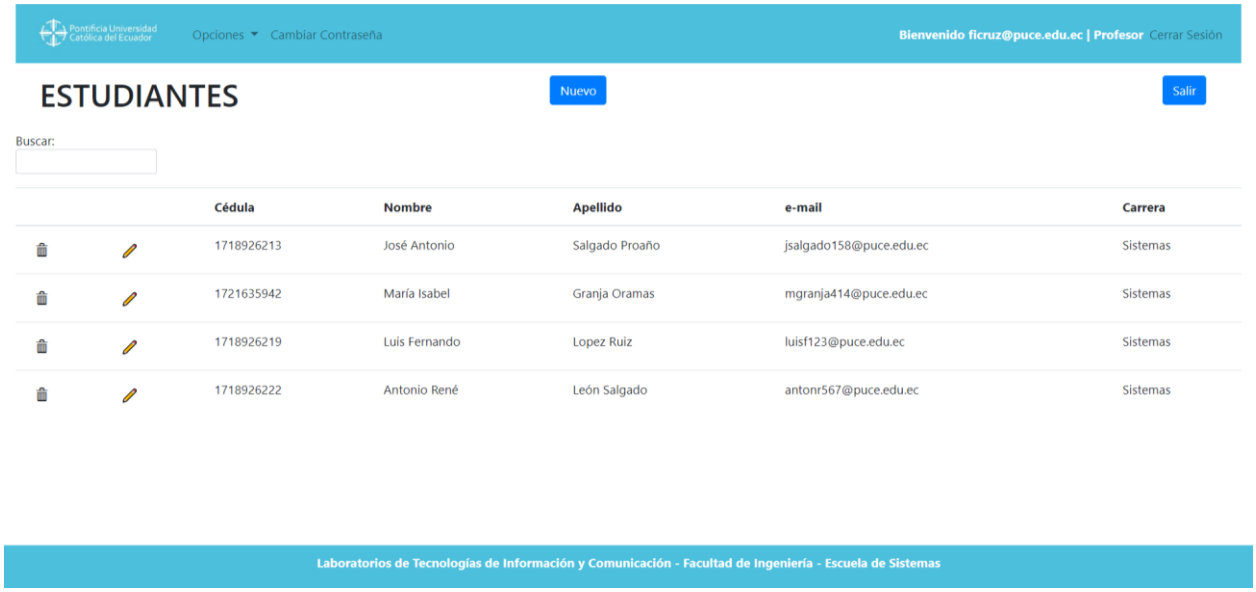


Figura 213. Página de Estudiantes Principal

Como se puede observar, la Interface Principal de Estudiantes tiene un diseño de tabla Responsivo que permite manejar las 4 operaciones básicas que componen a un CRUD (crear, borrar, actualizar y consultar). Sin embargo, no se debe olvidar que para alterar e ingresar un nuevo registro se accede a la Interface de Ingreso Estudiantes y a la Interface Edición Estudiantes, mediante el Controlador de Estudiantes.

Interface Ingreso Estudiantes por Formulario:

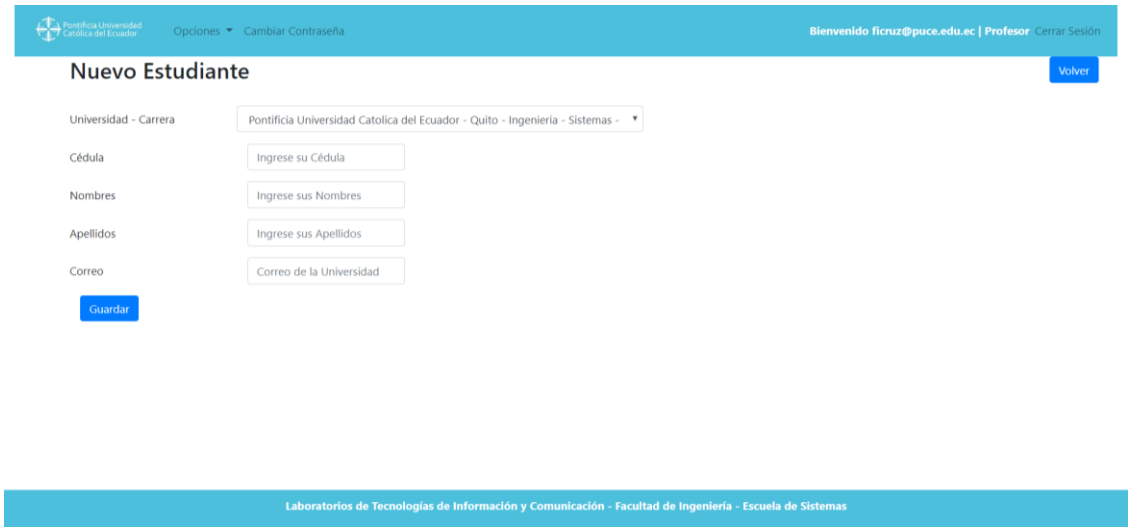


Figura 214. Página de Estudiantes Crear

Como se puede observar, en la Interface de Ingreso de Estudiantes se colocan los datos para agregar un nuevo registro en la tabla de Estudiantes. Al presionar guardar se agrega un nuevo registro y se coloca en la tabla CRUD. La clave primaria de esta tabla corresponde las cédulas de identidad de un Estudiante.

Interface Ingreso Estudiantes por Excel:

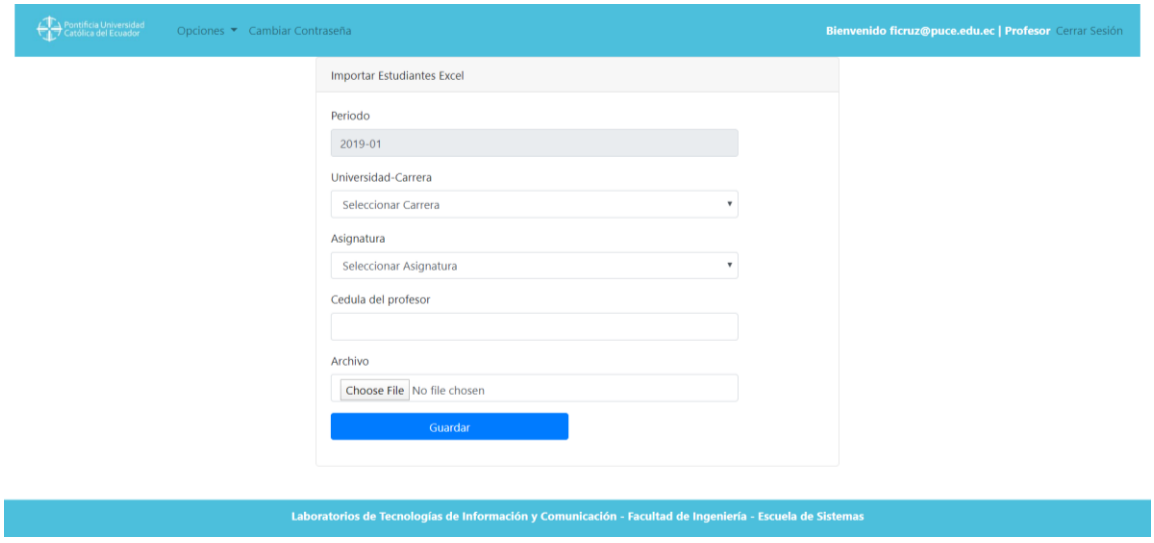
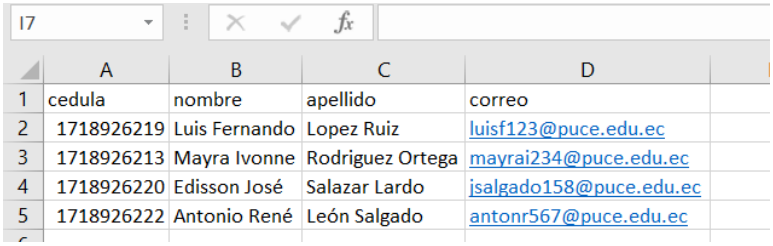


Figura 215. Página de Estudiantes Excel Crear

En la Interface de Ingreso de Estudiantes Excel se seleccionan y colocan los datos para agregar nuevos registros en la tabla de Estudiantes al subir un documento de Excel con extensión .xls. Al presionar guardar se agregan los nuevos registros y se colocan en la tabla CRUD. La clave primaria de esta tabla corresponde las cédulas de identidad de un Estudiante. No obstante, se realiza también una asignación de Estudiantes por materia en base a la Asignatura dictada por el Profesor dentro de la tabla AsignaturasxEstudiantes; el documento lo proporciona el Profesor y cumple con el siguiente formato en Excel:



	A	B	C	D
1	cedula	nombre	apellido	correo
2	1718926219	Luis Fernando	Lopez Ruiz	luisf123@puce.edu.ec
3	1718926213	Mayra Ivonne	Rodriguez Ortega	mayrai234@puce.edu.ec
4	1718926220	Edisson José	Salazar Lardo	jsalgado158@puce.edu.ec
5	1718926222	Antonio René	León Salgado	antonr567@puce.edu.ec

Figura 216. Formato de Ejemplo de Documento de Excel para Agregar Estudiantes

En el documento base de Excel para agregar nuevos registros se deben declarar las 4 primeras celdas A1, B1, C1 y D1 con nombres referentes a los Atributos de la tabla Estudiantes dentro de la Base de Datos; los nombres base son cedula, nombre, apellido y correo respectivamente. Finalmente, de forma adyacente, se colocan los datos con cada uno de los valores requeridos por columna, y se procede a guardar el documento de Excel con formato .xls.

Interface Edición Estudiantes:

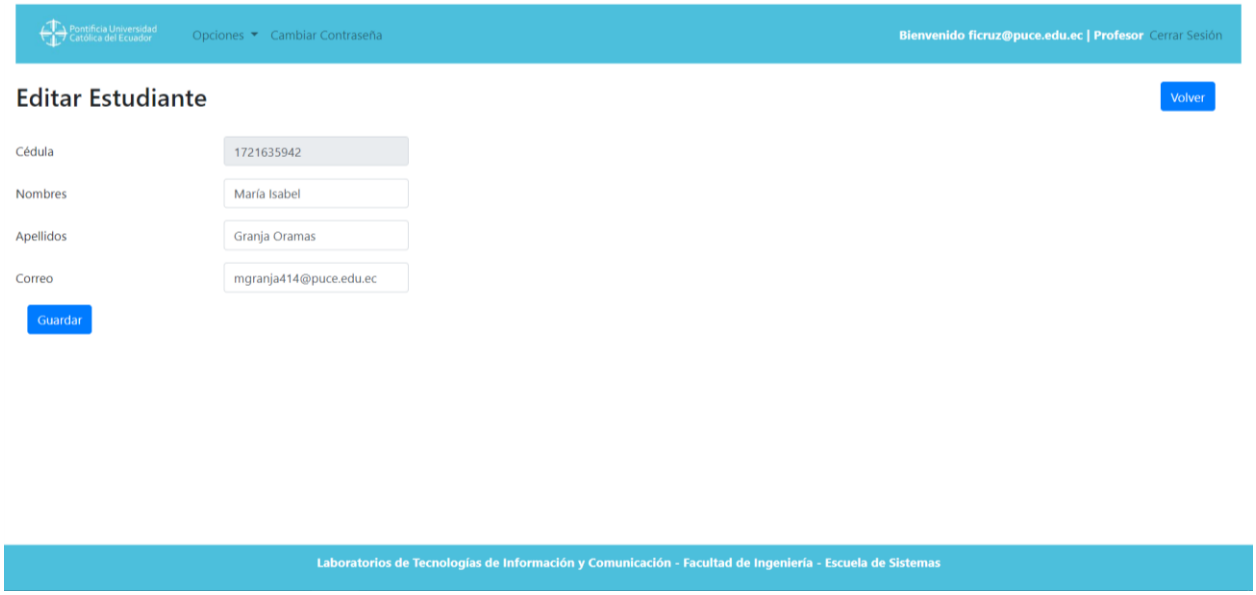


Figura 217. Página de Estudiantes Editar

En la Interface de Edición de Estudiantes se colocan los datos recuperados de un registro previamente ingresados, para editar y actualizar un registro ya existente de la tabla Estudiantes. Al presionar guardar se modifica el registro ya en base a la cédula del Estudiante con formato no editable y se coloca en la tabla CRUD.

Requisitos, Relaciones, Procesos de Grabado, Validaciones y Estructuras que se Deben Tomar en Cuenta para la Construcción de Interfaces:

Estudiantes General:

Sistema: Agenda Electrónica

Tabla: ESTUDIANTES

Relaciones: carr_tiene_est, est_toma_asigxest

Requisitos Previos: Validar existencia de Carreras

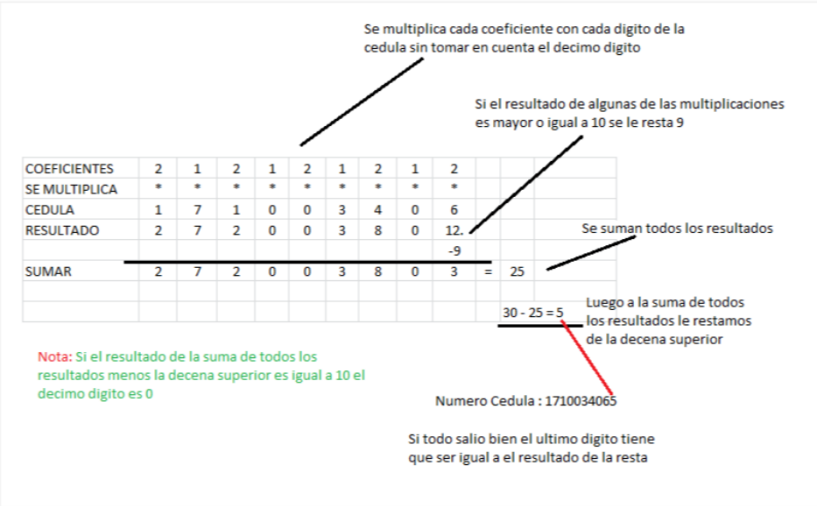
Proceso de grabado: Tabla Estudiantes.

Validaciones:

Se selecciona la descripción de Universidad, Sede, Facultad, Escuela, Carrera en base al combo box Universidad-Carrera, en base a esta selección se cargan internamente las cédulas disponibles de los Profesores por Universidad. Ej.: P.U.C.E - Quito - Ingeniería - Sistemas – Sistemas. Se debe almacenar el código de Carrera en la base de datos.

Se ingresa la cedula del Profesor y de existir, se debe cargar los datos del Profesor en los campos visibles (nombres, apellidos, correo y celular).

Validar que la cédula del Estudiante sea única y que no exceda de 10 caracteres, para esto se va a utilizar el algoritmo del digito verificador para validar que sea una cedula real, dicho algoritmo se puede observar a continuación:



COEFICIENTES	2	1	2	1	2	1	2	1	2		
SE MULTIPLICA	*	*	*	*	*	*	*	*	*		
CEDULA	1	7	1	0	0	3	4	0	6		
RESULTADO	2	7	2	0	0	3	8	0	12.		
									-9		
SUMAR	2	7	2	0	0	3	8	0	3	=	25

Figura 218. Comprobación Formato de Cédula

Se valida que no existan caracteres numéricos en el nombre y apellido.

Se ingresa el celular del Estudiante y se valida que solo existan caracteres numéricos.

Se ingresa el correo electrónico y se valida que conserve el formato "……."+@puce.edu.ec.

Para borrar un Estudiante, debe validar que no exista una ASIGNATURASXESTUDIANTES relacionadas a dicho Estudiante.

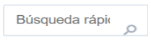
Botones:

Agrega una nueva fila para ingresar un nuevo registro.

Para confirmar y guardar en la base de datos se presiona el botón de guardar ya sea que se esté ingresando o modificando.

El botón borrar permite eliminar un registro ingresado previamente de acuerdo con las validaciones definidas. Antes de eliminar se muestra un mensaje de confirmación si es que de verdad desea eliminar ese registro.

El botón salir permite regresar a la pantalla principal.



La barra de búsqueda despliega el registro que se encuentra realizando una consulta en base a cualquier atributo de la tabla.

Estudiantes Excel:

Sistema: Agenda Electrónica

Tabla: ESTUDIANTES

Relaciones: carr_tiene_est, est_toma_asigxest

Requisitos Previos: Validar existencia de Facultades, Escuelas, Carreras

Proceso de grabado: Tabla Estudiantes.

Validaciones:

Se selecciona la descripción de Universidad, Sede, Facultad, Escuela, Carrera en base al combo box Universidad-Carrera, con esta selección se cargan las Asignaturas existentes en el combo box de Asignaturas. Ej.: P.U.C.E - Quito - Ingeniería - Sistemas – Sistemas. Se debe almacenar el código de Carrera en la base de datos.

Se selecciona la descripción de Asignatura en base a las Asignaturas existentes en la tabla ASIGNATURAS, con esta selección se cargan los datos el combo box de unidades. El nombre de Asignatura de la Evaluación se coloca automáticamente en base a una consulta por código de tema, el campo es no es editable.

Se debe presionar el botón seleccionar archivo, al hacerlo se debe validar que se tenga un archivo separado por comas con las siguientes columnas: cedula, apellido, nombre, nombre de usuario.

El código de Carrera es ingresado automáticamente por el programa lector de Excel como una columna adicional.

El correo institucional se lo genera en base al nombre de usuario + @puce.edu.ec. Finalmente, se lo ingresa como una columna adicional en el Excel.

Se elimina columna nombre de usuario y se deja la tabla de Excel en el siguiente formato: cedula, código de Carrera, nombre de Estudiante, apellido de Estudiante y correo del Estudiante.

Al haber cumplido con los puntos anteriores, se lee la tabla de Excel y se almacena los registros en la base de datos, tabla ESTUDIANTES presionando el botón guardar.

Botones:

Agrega una nueva fila para ingresar un nuevo registro.

Para confirmar y guardar en la base de datos se presiona el botón de guardar ya sea que se esté ingresando o modificando.

El botón salir permite regresar a la pantalla principal.



La barra de búsqueda despliega el registro que se encuentra realizando una consulta en base a cualquier atributo de la tabla.

F10: Módulo Gestionar Profesor

La Gestión de Profesores en el sistema permite manejar diversos Profesores que dictan Asignaturas en una Universidad. El propósito de cada Profesor es acceder a la Agenda Electrónica y gestionar Contenido interactivo, que le permita a uno o varios Estudiantes tomar Anotaciones relevantes de cada Tema que compone a una Materia. Además, un Profesor puede gestionar todo el syllabus de varias de sus Asignaturas y asignar uno o varios Estudiantes que se encuentren matriculados a la mismas.

Controlador de Profesores:

El Controlador de Profesores es el lugar que permite definir los métodos en la capa lógica (DP) que utilizan las peticiones del archivo routes.php→web.php, para realizar acciones dentro de una vista con respecto al ingreso, edición, borrado y consulta de información con respecto a Profesores.

El ingreso de Profesores lo realiza un Administrativo del sistema, sin embargo, siempre puede cambiar su contraseña en la pantalla principal de Agenda Electrónica. En

donde, además, tiene acceso para revisar las Anotaciones realizadas por cada Estudiante perteneciente a alguna de sus Asignaturas dictadas. Sin embargo, no puede editar alguna anotación observada, simplemente puede verlas.

Index:

Método que permite cargar información de Profesores en la tabla tipo CRUD. Además de enviar un arreglo de Profesores para cargar información en <input> y <label> tags.

```php
public function index(Request $request)
{
 if($request->get('cedula')!= ""){
    $profesores = DB::table('profesores')->join('carreras','carreras.codcarrera','=','profesores.codcarrera')->where('profesores.cedprofesor','LIKE','%'.$request->get('cedula').'%')->paginate(10);
    return view('profesores',compact('profesores'));
 }
 else{
    $profesores = DB::table('profesores')->join('carreras','carreras.codcarrera','=','profesores.codcarrera')->paginate(10);
    return view('profesores',compact('profesores'));
 }
}
```

Figura 219. Método Index Profesores

Create:

Método que permite abrir y redirigir a la Vista Ingresar Profesores. Además de enviar un arreglo de Carreras para cargar información en <input> y <label> tags.

```php
public function create()
{
   $carreras = DB::select('SELECT carreras.codcarrera, carreras.desccarrera, escuelas.descescuela, facultades.descfacultad, sedes.descsede, universidades.descuniversidad
                FROM carreras natural join escuelas natural join facultades natural join facultadesxsedes natural join sedes natural join universidades;');

   return view('profesores_crear', compact('carreras'));
}
```

Figura 220. Método Create Profesores

Store:

Método que permite almacenar un nuevo registro de Profesores dentro de la base de datos del sistema mediante el uso de un formulario dentro de la vista Crear Profesores.

```php
public function store(Request $request)
{
    $buscarProf = Profesor::where('cedprofesor', $request->input('cedprofesor'))->first();
    $buscarEst = Estudiante::where('cedestudiante', $request->input('cedprofesor'))->first();

    if(is_null($buscarProf) && is_null($buscarEst))
    {
        $newmail = $request->input('correprofesor');
        $existe = User::where('email', $newmail)->first();
        if (is_null($existe))
        {
            //profesor
            $profesor = new Profesor();
            $profesor->cedprofesor = $request->input('cedprofesor');
            $profesor->codcarrera = $request->input('codcarrera');
            $profesor->nomprofesor = $request->input('nomprofesor');
            $profesor->apeprofesor = $request->input('apeprofesor');
            $profesor->correprofesor = $request->input('correprofesor');
            $profesor->celuprofesor = $request->input('celuprofesor');
            $profesor->save();
            //usuario
            $testmail=$request->input('correprofesor');
            User::create([
                'name' => Str::substr($testmail, 0, strrpos($testmail, '@')),
                'email' => $request->input('correprofesor'),
                'password' => bcrypt((String)($request->input('cedprofesor'))),
                'remember_token' => Str::random(5),
            ]);
            $id_usuario= User::where('email', $request->input('correprofesor'))->first();
            $id_usuarioC = $id_usuario->id;
            //rol de profesor
            DB::table('role_user')->insert([
                'role_id'=> 2,
                'user_id'=> (int)$id_usuarioC,
            ]);
            return redirect('pagProfesores');
        }
        else
        {
            return redirect()->back() ->with(['error' => 'No se puede ingresar el profesor. El correo que esta ingresando ya lo tiene registrado otro usuario']);
        }
```

Figura 221. Método Store Profesores

Este método particular permite no solo controlar el ingreso de los Profesores en la base de datos del sistema, sino que también permite manejar el ingreso de nuevos usuarios con rol de Profesor dentro de la tabla Usuarios y Roles en la base de datos del sistema.

Para esto primero verifica que el correo del Usuario en general no exista dentro del sistema, y posteriormente se compara la existencia de las cédulas de identidad para evitar que se ingrese un usuario duplicado. Finalmente, se ingresa un Profesor y Usuario dentro de sus respectivas tablas y se manda un mensaje a consola que dice registro ingresado correctamente.

Destroy:

Método que permite eliminar un registro existente de la tabla Profesores y Usuarios en la base de datos del sistema. Para poder eliminar se recibe la cédula de identidad del Profesor y si se logra eliminar el registro exitosamente, se manda un mensaje a consola que diga registro borrado exitosamente. Caso contrario y si no se puede eliminar debido a que se irrumpe con la integridad referencial de las tablas y sus registros, se manda un mensaje que dice que no se puede eliminar Profesor debido a que está encargado de una Asignatura.

```php
public function destroy($id)
{
    $asigxest = DB::table('asignaturasxestudiantes')
    ->join('asignaturasxprofesor','asignaturasxestudiantes.cedprofesor','=','asignaturasxprofesor.cedprofesor')->where('asignaturasxprofesor.cedprofesor', '=', $id)->get();
    if($asigxest->count()==0)
    {
        //profesor
        $profesor = Profesor::find($id);
        $profesor->delete();
        //usuario
        $usuarioProf= User::where('email', $profesor->correprofesor)->first();
        $usuarioProf->delete();
        return redirect()->action('ProfesorController@index', ['success' => 'Profesor eliminado']);
    }
    else{

        return redirect()->back() ->with(['error' => 'No se puede eliminar el profesor. Está encargado de una asignatura']);
    }
}
```

Figura 222. Método Destroy Profesores

Edit:

Método que permite abrir y redirigir a la Vista Editar Profesores, para esto se recibe la cédula del Profesor, que en base a una consulta SQL, permita recuperar el objeto Profesor que posteriormente se va a editar de la base de datos.

```php
public function edit($id)
{
    $profesor = Profesor::where('cedprofesor', $id)->first();
    return view('profesores_editar', compact('profesor'));
}
```

Figura 223. Método Edit Profesores

Update:

Método que permite actualizar un registro existente de la tabla Profesores y Usuarios en la base de datos del sistema, para esto se utiliza la cédula de identidad del

Profesor recibida a través de la vista Editar Profesores. Si se ingresa un correo ya existente por otro Usuario, se envía un mensaje que dice que no se puede actualizar Profesor.

```php
public function update(Request $request, $id)
{
    $profesor = Profesor::find($id);
    $oldmail = $profesor->correprofesor;
    $usuarioProf= User::where('email', $oldmail)->first();
    $newmail = $request->input('correprofesor');
    $existe = User::where('email', $newmail)->first();
    if ( is_null($existe) || $existe->email==$oldmail)
    {
        ///profesor table
        $profesor->nomprofesor = $request->input('nomprofesor');
        $profesor->apeprofesor = $request->input('apeprofesor');
        $profesor->correprofesor = $request->input('correprofesor');
        $profesor->celuprofesor = $request->input('celuprofesor');
        $profesor->save();
        ///user table
        $usuarioProf->name = Str::substr($newmail, 0, strrpos($newmail, '@'));
        $usuarioProf->email = $newmail;
        $usuarioProf->remember_token = Str::random(5);
        $usuarioProf->save();
        return redirect()->action('ProfesorController@index', ['success' => 'Profesor actualizado']);
    }
    else
    {
        return redirect()->back()->with(['error' => 'No se puede actualizar el profesor. El correo que esta ingresando ya lo tiene registrado otro profesor']);
    }

}
```

Figura 224. Método Update Profesores

Modelo de Profesores:

El Modelo de Profesores permite enlazar la capa de manejo de datos (MD) para recibir y enviar información con respecto a la tabla de la base de datos llamada Profesores que se encuentra en PostgreSQL; la base de datos total del sistema se encuentra albergada en un gestor para el manejo de base de datos, que en este caso es pgAdmin 4.

El Modelo de Profesores se lo utiliza dentro del Controlador de Profesores para llamar al ORM por defecto de Laravel llamado Eloquent, que interpreta procesos de bases de datos escritos con instrucciones php; los cuales posteriormente son traducidos a sentencias SQL que realizan operaciones dentro de la tabla Profesores en la base de datos.

Es importante recordar que, para construir el Modelo de Profesores, se debe tomar en cuenta el nombre de los atributos como se encuentran en la tabla Profesores de la base de datos y la clave primaria que identifica a la tabla Profesores.

```
class Profesor extends Model
{
        protected $table = 'profesores';
        protected $primaryKey = 'cedprofesor';
        public $timestamps = false;
        protected $fillable = ['codcarrera','cedprofesor','nomprofesor','apeprofesor','correprofesor','celuprofesor'];
}
```

Figura 225. Profesores Model

Rutas de Profesores:

En el archivo de Rutas de Profesores se encuentran las direcciones que permiten conectar la capa lógica (DP) con las vistas (GUI) de los Profesores dentro del sistema.

```
//PROFESORES
Route::get('/pagProfesores','ProfesorController@index');
Route::get('/pagProfesores/crear','ProfesorController@create');
Route::post('/pagProfesores/crear','ProfesorController@store');
Route::get('/pagProfesores/editar/{id}','ProfesorController@edit');
Route::post('/pagProfesores/editar/{id}','ProfesorController@update');
Route::post('/pagProfesores/{id}','ProfesorController@destroy');
```

Figura 226. Profesores Formularios Route Path

Vista Principal de Profesores:

La Vista Principal de Profesores permite mostrar todos los datos que se encuentran almacenados en la tabla Profesores de la Base de Datos del sistema. Los datos del registro se reflejan en una tabla tipo CRUD que permite seleccionar el tipo de operación que se quiere realizar con respecto a los datos (insertar, borrar y actualizar). Las operaciones se ven reflejadas en forma de botones.

```
<div class="row"style="    margin-bottom: 1%;  margin-top: 1%; margin-left: 1%">
        <div class="col-sm-3">
            <h1>PROFESORES</h1>
        </div>
        <div class="col-md-2"></div>
        <div class="col-md-6">
                <a href="pagProfesores/crear"><button type="submit" class="btn btn-primary">Nuevo</button></a>
            </div>
            <div class = "col-md-1">
            <a href="/"><button type="submit" class="btn btn-primary">Salir</button></a>
            </div>
        </div>
    <div class="table-container">
        <table id="myTable" class="table table-hover "  style="width:100%; margin-top:10px">
        <thead class="thead-light">
            <tr>
                <th></th>
                <th ></th>
                <th >Cédula</th>
                <th >Nombre</th>
                <th >Apellido</th>
                <th >e-mail</th>
                <th >Celular</th>

                <th>Carrera</th>

            </tr>
        </thead>
```

Figura 227. Profesores Blade 1

Es necesario crear una tabla que sirva como almacén para los registros recuperados de la tabla Profesores que se encuentra en la base de datos del sistema. Sin embargo, se debe usar un tag <form> con método get para poder recuperar la información en un solo sentido y se debe especificar el url exacto de la ruta definida en el archivo web.php.

```
@if($profesores->count())
@foreach ($profesores as $profesor)
<tr>
    <form action="{{action('ProfesorController@destroy', $profesor->cedprofesor)}}" method="POST" onsubmit="return ConfirmDelete()">
        @csrf
        <input name="_method" type="hidden" value="POST">
        <td><button class='btn btn-danger-glyphicon glyphicon glyphicon-trash'></button></td>
    </form>
    <td><a href="pagProfesores/editar/{{$profesor->cedprofesor}}"><button class='btn btn-warning-glyphicon glyphicon-pencil'></button></a></td>
    <td >
    <label name="estudianteci">{{ $profesor->cedprofesor }}</label>
    </td>

    <td >
        <label name="estudiantenombre">{{ $profesor->nomprofesor }}</label>
    </td>
    <td >
        <label name="estudianteapellido">{{ $profesor->apeprofesor }}</label>
    </td>

    <td >
        <label name="estudianteemail">{{ $profesor->correprofesor }}</label>
    </td>
    <td >
        <label name="estudianteemail">{{ $profesor->celuprofesor }}</label>
    </td>
    <td >
        <label name="estudianteescuela">{{ $profesor->desccarrera }}</label>
    </td>
</tr>
```

Figura 228. Profesores Blade 2

Es importante definir las columnas <td> en base al nombre declarado en la función Index del controlador. Ya que caso contrario no se va a poder recuperar los datos y cargarlos en las celdas de la tabla.

```
<script>
    $(document).ready(function () {
        $('#myTable').DataTable({
            "paging": false,
        "info": false,
        "autoWidth": true,
        "searching": true, // Search box and search function will be actived
        "dom": '<"top"f>rt<"bottom"ilp><"clear">',
        "language": {
        "zeroRecords": "No existe registros",
        "infoEmpty": "No se econtró ningún registro",
        "sSearch": "Buscar:   "
            }
        });
    });
    function ConfirmDelete()
    {

    var x = confirm("¿Esta seguro que desea eliminar este profesor?");
    if (x)
        return true;
    else
        return false;
    }
</script>
```

Figura 229. Estudiantes Blade 3

La librería Data Tables se utiliza para poder cargar de forma dinámica y con posibilidad de obtener un campo de búsqueda por columna y coincidencia los datos de una consulta SQL en la base de datos. Aquí se considera el tipo de notación JSON para indexar campos y así realizar consultas que reordenen la tabla por id.

Vista Ingresar Profesores:

La Vista Ingresar Profesores permite ingresar un nuevo registro en la tabla Profesores de la Base de Datos del sistema.

```
<form method="post" action = "/pagProfesores/crear">
    {{ csrf_field() }}
    <div class="form-group row">
    <label for="sedeNombre" class="col-sm-2 col-form-label">Universidad - Carrera</label>
    <div style="width: 38%;">
    <select class="form-control" id="codcarrera" name="codcarrera">
            @foreach ($carreras as $carrera)
                <option value="{{$carrera->codcarrera}}">{{$carrera->descuniversidad}} -  {{$carrera->descsede}} -  {{$carrera->descfacultad}} -  {{$carrera->descescuela}}
                -  {{$carrera->desccarrera}}</option>
            @endforeach
    </select>
    </div>
```

Figura 230. Profesores Create Blade 1

Como se pudo observar, es necesario colocar la instrucción {{ csrf_field()}} que permite denegar la falsificación de petición en sitios cruzados (se origina cuando un sitio realiza peticiones en nombre de un usuario sobre nuestro sitio). Al denegar la petición permite acceder a las rutas declaradas en el web.php con post y get para ingresar, recuperar y alterar información dentro de un Blade al usar los tags <input>, <select>, etc.



Figura 231. Profesores Create Blade 2

Finalmente, es importante definir el nombre de cada tag <input> y <select> en base al nombre declarado en la función Index del controlador. Ya que caso contrario no se va a poder recuperar los datos y posteriormente ingresar el registro.

Vista Modificar Profesores:

La Vista Modificar Profesores permite modificar un registro previamente ingresado dentro de la tabla Estudiantes de la Base de Datos del sistema.

```
<div class="row">
    <div class="col-sm-11">
        <h2>Editar Profesor</h2>
    </div>
    <div class="col-sm-1">
        <a href="/pagProfesores" class="btn btn-primary"> Volver</a>
    </div>
</div>

<br>


<form method="post" action = "">
    {{ csrf_field() }}
    <div class="form-group row">
        <label for="cedprofesor" class="col-sm-2 col-form-label">Cédula</label>
        <div class="col-sm-2">
            <input type="text" class="form-control" name="cedprofesor" id="cedprofesor" maxlength="10" value="{{$profesor->cedprofesor}}" disabled>
        </div>
```

Figura 232. Profesores Edit Blade 1

Es necesario colocar la instrucción {{ csrf_field()}} que permite denegar la falsificación de petición en sitios cruzados (se origina cuando un sitio realiza peticiones en nombre de un usuario sobre nuestro sitio). Al denegar la petición permite acceder a las rutas declaradas en el web.php con post y get para ingresar, recuperar y alterar información dentro de un Blade al usar los tags <input> , <select>, etc.

```
<div class="form-group row">
    <label for="cedprofesor" class="col-sm-2 col-form-label">Cédula</label>
    <div class="col-sm-2">
        <input type="text" class="form-control" name="cedprofesor" id="cedprofesor" maxlength="10" value="{{$profesor->cedprofesor}}" disabled>
    </div>
</div>
<div class="form-group row">
    <label for="nomprofesor" class="col-sm-2 col-form-label">Nombres</label>
    <div class="col-sm-2">
        <input type="text" class="form-control" name="nomprofesor" id="nomprofesor" required value="{{$profesor->nomprofesor}}">
    </div>
</div>
<div class="form-group row">
    <label for="apeprofesor" class="col-sm-2 col-form-label">Apellidos</label>
    <div class="col-sm-2">
        <input type="text" class="form-control" name="apeprofesor" id="apeprofesor" required value="{{$profesor->apeprofesor }}">
    </div>
</div>
<div class="form-group row">
    <label for="celuprofesor" class="col-sm-2 col-form-label">Celular</label>
    <div class="col-sm-2">
        <input type="text" class="form-control" name="celuprofesor" id="celuprofesor" required value="{{$profesor->celuprofesor }}">
    </div>
</div>

<div class="form-group row">
    <label for="correprofesor" class="col-sm-2 col-form-label">Correo</label>
    <div class="col-sm-2">
        <input type="email" pattern=".+@puce.edu.ec"   class="form-control" name="correprofesor" id="correprofesor" value="{{$profesor->correprofesor}}">
    </div>
</div>
<div class="col-md-4">
    <button type="submit" class="btn btn-primary">Guardar</button>
</div>
```

Figura 233. Profesores Edit Blade 2

Finalmente, es importante definir el nombre de cada tag <input> y <select> en base al nombre declarado en la función Index del controlador. Ya que caso contrario no

se va a poder recuperar los datos para cargar los valores del registro previamente ingresado en la tabla CRUD y posteriormente actualizar el registro. No se debe olvidar que la/las claves primarias se mantienen fijas hasta acabar la edición de los campos seleccionables o editables.

Vista Eliminar Profesores:

La Vista Eliminar Profesores permite eliminar un registro previamente ingresado dentro de la tabla Profesores de la Base de Datos del sistema. Esta opción se encuentra disponible directamente desde la tabla CRUD en la Vista Principal de Profesores.

```
<form action="{{action('ProfesorController@destroy', $profesor->cedprofesor)}}" method="POST" onsubmit="return ConfirmDelete()"
    @csrf
    <input name="_method" type="hidden" value="POST">
    <td><button class='btn btn-danger-glyphicon glyphicon glyphicon-trash'></button></td>
</form>
```

Figura 234. Profesores Blade – Eliminar Registro

Es importante recordar que para borrar un registro se necesita confirmar el proceso de borrado. Para esto se utiliza una propiedad del tag <form> llamada onsubmit, que permite confirmar el proceso de borrado; que por cierto en el caso de Profesores no es lógico, pero impide borrar en cascada si existe clave foránea de la tabla Profesores en otro módulo.

Interfaces Profesores:

Las Interfaces de Profesores se encuentran separadas por cada funcionalidad descrita en las Vistas de Profesores.

Interface Principal de Profesores (Tabla CRUD):

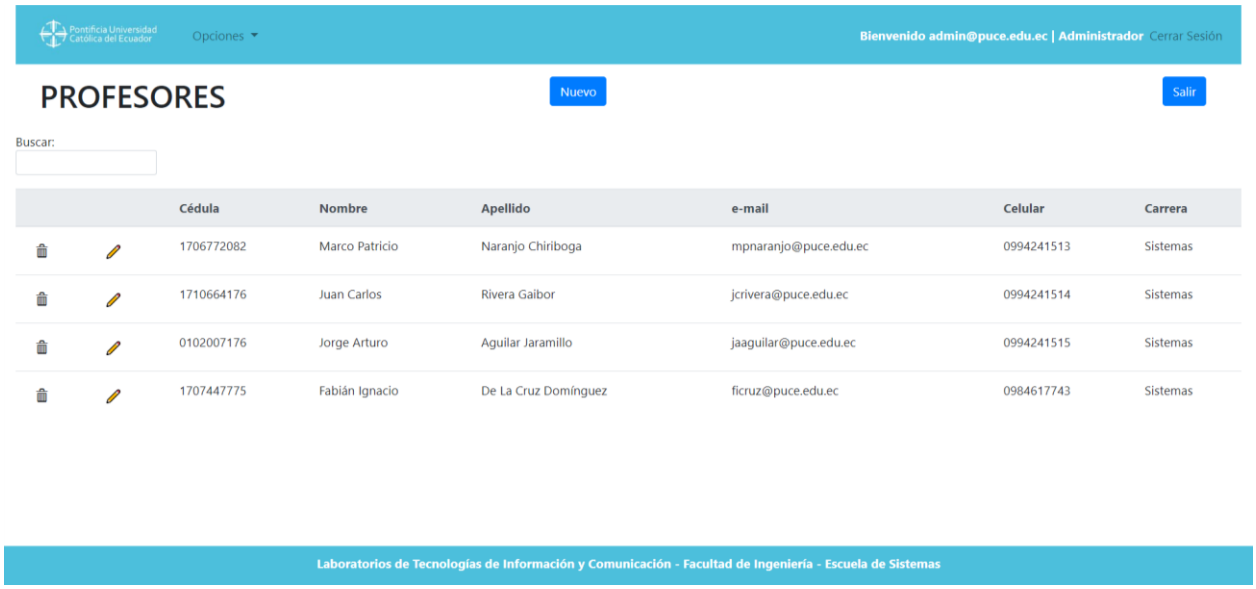


Figura 235. Página de Profesores Principal

Como se puede observar, la Interface Principal de Profesores tiene un diseño de tabla Responsivo que permite manejar las 4 operaciones básicas que componen a un CRUD (crear, borrar, actualizar y consultar).

Interface Ingreso Profesores por Formulario:

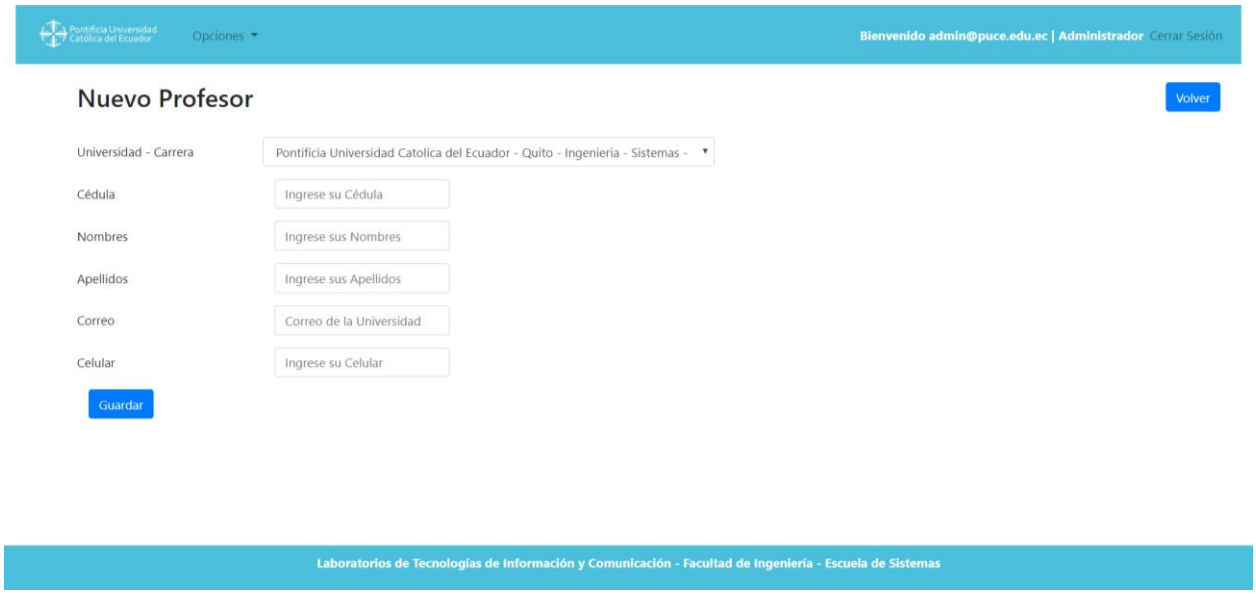


Figura 236. Página de Profesores Crear

Como se puede observar, en la Interface de Ingreso de Profesores se colocan los datos para agregar un nuevo registro en la tabla de Profesores. Al presionar guardar se agrega un nuevo registro y se coloca en la tabla CRUD. La clave primaria de esta tabla corresponde las cédulas de identidad de un Profesor.

Interface Edición Profesores:

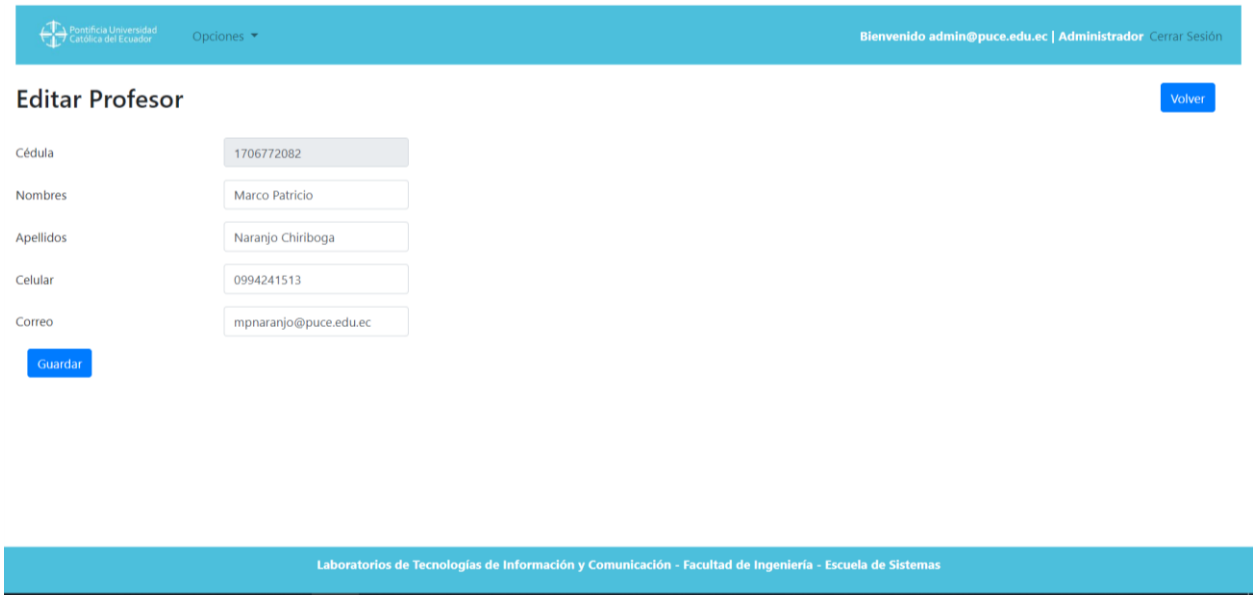


Figura 237. Página de Profesores Editar

Como se puede observar, en la Interface de Edición de Profesores se colocan los datos recuperados de un registro previamente ingresados, para editar y actualizar un registro ya existente de la tabla Profesores. Al presionar guardar se modifica el registro ya en base a la cédula del Profesor con formato no editable y se coloca en la tabla CRUD.

Requisitos, Relaciones, Procesos de Grabado, Validaciones y Estructuras que se Deben Tomar en Cuenta para la Construcción de Interfaces:

Profesores General:

Sistema: Agenda Electrónica

Tabla: PROFESORES

Relaciones: carr_conf_por_prof, prof_dicta_asigxest, prof_dicta_asigxprof

Requisitos Previos: Validar existencia de Carreras

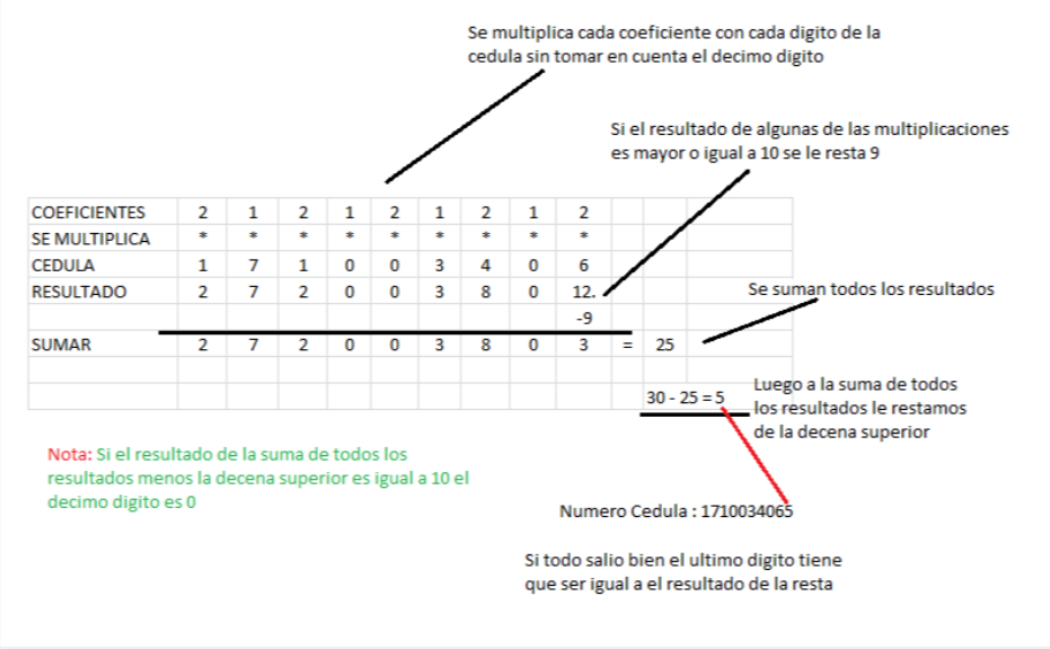
Proceso de grabado: Tabla PROFESORES.

Validaciones:

Se selecciona la descripción de Universidad, Sede, Facultad, Escuela, Carrera en base al combo box Universidad-Carrera, en base a esta selección se cargan internamente las cédulas disponibles de los Profesores por Universidad. Ej.: P.U.C.E - Quito - Ingeniería - Sistemas – Sistemas. Se debe almacenar el código de Carrera en la base de datos.

Se ingresa la cédula del Profesor y de existir, se debe cargar los datos del Profesor en los campos visibles (nombres, apellidos, correo y celular).

Validar que la cédula de Profesor sea única y que no exceda de 10 caracteres, para esto se va a utilizar el algoritmo del digito verificador para validar que sea una cedula real, dicho algoritmo se puede observar a continuación:



COEFICIENTES	2	1	2	1	2	1	2	1	2			
SE MULTIPLICA	*	*	*	*	*	*	*	*	*			
CEDULA	1	7	1	0	0	3	4	0	6			
RESULTADO	2	7	2	0	0	3	8	0	12.			
									-9			
SUMAR	2	7	2	0	0	3	8	0	3	=	25	
											30 - 25 = 5	

Figura 238. Comprobación Formato de Cédula

Se ingresa por lo menos un nombre y apellido y se valida que no existan caracteres numéricos.

Se ingresa el celular del Profesor y se valida que solo existan caracteres numéricos.

Se ingresa el correo electrónico y se valida que conserve el formato "……."+@puce.edu.ec.

Para borrar un Profesor, debe validar que no exista una ASIGNATURASXESTUDIANTES y ASIGNATURASXPROFESOR relacionadas a dicho Profesor.

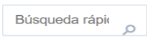
Botones:

Agrega una nueva fila para ingresar un nuevo registro.

Para confirmar y guardar en la base de datos se presiona el botón de guardar ya sea que se esté ingresando o modificando.

El botón borrar permite eliminar un registro ingresado previamente de acuerdo con las validaciones definidas. Antes de eliminar se muestra un mensaje de confirmación si es que de verdad desea eliminar ese registro.

El botón salir permite regresar a la pantalla principal.



La barra de búsqueda despliega el registro que se encuentra realizando una consulta en base a cualquier atributo de la tabla.

F11: Módulo Gestionar Asignatura

La Gestión de Asignaturas en el sistema permite manejar diversas Asignaturas que son dictadas por varios Profesores en una Universidad. El propósito de la Asignatura es

colocar una descripción y nombre que la identifique y que permita que se le asignen Unidades, Temas y Contenidos. Posteriormente los Contenidos le permitirán a uno o varios Estudiantes tomar Anotaciones relevantes de cada Tema que compone a una Asignatura.

Controlador de Asignaturas:

El Controlador de Asignaturas es el lugar que permite definir los métodos en la capa lógica (DP) que utilizan las peticiones del archivo routes.php→web.php, para realizar acciones dentro de una vista con respecto al ingreso, edición, borrado y consulta de información con respecto a Asignaturas.

Index:

Método que permite cargar información de Asignaturas en la tabla tipo CRUD. Además de enviar un arreglo de Asignaturas para cargar información en <input> y <label> tags. De existir algún error en conexión, se envía un mensaje de error con la excepción encontrada.

```
public function index()
{
    try {
    $asignaturas = AsignaturaModel::select('*')->get();
    } catch (\Exception $exception) {
        return back()->withError($exception->getMessage());
    }
    return view('IndexAsignaturas',['asignaturas'=>$asignaturas]);
}
```

Figura 239. Método Index Asignaturas

Create:

Método que permite abrir y redirigir a la Vista Ingresar Asignaturas. Además de enviar un arreglo de Carreras para cargar información en <input> y <label> tags. De existir algún error en conexión, se envía un mensaje de error con la excepción encontrada.

```
public function create()
{
    try {
    $carreras = DB::select('SELECT carreras.codcarrera, carreras.desccarrera, escuelas.descescuela, facultades.descfacultad, sedes.descsede, universidades.descuniversidad
    FROM carreras natural join escuelas natural join facultades natural join facultadesxsedes natural join sedes natural join universidades;');
    } catch (\Exception $exception) {
        return back()->withError($exception->getMessage());
    }
    return view('CreateAsignatura',['carreras'=>$carreras]);
}
```

Figura 240. Método Create Asignaturas

Store:

Método que permite almacenar un nuevo registro de Asignaturas dentro de la base de datos del sistema mediante el uso de un formulario que se encuentra dentro de la vista Crear Asignaturas. De existir algún error en conexión o si se intenta ingresar un registro duplicado, se envía un mensaje de error con la excepción encontrada.

```php
public function store(Request $request)
{
    try {
    $codasignatura = $request->input('codasignatura');
    $descasignatura = $request->input('descasignatura');
    $credasignatura = $request->input('credasignatura');
    $nivelasignatura = $request->input('nivelasignatura');
    $objeasignatura = $request->input('objeasignatura');
    $resulapreasignatura = $request->input('resulapreasignatura');
    $caracapreasignatura = $request->input('caracapreasignatura');


    $codcarrera= $request->input('unies');


    $data = array('codcarrera'=>$codcarrera,
    'codasignatura'=>$codasignatura,
    'descasignatura'=>$descasignatura,
    'credasignatura'=>$credasignatura,
    'nivelasignatura'=>$nivelasignatura,
    'objeasignatura'=>$objeasignatura,
    'resulapreasignatura'=>$resulapreasignatura,
    'caracapreasignatura'=>$caracapreasignatura);
        DB::table('asignaturas')->insert($data);
    }
    catch(\Exception $e){
        return back()->withError($e->getMessage());
    }
    return redirect('pagAsignaturas')->with('success', 'Se añadió correctamente');
}
```

Figura 241. Método Store Asignaturas

Destroy:

Método que permite eliminar un registro existente de la tabla Asignaturas en la base de datos del sistema. Para poder eliminar se recibe el código de la Asignatura, y si se logra eliminar el registro exitosamente, se manda un mensaje a consola que diga registro borrado exitosamente. Caso contrario y si no se puede eliminar debido a que se irrumpe con la integridad referencial de las tablas y sus registros, se manda un mensaje que dice que no se puede eliminar Asignatura debido a que tiene una relación con un Profesor y/o Estudiante.

```php
public function destroy($codasignatura)
{
    try {
    DB::table('asignaturas')->where('codasignatura','=', $codasignatura)->delete();
    } catch (\Exception $exception) {
        return back()->withError($exception->getMessage());
    }
    return redirect('pagAsignaturas')->with('success','Se elimino correctamente');
}
```

Figura 242. Método Destroy Asignaturas

Edit:

Método que permite abrir y redirigir a la Vista Editar Asignaturas, para esto se recibe el código de Asignatura, que en base a una consulta SQL, permita recuperar el objeto Asignatura que posteriormente se va a editar de la base de datos y facilite enviar un objeto Carreras con los parámetros relacionados a la Asignatura recibida para cargar el combo box acumulativo y así poder gestionar las relaciones del ingreso en cascada. De existir algún error en conexión o si no se encuentra un registro usando el código de Asignaturas recibido, se envía un mensaje de error con la excepción encontrada.

```php
public function edit($codasignatura)//abre la ventana para modificar
{
    try {
    $asignaturas = DB::table('asignaturas')->select('*')->where('codasignatura',$codasignatura)->get();
    $codcarrera;
        foreach ($asignaturas as $asig){
            $codcarrera = $asig->codcarrera;
        }
    $carreras = DB::select('SELECT carreras.codcarrera, carreras.desccarrera, escuelas.descescuela, facultades.descfacultad, sedes.descsede, universidades.descuniversidad
    FROM carreras natural join escuelas natural join facultades natural join facultadesxsedes natural join sedes natural join universidades;');

   $cars = DB::select("SELECT carreras.codcarrera, carreras.desccarrera, escuelas.descescuela, facultades.descfacultad, sedes.descsede, universidades.descuniversidad
    FROM carreras natural join escuelas natural join facultades natural join facultadesxsedes natural join sedes natural join universidades where carreras.codcarrera = '".$codcarrera."';");
    } catch (\Exception $exception) {
        return back()->withError($exception->getMessage());
    }
    return view('EditAsignatura',compact('carreras','asignaturas','cars'));
}
```

Figura 243. Método Edit Asignaturas

Update:

Método que permite actualizar un registro existente de la tabla Asignaturas en la base de datos del sistema, para esto se utiliza un objeto de tipo Request recibido a través de la vista Editar Asignaturas. De existir algún error en conexión o si no se encuentra un registro usando el código de Asignaturas o de Carreras recibido por el objeto de tipo Request, se envía un mensaje de error con la excepción encontrada.

```
public function update(Request $request)//modificar
{
    try {
    $codasignatura = $request->input('codasignatura');
    $descasignatura = $request->input('descasignatura');
    $credasignatura = $request->input('credasignatura');
    $nivelasignatura = $request->input('nivelasignatura');
    $objeasignatura = $request->input('objeasignatura');
    $resulapreasignatura = $request->input('resulapreasignatura');
    $caracapreasignatura = $request->input('caracapreasignatura');
    $codcarrera= $request->input('unies');

    $data = array('codcarrera'=>$codcarrera,
    'codasignatura'=>$codasignatura,
    'descasignatura'=>$descasignatura,
    'credasignatura'=>$credasignatura,
    'nivelasignatura'=>$nivelasignatura,
    'objeasignatura'=>$objeasignatura,
    'resulapreasignatura'=>$resulapreasignatura,
    'caracapreasignatura'=>$caracapreasignatura);

    DB::table('asignaturas')->where('codasignatura','=', $codasignatura)->update($data);
    } catch (\Exception $exception) {
        return back()->withError($exception->getMessage());
    }
    return redirect('pagAsignaturas')->with('success', 'Se modifico correctamente');

}
```

Figura 244. Método Update Asignaturas

Modelo de Asignaturas:

El Modelo de Asignaturas permite enlazar la capa de manejo de datos (MD) para recibir y enviar información con respecto a la tabla de la base de datos llamada Asignaturas que se encuentra en PostgreSQL; la base de datos total del sistema se encuentra albergada en un gestor para el manejo de base de datos, que en este caso es pgAdmin 4.

El Modelo de Asignaturas se lo utiliza dentro del Controlador de Asignaturas para llamar al ORM por defecto de Laravel llamado Eloquent, que interpreta procesos de bases de datos escritos con instrucciones php; los cuales posteriormente son traducidos a sentencias SQL que realizan operaciones dentro de la tabla Asignaturas en la base de datos.

Es importante recordar que, para construir el Modelo de Asignaturas, se debe tomar en cuenta el nombre de los atributos como se encuentran en la tabla Asignaturas de la base de datos y la clave primaria que identifica a la tabla Asignaturas.

```
class AsignaturaModel extends Model
{
    protected $table = 'asignaturas';
    protected $primaryKey = 'codasignatura';
    public $timestamps = false;
    protected $fillable = ['codcarrera','descasignatura','credasignatura','nivelasignatura','objeasignatura','resulapreasignatura','caracapreasignatura'];
}
```

Figura 245. Asignaturas Model

Rutas de Asignaturas:

En el archivo de Rutas de Asignaturas se encuentran las direcciones que permiten conectar la capa lógica (DP) con las vistas (GUI) de los Asignaturas dentro del sistema.

```
//ASIGNATURAS
Route::view('/pagAsignaturas','asignaturas');
Route::view('/pagAsignaturasxEstudiante','asignaturasxEstudiante');
Route::view('/pagAsignaturasxProfesor','asignaturasxProfesor');
Route::resource('pagAsignaturas','AsignaturaController');
Route::post('actualizarAsignatura', 'AsignaturaController@update');
```

Figura 246. Asignaturas Formularios Route Path

Vista Principal de Asignaturas:

La Vista Principal de Asignaturas permite mostrar todos los datos que se encuentran almacenados en la tabla Asignaturas de la Base de Datos del sistema. Los datos del registro se reflejan en una tabla tipo CRUD que permite seleccionar el tipo de operación que se quiere realizar con respecto a los datos (insertar, borrar y actualizar). Las operaciones se ven reflejadas en forma de botones.

```
<a class="btn btn-primary" href="/pagAsignaturas/create">Nuevo</a>

<table id="tablita" class="table table-striped " cellspacing="0" width="100%">
<thead>
        <tr>
            <th>Carrera</th>
            <th>Codigo Asignatura</th>
            <th>Asignatura</th>
            <th>Creditos</th>
            <th>Nivel</th>
            <th>Modificar</th>
            <th>Eliminar</th>
        </tr>
    </thead>
    <tbody id="cuerpo">
```

Figura 247. Asignaturas Blade 1

Como se puede observar en la Figura 106, es necesario crear una tabla que sirva como almacén para los registros recuperados de la tabla Asignaturas que se encuentra en la base de datos del sistema. Sin embargo, se debe usar un tag <a> con una propiedad href para poder recuperar la información en un solo sentido; eso sí, especificando el url exacto de la ruta definida en el archivo web.php.

```
<td>{{$asignatura['codcarrera']}}</td>
<td>{{$asignatura['codasignatura']}}</td>
<td>{{$asignatura['descasignatura']}}</td>
<td>{{$asignatura['credasignatura']}}</td>
<td>{{$asignatura['nivelasignatura']}}</td>
```

Figura 248. Asignaturas Blade 2

Es importante definir las columnas <td> en base al nombre declarado en la función Index del controlador. Ya que caso contrario no se va a poder recuperar los datos y cargarlos en las celdas de la tabla.

```
<script>
$(document).ready(function() {


    $('#tablita').dataTable();


} );
</script>
```

Figura 249. Asignaturas Blade 3

La librería Data Tables se utiliza para poder cargar de forma dinámica y con posibilidad de obtener un campo de búsqueda por columna y coincidencia los datos de una consulta SQL en la base de datos. Aquí se considera el tipo de notación JSON para indexar campos y así realizar consultas que reordenen la tabla por id.

Vista Ingresar Asignaturas:

La Vista Ingresar Asignaturas permite ingresar un nuevo registro en la tabla Asignaturas de la Base de Datos del sistema.

```
<form    method="post" action="{{URL::to('/pagAsignaturas')}}" enctype="multipart/form-data">
@csrf
    <div class="form-horizontal">

    <div class="form-group row">
        <label class="control-label col-sm-3" for="codcarrera">Universidad-Carrera</label>
        <div class="col-sm-6">
            <select name="carreraSelect" class="form-control" id="codcarrera" name="codcarrera">
            <option>Seleccionar Universidad - Carrera</option>
                @foreach ($carreras as $carrera)
                <option value="{{$carrera->codcarrera}}" > {{$carrera->descuniversidad}} -  {{$carrera->descsede}} -  {{$carrera->descfacultad}} -  {{$carrera->descescuela}} -  {{$carrera->descarrera}}
                @endforeach
            </select>
        </div>
    </div>

    <div class="form-group row">
        <label class="control-label col-sm-3" for="codasignatura">Código</label>
        <div class="col-sm-6">
            <input type="text" id="desccarrera" class="form-control" name="codasignatura">
        </div>
    </div>

    <div class="form-group row">
        <label class="control-label col-sm-3" for="descasignatura">Asignatura</label>
        <div class="col-sm-6">
            <input type="text" id="descasignatura" class="form-control" name="descasignatura">
            </div>
    </div>
```

Figura 250. Asignaturas Create Blade 1

Es necesario colocar la instrucción {{ csrf_field()}} que permite denegar la falsificación de petición en sitios cruzados (se origina cuando un sitio realiza peticiones en nombre de un usuario sobre nuestro sitio). Al denegar la petición permite acceder a las rutas declaradas en el web.php con post y get para ingresar, recuperar y alterar información dentro de un Blade al usar los tags <input> , <select>, etc.

```
<select class="form-control" id="nivelasignatura" name="nivelasignatura">
    <option>1</option>
    <option>2</option>
    <option>3</option>
    <option>4</option>
    <option>5</option>
    <option>6</option>
    <option>7</option>
    <option>8</option>
    <option>9</option>
    <option>10</option>
    <option>11</option>
    <option>12</option>
    <option>13</option>
    <option>14</option>
    <option>15</option>
</select>
    </div>
</div>


<div class="form-group row">
    <label class="control-label col-sm-3" for="objeasignatura">Objetivos</label>
    <div class="col-sm-6">
    <textarea class="form-control" rows="5" name="objeasignatura" id='objeasignatura' value=""></textarea>
        </div>
</div>

<div class="form-group row">
    <label class="control-label col-sm-3" for="resulapreasignatura">Resultados de Apredizaje</label>
    <div class="col-sm-6">
    <textarea class="form-control" rows="5" name="resulapreasignatura" id='resulapreasignatura' value=""></textarea>
        </div>
</div>
```

Figura 251. Asignaturas Create Blade 2

Finalmente, es importante definir el nombre de cada tag <input> y <select> en base al nombre declarado en la función Index del controlador. Ya que caso contrario no se va a poder recuperar los datos y posteriormente ingresar el registro.

Vista Modificar Asignaturas:

La Vista Modificar Asignaturas permite modificar un registro previamente ingresado dentro de la tabla Asignaturas de la Base de Datos del sistema.

```
<form    method="post" action="{{URL::to('/actualizarAsignatura')}}" enctype="multipart/form-data">
@csrf
    <div class="form-horizontal">

    @foreach ($asignaturas as $asignatura)
    <div class="form-group row">
        <label class="control-label col-sm-3" for="unies">Universidad-Carrera</label>
        <div class="col-sm-7">
        <select class="form-control" id="unies" name="unies">
        @foreach ($cars as $car)
    <option value="{{$car->codcarrera}}" > {{$car->descuniversidad}} -  {{$car->descsede}} -  {{$car->descfacultad}} -  {{$car->descescuela}} -  {{$car->desccarrera}}</option>
    @endforeach
    @foreach ($carreras as $carrera)
    <option value="{{$carrera->codcarrera}}" > {{$carrera->descuniversidad}} -  {{$carrera->descsede}} -  {{$carrera->descfacultad}} -  {{$carrera->descescuela}} -  {{$carrera->desccarrera}}</option>
    @endforeach
        </select>
            </div>
    </div>



    <div class="form-group row">
        <label class="control-label col-sm-3" for="codasignatura">Codigo</label>
        <div class="col-sm-6">
            <input type="text" id="desccarrera" class="form-control" name="codasignatura" value="{{$asignatura->codasignatura}}" readonly>
            </div>
    </div>

    <div class="form-group row">
        <label class="control-label col-sm-3" for="descasignatura">Asignatura</label>
        <div class="col-sm-6">
            <input type="text" id="descasignatura" class="form-control" name="descasignatura" value="{{$asignatura->descasignatura}}">
            </div>
    </div>
```

Figura 252. Asignaturas Edit Blade 1

Como se pudo observar, es necesario colocar la instrucción {{ csrf_field()}} que permite denegar la falsificación de petición en sitios cruzados (se origina cuando un sitio realiza peticiones en nombre de un usuario sobre nuestro sitio). Al denegar la petición permite acceder a las rutas declaradas en el web.php con post y get para ingresar, recuperar y alterar información dentro de un Blade al usar los tags <input> , <select>, etc.

```
<div class="form-group row">
<label class="control-label col-sm-3" for="nivelasignatura">Nivel</label>
<div class="col-sm-6">
<select class="form-control" id="nivelasignatura" name="nivelasignatura">
    <option>{{$asignatura->nivelasignatura}}</option>
    <option>1</option>
    <option>2</option>
    <option>3</option>
    <option>4</option>
    <option>5</option>
    <option>6</option>
    <option>7</option>
    <option>8</option>
    <option>9</option>
    <option>10</option>
    <option>11</option>
    <option>12</option>
    <option>13</option>
    <option>14</option>
    <option>15</option>
</select>
    </div>
</div>


<div class="form-group row">
    <label class="control-label col-sm-3" for="objeasignatura">Objetivos</label>
    <div class="col-sm-6">
    <textarea class="form-control" rows="5" name="objeasignatura" id='objeasignatura' value="{{$asignatura->objeasignatura}}">{{$asignatura->objeasignatura}}</textarea>
        </div>
</div>

<div class="form-group row">
    <label class="control-label col-sm-3" for="resulapreasignatura">Resultados de Apredizaje</label>
    <div class="col-sm-6">
    <textarea class="form-control" rows="5" name="resulapreasignatura" id='resulapreasignatura' value="{{$asignatura->resulapreasignatura}}">{{$asignatura->resulapreasignatura}}</textarea>
        </div>
</div>
```

Figura 253. Asignaturas Edit Blade 2

Es importante definir el nombre de cada tag <input> y <select> en base al nombre declarado en la función Index del controlador. Ya que caso contrario no se va a poder recuperar los datos para cargar los valores del registro previamente ingresado en la tabla CRUD y posteriormente actualizar el registro. No se debe olvidar que la/las claves primarias se mantienen fijas hasta acabar la edición de los campos seleccionables o editables.

Vista Eliminar Asignaturas:

La Vista Eliminar Asignaturas permite eliminar un registro previamente ingresado dentro de la tabla Asignaturas de la Base de Datos del sistema. Esta opción se encuentra disponible directamente desde la tabla CRUD en la Vista Principal de Asignaturas.

```
<form action="{{action('AsignaturaController@destroy', $asignatura['codasignatura'])}}" method="post">
  @csrf
  <input name="_method" type="hidden" value="DELETE">
  <button class="btn btn-danger" type="submit">Eliminar</button>
</form>
```

Figura 254. Asignaturas Blade – Eliminar Registro

Es importante recordar que para borrar un registro se necesita confirmar el proceso de borrado. Para esto se utiliza una propiedad del tag <form> llamada onsubmit, que permite confirmar el proceso de borrado; que por cierto en el caso de Asignaturas no es lógico, pero impide borrar en cascada si existe clave foránea de la tabla Asignaturas en otro módulo.

Interfaces Asignaturas:

Las Interfaces de Asignaturas se encuentran separadas por cada funcionalidad descrita en las Vistas de Asignaturas.

Interface Principal de Asignaturas (Tabla CRUD):

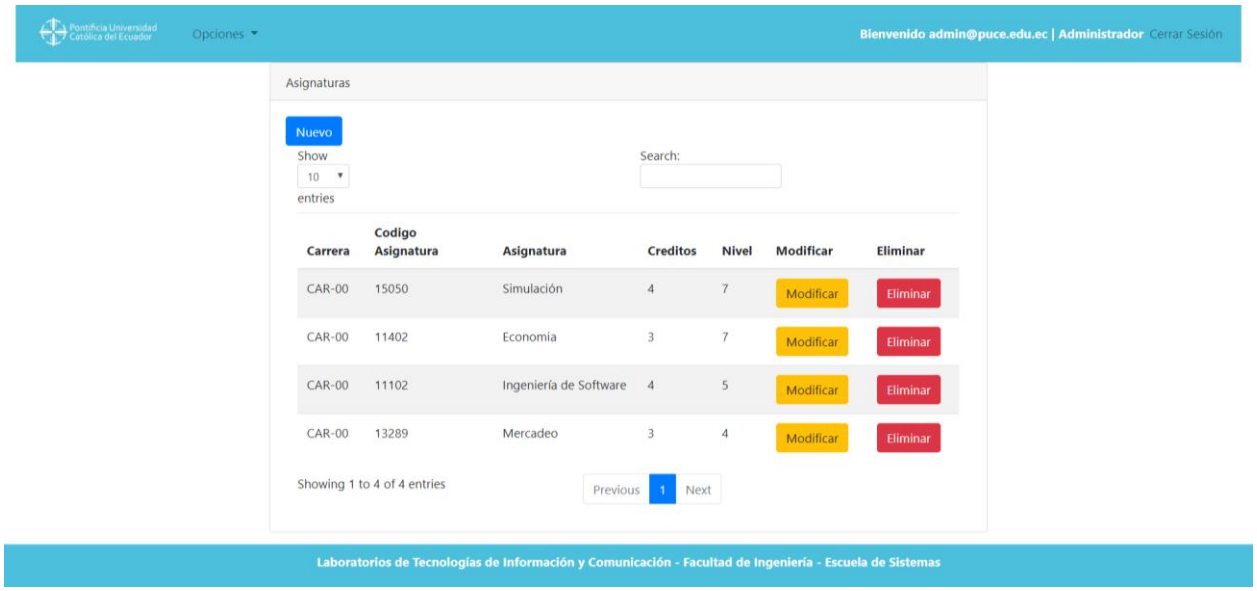


Figura 255. Página de Asignaturas Principal

Como se puede observar, la Interface Principal de Asignaturas tiene un diseño de tabla Responsivo que permite manejar las 4 operaciones básicas que componen a un CRUD (crear, borrar, actualizar y consultar).

Interface Ingreso Asignaturas por Formulario:

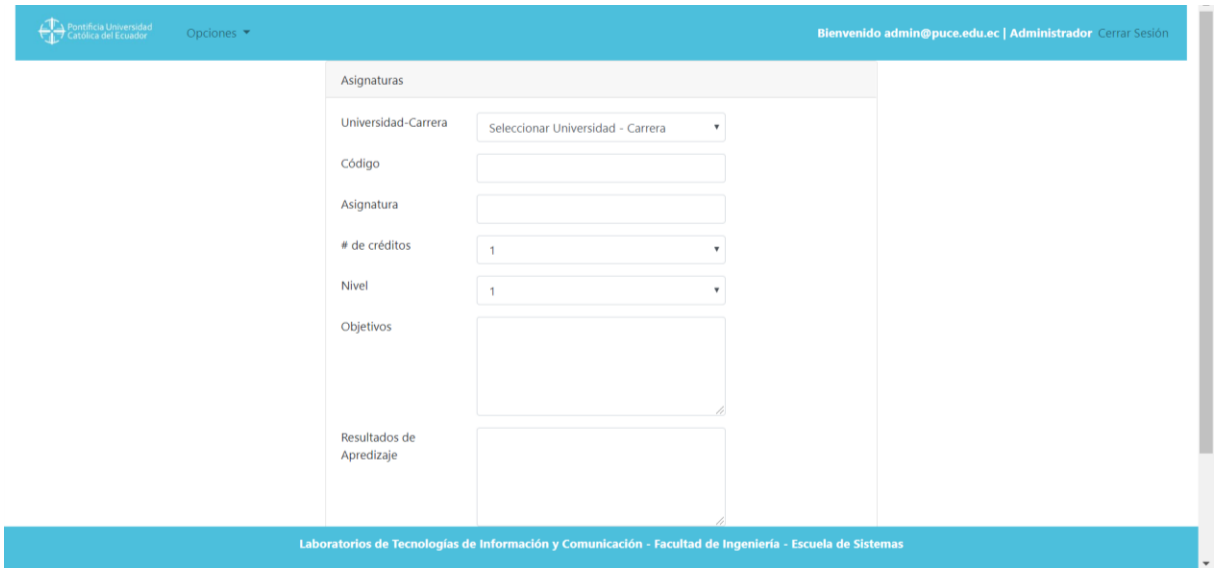


Figura 256. Página de Asignaturas Crear

Como se puede observar, en la Interface de Ingreso de Asignaturas se colocan los datos para agregar un nuevo registro en la tabla de Asignaturas. Al presionar guardar se agrega un nuevo registro y se coloca en la tabla CRUD. La clave primaria de esta tabla corresponde al código de una Asignatura.

Interface Edición Asignaturas:



Figura 257. Página de Asignaturas Editar

En la Interface de Edición de Asignaturas se colocan los datos recuperados de un registro previamente ingresados, para editar y actualizar un registro ya existente de la tabla Asignaturas. Al presionar guardar se modifica el registro ya en base al código de Asignatura con formato no editable y se coloca en la tabla CRUD.

Requisitos, Relaciones, Procesos de Grabado, Validaciones y Estructuras que se Deben Tomar en Cuenta para la Construcción de Interfaces:

Asignaturas General:

Sistema: Agenda Electrónica.

Tabla: ASIGNATURAS.

Relaciones: asig_tiene_unid, asig_tiene_tem, asig_tiene_cont, asig_tiene_asigxestu, sig_tiene_asigxprof

Requisitos Previos: Validar existencia de Carreras ingresadas anteriormente

Proceso Grabado: Grabar datos en tabla ASIGNATURAS.

Validaciones:

Se selecciona la descripción de Universidad, Sede, Facultad, Escuela en base al combo box Universidad-Escuela, con esta selección se cargan las Carreras existentes en el ComboBox de Carreras. Ej.: P.U.C.E - Quito - Ingeniería - Sistemas – Sistemas. Se debe almacenar el código de Carrera en la base de datos.

Se ingresa el código de la Asignatura en base al código establecido por la PUCE, el código es un número de 5 dígitos enteros.

Al ingresar un código de Asignatura se valida que este no exista, ya que si existe se deben rellenar los campos con los resultados del registro presente en la tabla ASIGNATURAS.

Se ingresa el nombre de la Asignatura establecido por la PUCE en base al código previamente mencionado.

Se ingresa el número de créditos que posee la Asignatura, en base al número de créditos asignados por la PUCE en cada Carrera.

Se selecciona el nivel actual en el que se encuentra asignada la materia, solo se puede seleccionar hasta el número 15.

Se ingresa el objetivo general de la Asignatura, descrito en el plan de estudios de la Asignatura.

Se ingresa el resultado de aprendizaje de la Asignatura, descrito en el plan de estudios de la Asignatura.

Se ingresa la caracterización de aprendizaje de la Asignatura, descrito en el plan de estudios de la Asignatura.

Para borrar una Asignatura, debe validar que no exista una ASIGNATURAXESTUDIANTES, ASIGNATURASXPROFESOR, TEMAS_ESTUDIO, UNIDADES_ESTUDIO y CONTENIDOS relacionados a dicha Asignatura.

Para consultar las Asignaturas existentes se presiona el botón consultar, que realiza una consulta general y muestra las 10 primeras Asignaturas en una tabla que se despliega en una ventana emergente.

El estado actual de la Asignatura se verifica en base al Periodo académico del Estudiante y a la materia que se encuentra tomando, es decir no es editable, cambia automáticamente y solo se lo puede observar.

Botones:

Agrega una nueva fila para ingresar un nuevo registro.

Para confirmar y guardar en la base de datos se presiona el botón de guardar ya sea que se esté ingresando o modificando.

El botón borrar permite eliminar un registro ingresado previamente de acuerdo con las validaciones definidas. Antes de eliminar se muestra un mensaje de confirmación si es que de verdad desea eliminar ese registro.

El botón salir permite regresar a la pantalla principal.



La barra de búsqueda despliega el registro que se encuentra realizando una consulta en base a cualquier atributo de la tabla.

F14: Módulo Gestionar Contenido

La Gestión de Contenidos en el sistema permite manejar diversos Contenidos digitales, tales como audios, videos e imágenes; estos Contenidos son subidos por los Profesores en base a cada Tema de Estudio. El propósito de los Contenidos es facilitar e incentivar a un Estudiante en el proceso de adquisición de conocimiento significativo con respecto a una Asignatura. Dichos Contenidos, además, le permitirán comprender al Estudiante de mejor manera los Temas de Estudio de una Asignatura y por lo tanto tomar Anotaciones importante que relacionen conocimiento específico de lo aprendido para su posterior uso.

Controlador de Contenido:

El Controlador de Contenido es el lugar que permite definir los métodos en la capa lógica (DP) que utilizan las peticiones del archivo routes.php→web.php, para realizar acciones dentro de una vista con respecto al ingreso, edición, borrado y consulta de información con respecto a Contenido.

Index:

Método que permite cargar Contenido en la tabla tipo CRUD. Además de enviar un arreglo de Contenidos y un arreglo de Temas de Estudio para cargar información en <input> y <label> tags. De existir algún error en conexión, se envía un mensaje de error con la excepción encontrada.

```php
public function index()//consultar
{
    try{
    $contenidos = ContenidosModel::select('*')->get();
    $temas = DB::table('temas_estudio')->select('*')->get();
    }catch(\Exception $e)
    {
        return back()->withError($e->getMessage());
    }
    return view('indexContenidos',['contenidos'=>$contenidos, 'tema'=> $temas]);

}
```

Figura 258. Método Index Contenido

Create:

Método que permite abrir y redirigir a la Vista Ingresar Contenidos. Además de enviar un arreglo de Contenidos, Temas de Estudio, Asignaturas y Carreras para cargar información en <input> y <label> tags. De existir algún error en conexión, se envía un mensaje de error con la excepción encontrada.

```php
public function create()
{
    try{
    $contenidos = DB::table('contenidos')->select('*')->get();
    $tema = DB::table('temas_estudio')->select('*')->get();
    $asignaturas = DB::table('asignaturas')->select('*')->get();
    $contenidos = DB::table('contenidos')->select('*')->get();

        $carreras = DB::select('SELECT carreras.codcarrera, carreras.desccarrera, escuelas.descescuela, facultades.descfacultad, sedes.descsede, universidades.descuniversidad
                FROM carreras natural join escuelas natural join facultades natural join facultadesxsedes natural join sedes natural join universidades;');
    }catch(\Exception $e)
    {
        return back()->withError($e->getMessage());
    }
    return view('contenidosNuevo', compact('carreras', 'asignaturas', 'tema', 'contenidos'));
}
```

Figura 259. Método Create Contenido

Store:

Método que permite almacenar un nuevo registro de Contenidos dentro de la base de datos del sistema mediante el uso de un formulario que se encuentra dentro de la vista Crear Contenidos.

El primer paso para ingresar los Contenidos es recibir toda la información del formulario y almacenarlas en variables que posteriormente permitirán construir el código de cada Contenido digital; el código es una combinación del código de Contenido añadido un '.' y añadida la extensión del archivo de origen. Subsiguientemente, se verifica que cada uno de los recursos digitales no sean nulos y se construye el código de cada uno (este código se ingresa como referencia al archivo digital dentro de la base de datos). Finalmente, se mueve el Contenido digital del formulario a la carpeta 'images' y/o 'audio' y/o 'video' (dependiendo del tipo de Contenido digital ingresado) que se encuentra como recurso público embebido dentro de la carpeta del Sistema de Agenda Electrónica del Estudiante y se procede a ingresar el registro en la base de datos con una sentencia SQL compuesto por un arreglo.

De existir algún error en conexión o si se intenta ingresar un registro duplicado, se envía un mensaje de error con la excepción encontrada.

```
public function store(Request $request)
{
    try{
    $codtema = $request->get('tema');
    $codasignatura = $request->get('asignatura');
    $codcontenido = $request->get('codcontenido');
    $numero = substr($codcontenido, -1);
    if ($numero == null) {
        $numero = 1;
    } else {
        $numero += 1;
    }
    $codcontenido = $request->get('tema');
    $codcontenido = $codcontenido . '-C' . $numero;

        $this->validate($request, [
            'imagencontenido' => 'mimes:jpeg,jpg|image'
        ]);


        $audio = $request->file('audiocontenido');
        $video = $request->file('videocontenido');
        $image = $request->file('imagencontenido');


        if($image)
        {
            $imagencontenido = $codcontenido . '.' . $image->getClientOriginalExtension();
            $image->move(public_path('images'), $imagencontenido);
        }
        else
            $imagencontenido = "";

        if($audio)
        {
            $audiocontenido = $codcontenido . '.' . $audio->getClientOriginalExtension();
            $audio->move(public_path('audio'), $audiocontenido);
        }
        else
            $audiocontenido = "";
```

```
    if($video)
    {

        $videocontenido = $codcontenido . '.' . $video->getClientOriginalExtension();
        $video->move(public_path('video'), $videocontenido);
    }
    else
        $videocontenido = "";


        $textocontenido = $request->get('textocontenido');

        if(!$textocontenido)
            $textocontenido = "";

        $infoapoyocontenido = $request->get('infoapoyo');

        if(!$infoapoyocontenido)
            $infoapoyocontenido = "";

$data = array('codcontenido' => $codcontenido, 'codtema' => $codtema, 'codasignatura' => $codasignatura,
    'textocontenido' => $textocontenido, 'imagencontenido' => $imagencontenido, 'videocontenido' => $videocontenido,
    'audiocontenido' => $audiocontenido, 'infoapoyocontenido' => $infoapoyocontenido);
DB::table('contenidos')->insert($data);

}catch(\Exception $e)
{
    return back()->withError($e->getMessage());
}
return redirect('contenidos');
```

Figura 260. Método Store Contenido

Destroy:

Método que permite eliminar un registro existente de la tabla Contenidos en la base de datos del sistema. Para poder eliminar se recibe el código de Contenido, y si se logra eliminar el registro exitosamente, se manda un mensaje a consola que diga registro borrado exitosamente. Caso contrario y si no se puede eliminar debido a que se irrumpe con la integridad referencial de las tablas y sus registros, se manda un mensaje que dice que no se puede eliminar Contenido con su respectiva excepción.

```
public function destroy($codcontenido)//modificar
{
    try{
    $contenido=DB::table('contenidos')->select('*')->where('codcontenido','=', $codcontenido)->get();
    $contenido = json_decode($contenido, true)[0];
    if($contenido['imagencontenido']!="")
    {
        $imagencontenido = "images/".$contenido['imagencontenido'];
        unlink($imagencontenido) or die("Couldn't delete file");
    }
    if($contenido['audiocontenido']!="")
    {
        $audiocontenido = "audio/".$contenido['audiocontenido'];
        unlink($audiocontenido) or die("Couldn't delete file");
    }
    if($contenido['videocontenido']!="")
    {
        $videocontenido = "video/".$contenido['videocontenido'];
        unlink($videocontenido) or die("Couldn't delete file");
    }

    DB::table('contenidos')->where('codcontenido','=', $codcontenido)->delete();
}catch(\Exception $e)
{
    return back()->withError($e->getMessage());
}
    return redirect('contenidos')->with('success','Se elimino correctamente');

}
```

Figura 261. Método Destroy Contenido

Edit:

Método que permite abrir y redirigir a la Vista Editar Contenidos, para esto se recibe el código de Contenido, que en base a una consulta SQL, permita recuperar el objeto Tema de Estudio y Asignaturas que posteriormente se va a editar en la base de datos y facilite enviar un objeto Carreras, Contenidos, Tema de Estudio y Asignaturas con los parámetros relacionados al Contenido para cargar los combo box acumulativos en la Vista Editar Contenidos y así poder gestionar las relaciones del ingreso en cascada. De existir

algún error en conexión o si no se encuentra un registro usando el código de Contenidos recibido, se envía un mensaje de error con la excepción encontrada.

```
public function edit($codcontenido)//abre la ventana para modificar
{
    try{

    $contenidos = DB::table('contenidos')->select('*')->where('codcontenido',$codcontenido)->get();
    $contenidos = json_decode($contenidos, true)[0];

    $tema = DB::table('temas_estudio')->select('*')->where('codtema', $contenidos['codtema'])->get();
    $tema = json_decode($tema, true)[0];

    $asignaturas = DB::table('asignaturas')->select('*')->where('codasignatura', $tema['codasignatura'])->get();
    $asignaturas = json_decode($asignaturas, true)[0];


    $carreras = DB::select('SELECT carreras.codcarrera, carreras.desccarrera, escuelas.descescuela, facultades.descfacultad, sedes.descsede, universidades.descuniversidad
    FROM carreras natural join escuelas natural join facultades natural join facultadesxsedes natural join sedes natural join universidades;');

    }catch(\Exception $e)
    {
        return back()->withError($e->getMessage());
    }
    return view('contenidosEdit',compact('carreras','contenidos',
    'asignaturas','tema'));

}
```

Figura 262. Método Edit Contenido

Update:

Método que permite actualizar un registro existente de la tabla Contenidos en la base de datos del sistema, para esto se utiliza un objeto de tipo Request recibido a través de la vista Editar Contenidos.

El primer paso para actualizar los Contenidos es recibir toda la información del formulario y almacenarlas en variables que posteriormente permitirán construir el código de cada Contenido digital; el código es una combinación del código de Contenido añadido un '.' y añadida la extensión del archivo de origen. Subsiguientemente, se verifica que cada uno de los recursos digitales no sean nulos y se construye el código de cada uno (este código se ingresa como referencia al archivo digital dentro de la base de datos). Finalmente, elimina y mueve el nuevo Contenido digital del formulario de la carpeta 'images' y/o 'audio' y/o 'video' (dependiendo del tipo de Contenido digital existente) que

se encuentra como recurso público embebido dentro de la carpeta del Sistema de Agenda Electrónica del Estudiante y se procede a actualizar el registro en la base de datos con una sentencia SQL compuesto por un arreglo.

De existir algún error en conexión o si no se encuentra un registro usando el código de Asignaturas o de Carreras recibido por el objeto de tipo Request, se envía un mensaje de error con la excepción encontrada.

```
public function update(Request $request)
{
   try{
    $codtema = $request->get('tema');
    $codasignatura = $request->get('codasignatura');
    $codcontenido = $request->get('codcontenido');
    $numero = substr($codcontenido, -1);

    $contenidos=DB::table('contenidos')->select('*')->where('codcontenido','=', $codcontenido)->get()
    $contenidos = json_decode($contenidos, true)[0];

        $this->validate($request, [
            'imagencontenido' => 'mimes:jpeg,jpg|image'
        ]);

    $imagencontenido = $contenidos['imagencontenido'];
    $audiocontenido = $contenidos['audiocontenido'];
    $videocontenido = $contenidos['videocontenido'];

    $codcontenido = $request->get('tema');
    $codcontenido = $codcontenido.'-C'. $numero;

    DB::table('contenidos')->where('codcontenido','=', $codcontenido)->delete();

    $audio = $request->file('audiocontenido');
    $video = $request->file('videocontenido');
    $image = $request->file('imagencontenido');

    if($image)
    {
        $imagencontenido = $codcontenido . '.' . $image->getClientOriginalExtension();
        $image->move(public_path('images'), $imagencontenido);
    }

    if($audio)
    {
        $audiocontenido = $codcontenido . '.' . $audio->getClientOriginalExtension();
        $audio->move(public_path('audio'), $audiocontenido);
```

```
    if($video)
    {
        $videocontenido = $codcontenido . '.' . $video->getClientOriginalExtension();
        $video->move(public_path('video'), $videocontenido);
    }

    $textocontenido = $request->get('textocontenido');

            if(!$textocontenido)
                $textocontenido = "";

            $infoapoyocontenido = $request->get('infoapoyo');

            if(!$infoapoyocontenido)
                $infoapoyocontenido = "";

    $data = array('codcontenido' => $codcontenido, 'codtema' => $codtema, 'codasignatura' => $codasignatura,
        'textocontenido' => $textocontenido, 'imagencontenido' => $imagencontenido, 'videocontenido' => $videocontenido,
        'audiocontenido' => $audiocontenido, 'infoapoyocontenido' => $infoapoyocontenido);
    DB::table('contenidos')->insert($data);

}catch(\Exception $e)
{
    return back()->withError($e->getMessage());
}
    return redirect('contenidos');


}
```

Figura 263. Método Update Contenido

Modelo de Contenidos:

El Modelo de Contenidos permite enlazar la capa de manejo de datos (MD) para recibir y enviar información con respecto a la tabla de la base de datos llamada Contenidos que se encuentra en PostgreSQL; la base de datos total del sistema se encuentra albergada en un gestor para el manejo de base de datos, que en este caso es pgAdmin 4.

El Modelo de Contenidos se lo utiliza dentro del Controlador de Contenidos para llamar al ORM por defecto de Laravel llamado Eloquent, que interpreta procesos de bases de datos escritos con instrucciones php; los cuales posteriormente son traducidos a sentencias SQL que realizan operaciones dentro de la tabla Contenidos en la base de datos.

```
class ContenidosModel extends Model
{
    protected $table = 'contenidos';
    public $timestamps = false;
}
```

Figura 264. Contenidos Model

Rutas de Contenido:

En el archivo de Rutas de Contenido se encuentran las direcciones que permiten conectar la capa lógica (DP) con las vistas (GUI) de los Contenidos dentro del sistema.

```
//CONTENIDOS
Route::view('/pagContenidos','contenidos');
Route::resource('contenidos', 'ContenidosController');
Route::post('actualizarContenidos', 'ContenidosController@update');
```

Figura 265. Contenidos Formularios Route Path

Vista Principal de Contenido:

La Vista Principal de Contenido permite mostrar todos los datos que se encuentran almacenados en la tabla Contenidos de la Base de Datos del sistema. Los datos del registro se reflejan en una tabla tipo CRUD que permite seleccionar el tipo de operación que se quiere realizar con respecto a los datos (insertar, borrar y actualizar). Las operaciones se ven reflejadas en forma de botones.

```
<table id="myTable" class="table table-striped">
    <thead>
      <tr>
        <th>Tema</th>
        <th>Información de Apoyo</th>
        <th>Imagen Contenido</th>
        <th>Audio Contenido</th>
        <th>Video Contenido</th>

        <th>Modificar</th>
        <th>Eliminar</th>
      </tr>
    </thead>
    <tbody>
```

Figura 266. Contenidos Blade 1

Como se puede observar, es necesario crear una tabla que sirva como almacén para los registros recuperados de la tabla Contenidos que se encuentra en la base de datos del sistema. Sin embargo, se debe usar un tag <a> con una propiedad href para poder recuperar la información en un solo sentido; eso sí, especificando el url exacto de la ruta definida en el archivo web.php.

```
@foreach($contenidos as $contenido)

<tr>
  <td>@foreach($tema as $dato)
  @if($dato->codtema == $contenido->codtema)
  {{$dato->desctema}}
  @endif
  @endforeach</td>
  <td>{{$contenido->infoapoyocontenido}}</td>
  <td>{{$contenido->imagencontenido}}</td>
  <td>{{$contenido->audiocontenido}}</td>
  <td>{{$contenido->videocontenido}}</td>
```

Figura 267. Contenidos Blade 2

Como se puede observar, es importante definir las columnas <td> en base al nombre declarado en la función Index del controlador. Ya que caso contrario no se va a poder recuperar los datos y cargarlos en las celdas de la tabla.

```
<script>
$(document).ready(function() {
    $('#myTable').DataTable({
      "pageLength": 10
    });
} );

</script>
```

Figura 268. Contenidos Blade 3

Finalmente, y como se puede observar, la librería Data Tables se utiliza para poder cargar de forma dinámica y con posibilidad de obtener un campo de búsqueda por columna y coincidencia los datos de una consulta SQL en la base de datos. Aquí se considera el tipo de notación JSON para indexar campos y así realizar consultas que reordenen la tabla por id.

Vista Ingresar Contenidos:

La Vista Ingresar Contenidos permite ingresar un nuevo registro en la tabla Asignaturas de la Base de Datos del sistema.

```
<form method="post" action="{{url('contenidos')}}" class="form-horizontal" data-toggle="validator"
    enctype="multipart/form-data">
    @csrf
    <div class="form-horizontal">
        <input type="hidden" id="codcontenido" class="form-control" name="codcontenido" >

        <div class="form-group">
            <label class="control-label col-sm-3" for="unies">Universidad-Carrera</label>
            <div class="col-sm-6">
                <select class="form-control" id="codcarrera" name="codcarrera" onchange="cambioCarrera()">
                    @foreach ($carreras as $carrera)
                    <option value="{{$carrera->codcarrera}}">{{$carrera->descuniversidad}} - {{$carrera->descsede}} -
                        {{$carrera->descfacultad}} - {{$carrera->descescuela}}
                        - {{$carrera->desccarrera}}</option>
                    @endforeach
                </select>
            </div>
        </div>
```

Figura 269. Contenidos Create Blade 1

Es necesario colocar la instrucción {{ csrf_field()}} que permite denegar la falsificación de petición en sitios cruzados (se origina cuando un sitio realiza peticiones en nombre de un usuario sobre nuestro sitio). Al denegar la petición permite acceder a las rutas declaradas en el web.php con post y get para ingresar, recuperar y alterar información dentro de un Blade al usar los tags <input> , <select>, etc.

```
<div class="form-group">
    <label class="control-label col-sm-3" for="asignatura">Asignatura</label>
    <div class="col-sm-6">
        <select class="form-control" id="asignatura" name="asignatura" onchange="cambioAsignatura()">
            <option></option>
        </select>
    </div>
</div>

<div class="form-group">
    <label class="control-label col-sm-3" for="tema">Tema</label>
    <div class="col-sm-6">
        <select class="form-control" id="tema" name="tema" onchange="cambioTema()">
            <option></option>
        </select>
    </div>
</div>

<div class="form-group" style="width:1000px; margin-left: 1%;">
    <h6>Texto</h6>
    <textarea name="textocontenido" id="summernote"></textarea>
</div>

<div class="form-group" style="width:1000px; margin-left: 1%;">
    <h6>Información de Apoyo</h6>
    <textarea style= "height: 100px;
    width: 1000px;" name="infoapoyo" id="infoapoyo"></textarea>
</div>

<div class="form-group" style="width:800px; margin-left: 1%;">
    <h6>Imagen</h6>
    {{ csrf_field() }}
    <div class="form-group">
        <table class="table">
            <tr>
                <td><label>Seleccione la imagen</label></td>
                <td><input type="file" name="imagencontenido" /></td>
                <td><span class="text-muted">Solo imagenes con extención .jpg</span></td>
            </tr>
        </table>
    </div>
```

Figura 270. Contenidos Create Blade 2

Es importante definir el nombre de cada tag <input> y <select> en base al nombre declarado en la función Index del controlador. Ya que caso contrario no se va a poder recuperar los datos y posteriormente ingresar el registro.

Vista Modificar Contenidos:

La Vista Modificar Contenidos permite modificar un registro previamente ingresado dentro de la tabla Contenidos de la Base de Datos del sistema.

```
<form   method="post"  action="{{URL::to('/actualizarContenidos')}}" class="form-horizontal" data-toggle="valicontenidosr" enctype="multipart/form-data"
@csrf
    <div class="form-horizontal">

    <input type="hidden" id="codcontenido" class="form-control" name="codcontenido" value="{{$contenidos['codcontenido']}}">
    <input type="hidden" id="tema" class="form-control" name="tema" value="{{$tema['codtema']}}">
    <input type="hidden" id="codasignatura" class="form-control" name="codasignatura" value="{{$asignaturas['codasignatura']}}">

    <div class="form-group">
        <label class="control-label col-sm-3" for="unies">Universidad-Carrera</label>
        <div class="col-sm-6">
        <select disabled class="form-control" id="codcarrera" name="codcarrera">
                    @foreach ($carreras as $carrera)
                    @if($carrera->codcarrera == $asignaturas['codcarrera'])
                    <option value="{{$carrera->codcarrera}}">{{$carrera->descuniversidad}} - {{$carrera->descsede}} -
                        {{$carrera->descfacultad}} - {{$carrera->descescuela}}
                        - {{$carrera->desccarrera}}</option>
                        @endif
                    @endforeach
                </select>
            </div>
```

Figura 271. Contenidos Edit Blade 1

Como se pudo observar, es necesario colocar la instrucción {{ csrf_field()}} que permite denegar la falsificación de petición en sitios cruzados (se origina cuando un sitio realiza peticiones en nombre de un usuario sobre nuestro sitio). Al denegar la petición permite acceder a las rutas declaradas en el web.php con post y get para ingresar, recuperar y alterar información dentro de un Blade al usar los tags <input> , <select>, etc.

```
    <div class="form-group">
        <label class="control-label col-sm-3" for="tema">Tema</label>
        <div class="col-sm-6">
            <select disabled class="form-control" id="selecttema" name="selecttema">
                <option>{{$tema['desctema']}}</option>
            </select>
        </div>
    </div>


<div class="form-group" style="width:1000px; margin-left: 1%;">
    <h6>Texto</h6>
    <textarea id="summernote"  name="textocontenido" >
    {{$contenidos['textocontenido']}}
    </textarea>

</div>

<div class="form-group" style="width:1000px; margin-left: 1%;">
    <h6>Texto</h6>
    <textarea id="infoapoyo"  name="infoapoyo" style= "height: 100px;
        width: 1000px;">
    {{$contenidos['infoapoyocontenido']}}
    </textarea>

</div>

<div class="form-group" style="width:800px; margin-left: 1%;">
<h6>Imagen</h6>
    {{ csrf_field() }}
    <div id="contenidoImagen">
    @if($contenidos['imagencontenido'] != "")
    <img src="/images/{{$contenidos['imagencontenido']}}" width="300" />
    @endif
    </div>
```

Figura 272. Contenidos Edit Blade 2

Finalmente, es importante definir el nombre de cada tag <input> y <select> en base al nombre declarado en la función Index del controlador. Ya que caso contrario no se va a poder recuperar los datos para cargar los valores del registro previamente ingresado en la tabla CRUD y posteriormente actualizar el registro. No se debe olvidar que la/las

claves primarias se mantienen fijas hasta acabar la edición de los campos seleccionables o editables.

Vista Eliminar Contenidos:

La Vista Eliminar Contenidos permite eliminar un registro previamente ingresado dentro de la tabla Contenidos de la Base de Datos del sistema. Esta opción se encuentra disponible directamente desde la tabla CRUD en la Vista Principal de Contenidos.

```
<form action="{{action('ContenidosController@destroy', $contenido['codcontenido'])}}" method="post">
  @csrf
  <input name="_method" type="hidden" value="DELETE">
  <button class="btn btn-danger" type="submit">Eliminar</button>
</form>
```

Figura 273. Contenidos Blade – Eliminar Registro

Es importante recordar que para borrar un registro se necesita confirmar el proceso de borrado. Para esto se utiliza una propiedad del tag <form> llamada onsubmit, que permite confirmar el proceso de borrado.

Interfaces Contenidos:

Las Interfaces de Contenidos se encuentran separadas por cada funcionalidad descrita en las Vistas de Contenidos.

Interface Principal de Contenidos (Tabla CRUD):

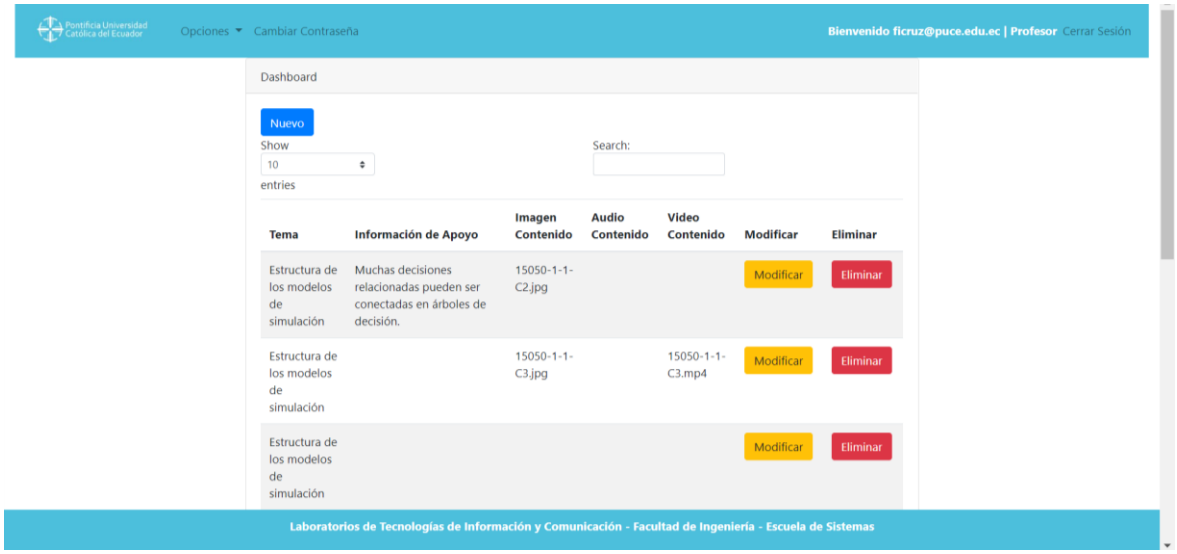


Figura 274. Página de Contenidos Principal

Como se puede observar, la Interface Principal de Contenidos tiene un diseño de tabla Responsivo que permite manejar las 4 operaciones básicas que componen a un CRUD (crear, borrar, actualizar y consultar).

Interface Ingreso Contenidos por Formulario:

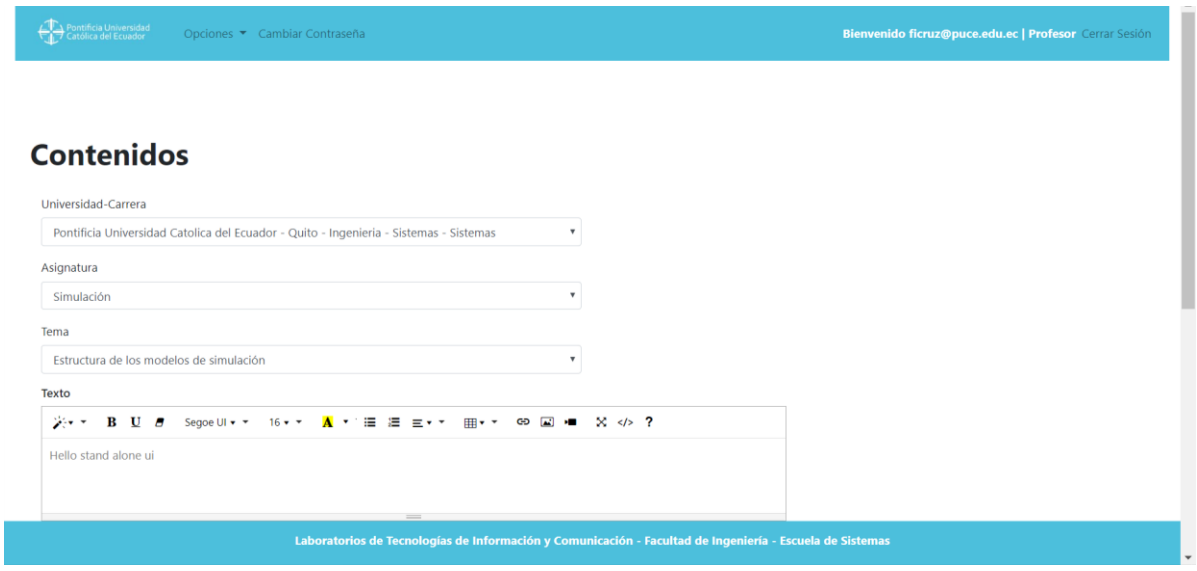


Figura 275. Página de Contenidos Crear

Como se puede observar, en la Interface de Ingreso de Contenidos se colocan los datos para agregar un nuevo registro en la tabla de Contenidos. Al presionar guardar se agrega un nuevo registro y se coloca en la tabla CRUD. La clave primaria de esta tabla corresponde al código de Contenido.

Interface Edición Contenidos:

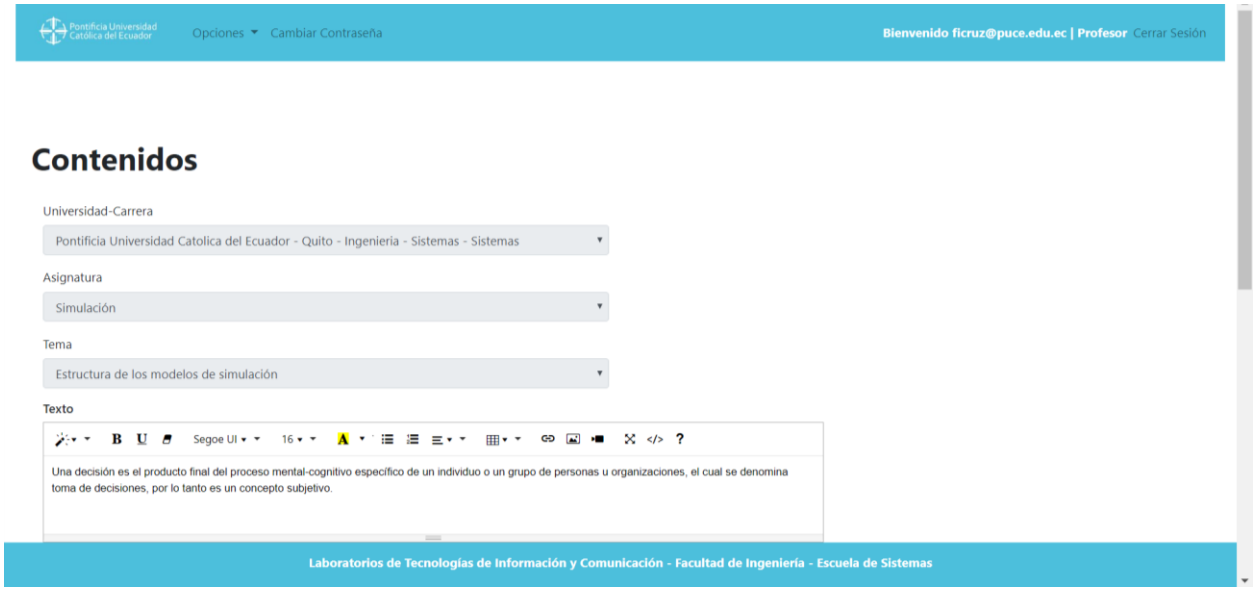


Figura 276. Página de Contenidos Editar

Como se puede observar, en la Interface de Edición de Contenidos se colocan los datos recuperados de un registro previamente ingresados, para editar y actualizar un registro ya existente de la tabla Contenidos. Al presionar guardar se modifica el registro ya en base al código de Contenido con formato no editable y se coloca en la tabla CRUD.

Requisitos, Relaciones, Procesos de Grabado, Validaciones y Estructuras que se Deben Tomar en Cuenta para la Construcción de Interfaces:

Contenidos General:

Sistema: Agenda Electrónica.

Tabla: CONTENIDOS.

Relaciones: tem_tiene_cont, cont_tiene_glo, asig_tiene_cont.

Requisitos Previos: Validar existencia de temas y Asignaturas ingresados anteriormente.

Proceso Grabado: Grabar datos en tabla CONTENIDOS.

Validaciones:

El estado actual de la Asignatura se verifica en base al Periodo académico del Estudiante y a la materia que se encuentra tomando, es decir no es editable, se cambia automáticamente y solo se lo puede observar.

Se selecciona la descripción de Universidad, Sede, Facultad, Escuela, Carrera en base al combo box Universidad-Carrera, en base a esta selección se cargan las Asignaturas existentes en el combo box de Asignaturas. Ej.: P.U.C.E - Quito - Ingeniería - Sistemas – Sistemas. Se debe almacenar el código de Carrera en la base de datos.

Se selecciona la descripción de Asignatura en base a las Asignaturas existentes en la tabla ASIGNATURAS en base a la Carrera seleccionada, con esta selección se cargan los datos el combo box de unidades. Se debe almacenar el código de Asignatura en la base de datos.

Se selecciona la descripción de la unidad en base a las unidades existentes en la tabla UNIDADES, con esta selección se cargan los datos el combo box de temas. El

nombre de la unidad que corresponde a una Asignatura en base a una consulta de todas las unidades existentes por código. Se debe almacenar el código de unidad en la base de datos.

Se selecciona el número del tema en base a los existentes en la tabla TEMAS_ESTUDIO, con esta selección se cargan los datos el combo box de Contenidos. La descripción del tema se carga en base al código de Contenido, la misma no es editable. El nombre del tema que corresponde a una unidad se realiza en base a una consulta de todos los temas existentes por código. Se debe almacenar el código de tema en la base de datos.

Se ingresa el texto del Contenido que se desea agregar, y en el caso de existir se muestra el Contenido en el área de texto.

El código de Contenido posee 18 caracteres y se debe validar en base a la combinación del código de tema + letra C + número autoincremental. Dicho código no se muestra en la pantalla, pero se autogenera y se autocarga.

Se pulsa el botón seleccionar archivo en dirección de imagen, aquí se selecciona la imagen desplegando un explorador de archivos que permita encontrarla con facilidad, seguidamente se autogenera el nombre de la imagen en base al: Código de Contenido + Número autoincremental + extensión .jpg y se muestra por pantalla el nombre en el campo de texto ubicado al lado izquierdo del botón de forma no es editable. Contemporáneamente, el archivo del disco duro de la PC en la que se realiza el ingreso es posicionado dentro del servidor mediante comandos de consola: **cd –** cambiar de directorio y **mv** – mover o renombrar un archivo. El proceso de subir una imagen opcional,

sin embargo, si ya existe una imagen en respectivo Contenido, se debe cargar solamente el nombre de esta respetando el formato en el cuadro de texto. Validar que solo se suban imágenes de tipo .jpg.

Se pulsa el botón seleccionar archivo en dirección de video, aquí se selecciona el video desplegando un explorador de archivos que permita encontrarla con facilidad, seguidamente se autogenera el nombre del video en base al: Código de Contenido + Número autoincremental + extensión .mp4 y se muestra por pantalla el nombre en el campo de texto ubicado al lado izquierdo del botón de forma no es editable. Contemporáneamente, el archivo del disco duro de la PC en la que se realiza el ingreso es posicionada dentro del servidor mediante comandos de consola: **cd –** cambiar de directorio y **mv** – mover o renombrar un archivo. El proceso de subir un video es opcional, sin embargo, si ya existe un video en respectivo Contenido, se debe cargar solamente el nombre de este respetando el formato en el cuadro de texto. Validar que solo se suban videos de tipo .mp4.

Se pulsa el botón seleccionar archivo en dirección de audio, aquí se selecciona el audio desplegando un explorador de archivos que permita encontrarla con facilidad, seguidamente se autogenera el nombre del video en base al: Código de Contenido + Número autoincremental + extensión .mp3 y se muestra por pantalla el nombre en el campo de texto ubicado al lado izquierdo del botón de forma no es editable. Contemporáneamente, el archivo del disco duro de la PC en la que se realiza el ingreso es posicionada dentro del servidor mediante comandos de consola: **cd –** cambiar de directorio y **mv** – mover o renombrar un archivo. El proceso de subir un audio es opcional, sin

embargo, si ya existe un audio en respectivo Contenido, se debe cargar solamente el nombre de este respetando el formato en el cuadro de texto. Validar que solo se suban videos de tipo .mp3.

Se pulsa el botón seleccionar archivo, aquí se selecciona el archivo desplegando un explorador de archivos que permita encontrarla con facilidad, seguidamente se autogenera el nombre del video en base al: Código de Contenido + Número autoincremental + extensión .(docx o xlsx o pdf) y se muestra por pantalla el nombre en el campo de texto ubicado al lado izquierdo del botón de forma no es editable. Contemporáneamente, el archivo del disco duro de la PC en la que se realiza el ingreso es posicionada dentro del servidor mediante comandos de consola: **cd –** cambiar de directorio y **mv** – mover o renombrar un archivo. El proceso de subir un archivo es opcional, sin embargo, si ya existe un archivo en respectivo Contenido, se debe cargar solamente el nombre de este respetando el formato en el cuadro de texto. Validar que solo se suban videos de tipo. (docx o xlsx o pdf).

Ingresar información complementaria(apoyo) dentro del cuadro de texto, esta opción también es opcional y de existir debe cargar en el cuadro de texto; esta información se carga al presionar el botón i en la parte inferior derecha de la pantalla principal.

Las anotaciones no se muestran debido a que se van añadiendo en base a la necesidad del Estudiante en la interfaz principal. Sin embargo, se debe validar que se inicialice con un texto vacío.

Para borrar un Contenido, se debe validar que no exista un Glosario relacionado a dicho Contenido.

Se debe controlar que exista por cada Contenido necesariamente y por lo mínimo un texto, archivo imagen, información complementaria, audio y/o video.

Botones:

Agrega una nueva fila para ingresar un nuevo registro.

Para confirmar y guardar en la base de datos se presiona el botón de guardar ya sea que se esté ingresando o modificando.

El botón borrar permite eliminar un registro ingresado previamente de acuerdo con las validaciones definidas. Antes de eliminar se muestra un mensaje de confirmación si es que de verdad desea eliminar ese registro.

El botón salir permite regresar a la pantalla principal.



La barra de búsqueda despliega el registro que se encuentra realizando una consulta en base a cualquier atributo de la tabla.

F19: Módulo Gestionar Asignación de Estudiantes por Asignatura

La Gestión de Asignación de Estudiantes por Asignatura en el sistema permite manejar la asignación de Estudiantes con respecto a una Asignatura y que a su vez es gestionada por un Profesor. Este proceso permite manejar la lógica académica de un Estudiante en el momento en que se inscribe en una Asignatura dictada por un Profesor. La construcción del proceso prácticamente se la realiza manteniendo un concepto de cabecera a detalle; en donde se tiene como cabecera el Estudiante y como detalle a las Asignaturas dictadas por un respectivo Profesor.

Controlador de Asignación de Estudiantes por Asignatura:

El Controlador de Asignación de Estudiantes por Asignatura es el lugar que permite definir los métodos en la capa lógica (DP) que utilizan las peticiones del archivo routes.php→web.php, para realizar acciones dentro de una vista con respecto al ingreso, edición, borrado y consulta de información con respecto a las Asignaturas pertenecientes a un Estudiante.

Index:

Método que permite cargar Asignación de Estudiantes por Asignatura en la tabla tipo CRUD. Además de enviar un arreglo de Asignaturas por Estudiantes para cargar información en <input> y <label> tags. De existir algún error en conexión, se envía un mensaje de error con la excepción encontrada.

```
public function index()//consultar
{
    try{

    $estudiantes = AsignaturasxestudianteModel::select('*')->get();
    }catch(\Exception $e)
    {
        return back()->withError($e->getMessage());
    }
    return view('indexAsigxestu',['estudiantes'=>$estudiantes]);

}
```

Figura 277. Método Index Asignación de Estudiantes por Asignatura

Create:

Método que permite abrir y redirigir a la Vista Ingresar Asignación de Estudiantes por Asignatura. Además de enviar un arreglo de Periodos (filtrando solo el Período activo), Estudiantes, Asignaturas, Carreras, Escuelas, Facultades y Profesores para cargar

información en <input> y <label> tags. De existir algún error en conexión, se envía un mensaje de error con la excepción encontrada.

```
public function create()
{
    try{
    $periodos = DB::table('periodos')->select('*')->where('estperiodo','A')->get();
    $estudiantes = DB::table('estudiantes')->select('*')->get();
    $asignaturas = DB::table('asignaturas')->select('*')->get();
    $carreras = DB::table('carreras')->select('*')->get();
    $escuelas = DB::table('escuelas')->select('*')->get();
    $facultades = DB::table('facultades')->select('*')->get();
    $profesores = DB::table('profesores')->select('*')->get();
    }catch(\Exception $e)
    {
        return back()->withError($e->getMessage());
    }
    return view('AsignaturasXEstudiante',compact('periodos','estudiantes',
    'asignaturas','carreras','profesores', 'escuelas', 'facultades'));

}
```

Figura 278. Método Create Asignación de Estudiantes por Asignatura

Store:

Método que permite almacenar un nuevo registro de Asignación de Estudiantes por Asignatura dentro de la base de datos del sistema mediante el uso de un formulario que se encuentra dentro de la vista Crear Asignación de Estudiantes por Asignatura.

El primer paso para ingresar registros de Asignación de Estudiantes por Asignatura es recibir toda la información del formulario y almacenarlas en variables que posteriormente permitirán asignar los 3 códigos de cada AsignaturaxEstudiante; los códigos o claves primarias corresponden al código de Asignatura, cédula del Estudiante y cédula del Profesor. Sin embargo, debido a que un Estudiante puede tener múltiples Asignaturas asignadas con uno o más Profesores, se utiliza un bucle ‘foreach’ para realizar múltiples inserciones dentro de la tabla ‘AsignaturasxEstudiantes’ de la base de datos del

sistema. Al finalizar la inserción de los registros con una sentencia SQL se manda un mensaje que dice ‘se añadió correctamente el/los registros’.

De existir algún error en conexión o si se intenta ingresar un registro duplicado, se envía un mensaje de error con la excepción encontrada.

```
    public function store(Request $request)
    {
        try{
        $periodos = $request->input('periodos');
        $cedestudiante = $request->input('cedestudiante');

        $asignaturas = $request->input('asignaturas');
        $profesores = $request->input('profesores');
        $counter = 0;
        foreach( $asignaturas as $asignatura){
            $asigs= DB::table('asignaturas')->select('codasignatura')->where('descasignatura','=',$asignatura)->get();
            $codasignatura;
            foreach ($asigs as $asig){
                $codasignatura = $asig->codasignatura;
            }

            DB::table('asignaturasxestudiantes')->insert([
                'codasignatura'=>$codasignatura,
                'cedestudiante'=>$cedestudiante,
                'codperiodo'=>$periodos[$counter],
                'cedprofesor'=>$profesores[$counter]
                ]);
            $counter++;
        }
    }catch(\Exception $e)
    {
        return back()->withError($e->getMessage());
    }
        return redirect('pagAsigxEstu')->with('success', 'Se añadio correctamente el/los registros');

}
```

Figura 279. Método Store Asignación de Estudiantes por Asignatura

Destroy:

Método que permite eliminar un registro existente de la tabla Asignación de Estudiantes por Asignatura en la base de datos del sistema. Para poder eliminar se recibe la cédula del Estudiante, y si se logra eliminar el registro exitosamente, se manda un mensaje a consola que diga registro borrado exitosamente. Caso contrario y si no se puede

eliminar debido a que se irrumpe con la integridad referencial de las tablas y sus registros, se manda un mensaje con la excepción encontrada.

```
public function destroy($cedestudiante)
{
    try{
    DB::table('asignaturasxestudiantes')->where('cedestudiante','=', $cedestudiante)->delete();
    }catch(\Exception $e)
    {
        return back()->withError($e->getMessage());
    }
    return redirect('pagAsigxEstu')->with('success','Se elimino correctamente');

}
```

Figura 280. Método Destroy Asignación de Estudiantes por Asignatura

Edit:

Método que permite abrir y redirigir a la Vista Editar Asignación de Estudiantes por Asignatura, para esto se recibe la cédula del Estudiante, que en base a una consulta SQL, permita recuperar el objeto Tema de Estudio y Asignaturas que posteriormente se va a editar en la base de datos y facilite enviar un arreglo de Períodos, Profesores, Asignaturas y Asignaturas por Estudiantes y un objeto de Estudiantes, Carreras, Escuelas y Facultades con los parámetros relacionados al Estudiante para cargar los combo box acumulativos en la Vista Editar Contenidos y así poder gestionar las relaciones del ingreso en cascada. De existir algún error en conexión o si no se encuentra un registro usando el código de Contenidos recibido, se envía un mensaje de error con la excepción encontrada.

```
public function edit($cedestudiante)//abre la ventana para modificar
{
    try{
    $periodos = DB::table('periodos')->select('*')->get();
    $estudiantes = DB::table('estudiantes')->select('*')->where('cedestudiante',$cedestudiante)->get();
    $estudiantes = json_decode($estudiantes,true)[0];

    $carreras = DB::table('carreras')->select('*')->where('codcarrera', $estudiantes['codcarrera'])->get();
    $carreras = json_decode($carreras,true)[0];

    $escuelas = DB::table('escuelas')->select('*')->where('codescuela', $carreras['codescuela'])->get();
    $escuelas = json_decode($escuelas,true)[0];

    $facultades = DB::table('facultades')->select('*')->where('codfacultad', $escuelas['codfacultad'])->get();
    $facultades = json_decode($facultades,true)[0];

    $asignaturas = DB::table('asignaturas')->select('*')->where('codcarrera', $estudiantes['codcarrera'])->get();

    $asignaturasxestudiantes = DB::table('asignaturasxestudiantes')->select('*')
    ->join('asignaturas', 'asignaturasxestudiantes.codasignatura', '=', 'asignaturas.codasignatura')
    ->join('profesores', 'asignaturasxestudiantes.cedprofesor', '=', 'profesores.cedprofesor')
    ->where('cedestudiante',$cedestudiante)
    ->get();

    $profesores = DB::table('profesores')->select('*')->where('codcarrera', $estudiantes['codcarrera'])->get();
    }catch(\Exception $e)
    {
        return back()->withError($e->getMessage());
    }
    return view('EditAsigxestu',compact('periodos','estudiantes','asignaturasxestudiantes',
    'asignaturas','carreras','profesores', 'escuelas', 'facultades'));

}
```

Figura 281. Método Edit Asignación de Estudiantes por Asignatura

Update:

Método que permite actualizar un registro existente de la tabla Asignación de Estudiantes por Asignatura en la base de datos del sistema, para esto se utiliza un objeto de tipo Request recibido a través de la vista Editar Asignación de Estudiantes por Asignatura.

El primer paso para actualizar los Contenidos es recibir toda la información del formulario y almacenarlas en variables que posteriormente permitirán asignar los 3 códigos de cada AsignaturaxEstudiante; los códigos o claves primarias corresponden al

código de Asignatura, cédula del Estudiante y cédula del Profesor. Sin embargo, debido a que un Estudiante puede tener múltiples Asignaturas asignadas con uno o más Profesores, se utiliza un bucle 'foreach' para realizar múltiples actualizaciones dentro de la tabla 'AsignaturasxEstudiantes' de la base de datos del sistema en base a los cambios recibidos por cada registro en base al formulario. Al finalizar la inserción de los registros con una sentencia SQL se manda un mensaje que dice 'se actualizó correctamente el/los registros'.

De existir algún error en conexión o si no se encuentra un registro usando el código de Asignaturas o de Carreras recibido por el objeto de tipo Request, se envía un mensaje de error con la excepción encontrada.

```
public function update(Request $request)//modificar
{
    try{
    $periodos = $request->input('periodos');
    $cedestudiante = $request->input('cedestudiante');

    $asignaturas = $request->input('asignaturas');


    DB::table('asignaturasxestudiantes')->where('cedestudiante','=', $cedestudiante)->delete();
    $profesores = $request->input('profesores');
    $counter = 0;
    foreach( $asignaturas as $asignatura){
        $asigs= DB::table('asignaturas')->select('codasignatura')->where('descasignatura','=',$asignatura)->get();
        $codasignatura;
        foreach ($asigs as $asig){
            $codasignatura = $asig->codasignatura;
        }

        DB::table('asignaturasxestudiantes')->insert([
            'codasignatura'=>$codasignatura,
            'cedestudiante'=>$cedestudiante,
            'codperiodo'=>$periodos[$counter],
            'cedprofesor'=>$profesores[$counter]
            ]);
        $counter++;
    }
}catch(\Exception $e)
{
    return back()->withError($e->getMessage());
}
    return redirect('pagAsigxEstu')->with('success', 'Se actualizó correctamente el/los registros');
```

Figura 282. Método Update Asignación de Estudiantes por Asignatura

Modelo de Asignación de Estudiantes por Asignatura:

El Modelo de Asignación de Estudiantes por Asignatura permite enlazar la capa de manejo de datos (MD) para recibir y enviar información con respecto a la tabla de la base de datos llamada AsignaturasxEstudiantes que se encuentra en PostgreSQL; la base de datos total del sistema se encuentra albergada en un gestor para el manejo de base de datos, que en este caso es pgAdmin 4.

El Modelo de Asignación de Estudiantes por Asignatura se lo utiliza dentro del Controlador de Asignación de Estudiantes por Asignatura para llamar al ORM por defecto de Laravel llamado Eloquent, que interpreta procesos de bases de datos escritos con instrucciones php; los cuales posteriormente son traducidos a sentencias SQL que realizan operaciones dentro de la tabla AsignaturasxEstudiantes en la base de datos.

```
class AsignaturasxestudianteModel extends Model
{
    protected $table = 'asignaturasxestudiantes';
    public $timestamps = false;
}
```

Figura 283. Asignación de Estudiantes por Asignatura Model

Rutas de Asignación de Estudiantes por Asignatura:

En el archivo de Rutas de Asignación de Estudiantes por Asignatura se encuentran las direcciones que permiten conectar la capa lógica (DP) con las vistas (GUI) de las Asignaturas por Estudiantes dentro del sistema.

```
//ASIGNATURAS POR ESTUDIANTE
Route::resource('pagAsigxEstu','AsignaturaXEstudianteController');
Route::post('actualizarAsigxestu', 'AsignaturaXEstudianteController@update');
```

Figura 284. Asignación de Estudiantes por Asignatura Formularios Route Path

Vista Principal de Asignación de Estudiantes por Asignatura:

La Vista Principal de Asignación de Estudiantes por Asignatura permite mostrar todos los datos que se encuentran almacenados en la tabla AsignaturasxEstudiantes de la Base de Datos del sistema. Los datos del registro se reflejan en una tabla tipo CRUD que permite seleccionar el tipo de operación que se quiere realizar con respecto a los datos (insertar, borrar y actualizar). Las operaciones se ven reflejadas en forma de botones.

```
<table id="myTable" class="table table-striped">
    <thead>
        <tr>

            <th>Cedula de Estudiante</th>
            <th>Nombre de Estudiante</th>
            <th>Apellido de Estudiante</th>
            <th>Carrera</th>
            <th>Correo de Estudiante</th>
            <th>Nuevo</th>
            <th>Modificar</th>
            <th>Eliminar</th>
        </tr>
    </thead>
```

Figura 285. Asignación de Estudiantes por Asignatura Blade 1

Como se puede observar, es necesario crear una tabla que sirva como almacén para los registros recuperados de la tabla AsignaturasxEstudiantes que se encuentra en la base de datos del sistema.

```
<td>{{$estudiante['cedestudiante']}}</td>
<td>{{$estudiante['nomestudiante']}}</td>
<td>{{$estudiante['apeestudiante']}}</td>
<td>{{$estudiante['codcarrera']}}</td>
<td>{{$estudiante['correestudiante']}}</td>
<td><a href="{{action('AsignaturaXEstudianteController@edit', $estudiante['cedestudiante'])}}"
      class="btn btn-primary">Nuevo</a></td>
<td><a href="{{action('AsignaturaXEstudianteController@edit', $estudiante['cedestudiante'])}}"
      class="btn btn-warning">Modificar</a></td>
```

Figura 286. Asignación de Estudiantes por Asignatura Blade 2

Como se puede observar, es importante definir las columnas <td> en base al nombre declarado en la función Index del controlador. Ya que caso contrario no se va a poder recuperar los datos y cargarlos en las celdas de la tabla. Sin embargo, se debe usar un tag <a> con una propiedad href para poder recuperar la información en un solo sentido; eso sí, especificando el url exacto de la ruta definida en el archivo web.php.

```
<script>
$(document).ready(function() {
    $('#myTable').DataTable({
      "pageLength": 10
    });
} );

</script>
```

Figura 287. Asignación de Estudiantes por Asignatura Blade 3

Finalmente, y como se puede observar, la librería Data Tables se utiliza para poder cargar de forma dinámica y con posibilidad de obtener un campo de búsqueda por columna y coincidencia los datos de una consulta SQL en la base de datos. Aquí se considera el tipo de notación JSON para indexar campos y así realizar consultas que reordenen la tabla

por id; en este caso se tiene 3 parámetros de reordenamiento, la cédula del Estudiante y Profesor, y el código de Asignatura.

Vista Ingresar Asignación de Estudiantes por Asignatura:

La Vista Ingresar Asignación de Estudiantes por Asignatura permite ingresar nuevos registros en la tabla AsignaturasxEstudiantes de la Base de Datos del sistema en base a los datos de carga de los <input> tag cedEstudiante, descFacultad, desEscuela, descCarrera y nombreApellido.

```
<form method="post" action="{{url('pagAsigxEstu')}}" enctype="multipart/form-data">
    @csrf
    <div class="form-horizontal">
        <div class="form-group row">
            <label class="control-label col-sm-3" for="cedestudiante">Cedula del estudiante</label>
            <div class="col-sm-6">
                <input type="text" id="cedestudiante" class="form-control" name="cedestudiante">
            </div>
        </div>

        <div class="form-group row">
            <label class="control-label col-sm-3" for="descfacultad">Facultad</label>
            <div class="col-sm-6">
                <input type="text" id="descfacultad" class="form-control" name="descfacultad"
                    readonly>
            </div>
        </div>

        <div class="form-group row">
            <label class="control-label col-sm-3" for="descescuela">Escuela</label>
            <div class="col-sm-6">
                <input type="text" id="descescuela" class="form-control" name="descescuela"
                    readonly>
            </div>
        </div>

        <div class="form-group row">
            <label class="control-label col-sm-3" for="desccarrera">Carrera</label>
            <div class="col-sm-6">
                <input type="text" id="desccarrera" class="form-control" name="desccarrera"
                    readonly>
            </div>
        </div>


        <div class="form-group row">
            <label class="control-label col-sm-3" for="nombreapellido">Nombre</label>
            <div class="col-sm-6">
                <input type="text" id="nombreapellido" class="form-control" name="nombreapellido"
                    readonly>
            </div>
        </div>
```

Figura 288. Asignación de Estudiantes por Asignatura Create Blade 1

Como se pudo observar, es necesario colocar la instrucción {{ csrf_field()}} que permite denegar la falsificación de petición en sitios cruzados (se origina cuando un sitio realiza peticiones en nombre de un usuario sobre nuestro sitio). Al denegar la petición

permite acceder a las rutas declaradas en el web.php con post y get para ingresar, recuperar y alterar información dentro de un Blade al usar los tags <input> , <select>, etc. Además, es importante definir el nombre de cada tag <input> y <select> en base al nombre declarado en la función Index del controlador. Ya que caso contrario no se va a poder recuperar los datos y posteriormente ingresar el registro.

```
    <table class="table table-striped" id="myTable">
        <thead>
            <tr>
                <th>Asignatura</th>
                <th>Profesores</th>
                <th>Periodo</th>
                <th>Accion</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td></td>
                <td></td>
                <td></td>
                <td></td>
            </tr>
        </tbody>
    </table>
</div>

<div class="row">
    <div class="col-md-5"></div>
    <div class="form-group col-md-4">
        <button type="submit" class="btn btn-primary btn-save">Guardar</button>
    </div>
</div>
```

Figura 289. Asignación de Estudiantes por Asignatura Create Blade 2

Es necesario colocar un esqueleto de la tabla de detalle en la estructura de cabecera a detalle para cargar los registros que se van a colocar de Asignaturas por Estudiante.

Vista Modificar Asignación de Estudiantes por Asignatura:

La Vista Modificar Asignación de Estudiantes por Asignatura permite modificar un registro previamente ingresado dentro de la tabla AsignaturasxEstudiantes de la Base de Datos del sistema.

```
<form    method="post" action="{{URL::to('/actualizarAsigxestu')}}" enctype="multipart/form-data">
@csrf

<div class="form-horizontal">
                        <div class="form-group row">
                            <label class="control-label col-sm-3" for="cedestudiante">Cedula del estudiante</label>
                            <div class="col-sm-6">
                                <input type="text" id="cedestudiante" class="form-control" name="cedestudiante"
                                value="{{$estudiantes['cedestudiante']}}" readonly>
                            </div>
                        </div>

                        <div class="form-group row">
                            <label class="control-label col-sm-3" for="descfacultad">Facultad</label>
                            <div class="col-sm-6">
                                <input type="text" id="descfacultad" class="form-control" name="descfacultad"
                                value="{{$facultades['descfacultad']}}" readonly>
                            </div>
                        </div>

                        <div class="form-group row">
                            <label class="control-label col-sm-3" for="descescuela">Escuela</label>
                            <div class="col-sm-6">
                                <input type="text" id="descescuela" class="form-control" name="descescuela"
                                value="{{$escuelas['descescuela']}}"readonly>
                            </div>
                        </div>

                        <div class="form-group row">
                            <label class="control-label col-sm-3" for="desccarrera">Carrera</label>
                            <div class="col-sm-6">
                                <input type="text" id="desccarrera" class="form-control" name="desccarrera"
                                value="{{$carreras['desccarrera']}}" readonly>
                            </div>
                        </div>


                        <div class="form-group row">
                            <label class="control-label col-sm-3" for="nombreapellido">Nombre</label>
                            <div class="col-sm-6">
                                <input type="text" id="nombreapellido" class="form-control" name="nombreapellido"
                                value="{{$estudiantes['nomestudiante']}} {{$estudiantes['apeestudiante']}}" readonly>
                            </div>
                        </div>
```

Figura 290. Asignación de Estudiantes por Asignatura Edit Blade 1

Como se pudo observar, es necesario colocar la instrucción {{ csrf_field()}} que permite denegar la falsificación de petición en sitios cruzados (se origina cuando un sitio realiza peticiones en nombre de un usuario sobre nuestro sitio). Al denegar la petición permite acceder a las rutas declaradas en el web.php con post y get para ingresar, recuperar y alterar información dentro de un Blade al usar los tags <input> , <select>, etc.

```
<table id="myTable" class="table table-striped">
<thead class="thead-light">
  <tr>
        <th>Asignatura</th>
        <th>Profesores</th>
        <th>Periodo</th>
        <th>Accion</th>
  </tr>
</thead>
<tbody>
  @foreach($asignaturasxestudiantes as $ass)
  @foreach($periodos as $periodo)
  @if($ass->codperiodo == $periodo->codperiodo && $periodo->estperiodo == "A")
  <tr>
  <td><div class="form-row"><div class="form-group col-md-12">
  <select class="selectpicker form-control " name="asignaturas[]" data-live-search="true" required novalidate>
  <option value="{{$ass->descasignatura}}">{{$ass->descasignatura}}</option>
  @foreach ($asignaturas as $asig)
  <option value="{{$asig->descasignatura}}">{{$asig->descasignatura}}</option>
  @endforeach
  </select></div></td>
  <td><div class="form-row"><div class="form-group col-md-12">
  <select class="selectpicker form-control " name="profesores[]" data-live-search="true" required novalidate>'+
  <option value='{{$ass->cedprofesor}}'>{{$ass->nomprofesor}} {{$ass->apeprofesor}}</option>
  @foreach ($profesores as $prof)
  <option value="{{$prof->cedprofesor}}">{{$prof->nomprofesor}} {{$prof->apeprofesor}} </option>@endforeach
  </select></div></td>
  <td><div class="form-row"><div class="form-group col-md-12">
  <select class="selectpicker form-control " name="periodos[]" data-live-search="true" required novalidate>
  <option selected value='{{$ass->codperiodo}}'>{{$ass->codperiodo}}</option>
  </select></div></td>
  <td><div class="form-group col-md-4"><button type="button" class="btn ibtnDel btn-danger"><span class="glyphicon glyphicon-remove-sign">Remover</span></button></div></div></div></td>
  @elseif($ass->codperiodo == $periodo->codperiodo && $periodo->estperiodo == "I")
  <tr>
  <td><div class="form-row"><div class="form-group col-md-12">
  <select class="selectpicker form-control " name="asignaturas[]" data-live-search="true" required novalidate>
  <option selected value="{{$ass->descasignatura}}">{{$ass->descasignatura}}</option>
  </select></div></td>
```

Figura 291. Asignación de Estudiantes por Asignatura Edit Blade 2

Finalmente, y en base a la Figura 150, es importante definir el nombre de cada tag <input> y <select> en base al nombre declarado en la función Index del controlador. Ya que caso contrario no se va a poder recuperar los datos para cargar los valores del registro previamente ingresado en la tabla CRUD y posteriormente actualizar el registro. No se debe olvidar que las claves primarias se mantienen fijas hasta acabar la edición de los campos seleccionables o editables. es necesario colocar los datos del detalle en una tabla que conserve la estructura de cabecera a detalle para poder editar con opciones existentes de Asignaturas y Profesores asignados.

Vista Eliminar Asignación de Estudiantes por Asignatura:

La Vista Eliminar Asignación de Estudiantes por Asignatura permite eliminar un registro previamente ingresado dentro de la tabla AsignaturasxEstudiantes de la Base de

Datos del sistema. Esta opción se encuentra disponible directamente desde la tabla CRUD en la Vista Principal de Asignación de Estudiantes por Asignatura.

```
<form
    action="{{action('AsignaturaXEstudianteController@destroy', $estudiante['cedestudiante'])}}"
    method="post">
    @csrf
    <input name="_method" type="hidden" value="DELETE">
    <button class="btn btn-danger" type="submit">Eliminar</button>
</form>
```

Figura 292. Asignación de Estudiantes por Asignatura Blade – Eliminar Registro

Es importante recordar que para borrar un registro se necesita confirmar el proceso de borrado. Para esto se utiliza una propiedad del tag <form> llamada onsubmit, que permite confirmar el proceso de borrado.

Interfaces Asignación de Estudiantes por Asignatura:

Las Interfaces de Asignación de Estudiantes por Asignatura se encuentran separadas por cada funcionalidad descrita en las Vistas de Asignación de Estudiantes por Asignatura.

Interface Principal de Asignación de Estudiantes por Asignatura (Tabla CRUD):

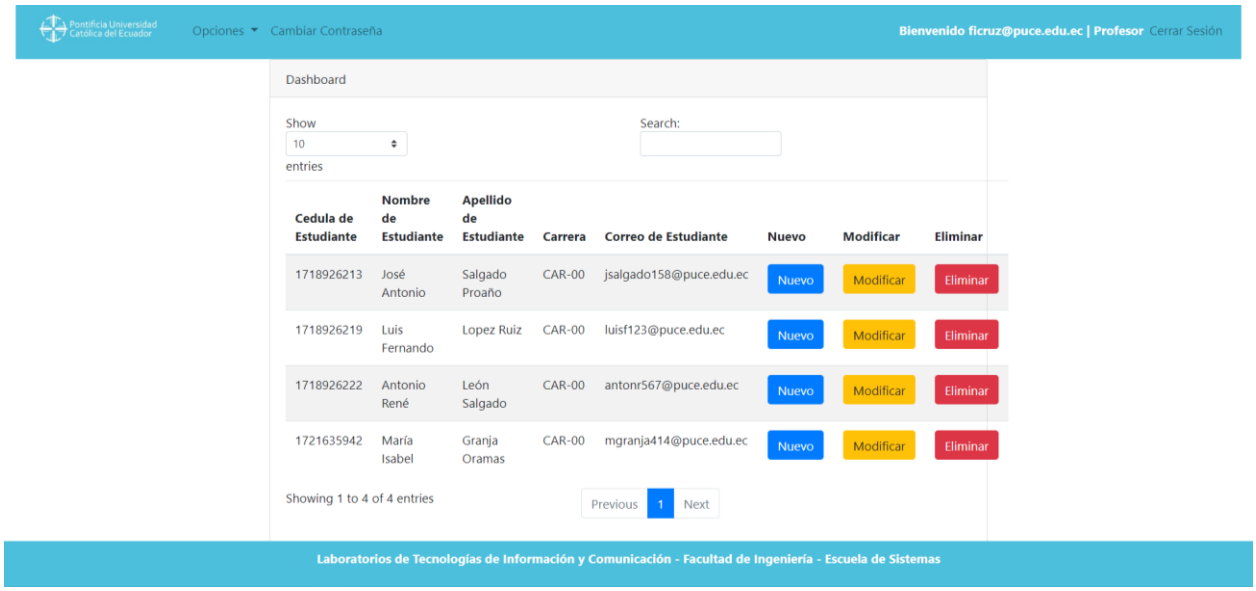


Figura 293. Página de Asignación de Estudiantes por Asignatura Principal

La Interface Principal de Asignación de Estudiantes por Asignatura tiene un diseño de tabla Responsivo que permite manejar las 4 operaciones básicas que componen a un CRUD (crear, borrar, actualizar y consultar).

Interface Ingreso Asignación de Estudiantes por Asignatura por Formulario:

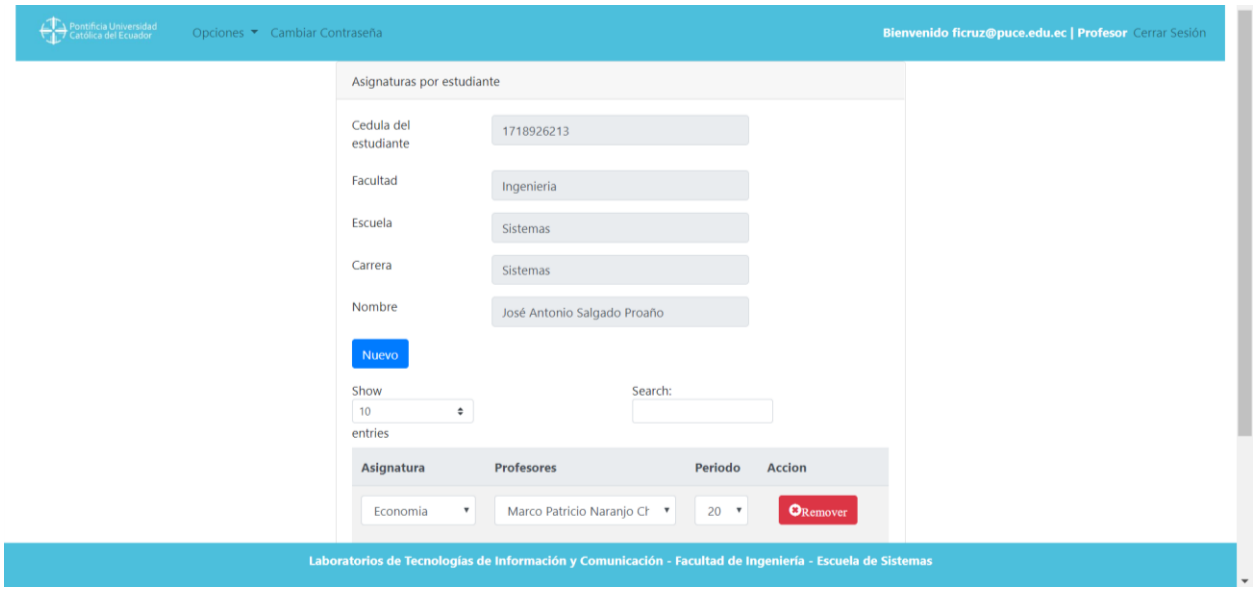


Figura 294. Página de Asignación de Estudiantes por Asignatura Crear

Como se puede observar, en la Interface de Ingreso de Asignación de Estudiantes por Asignatura se colocan los datos para agregar un nuevo registro en la tabla de AsignaturasxEstudiantes en base a la cabecera de datos del Estudiante. Al presionar guardar se agrega un nuevo registro y se coloca en la tabla CRUD.

Interface Edición Asignación de Estudiantes por Asignatura:

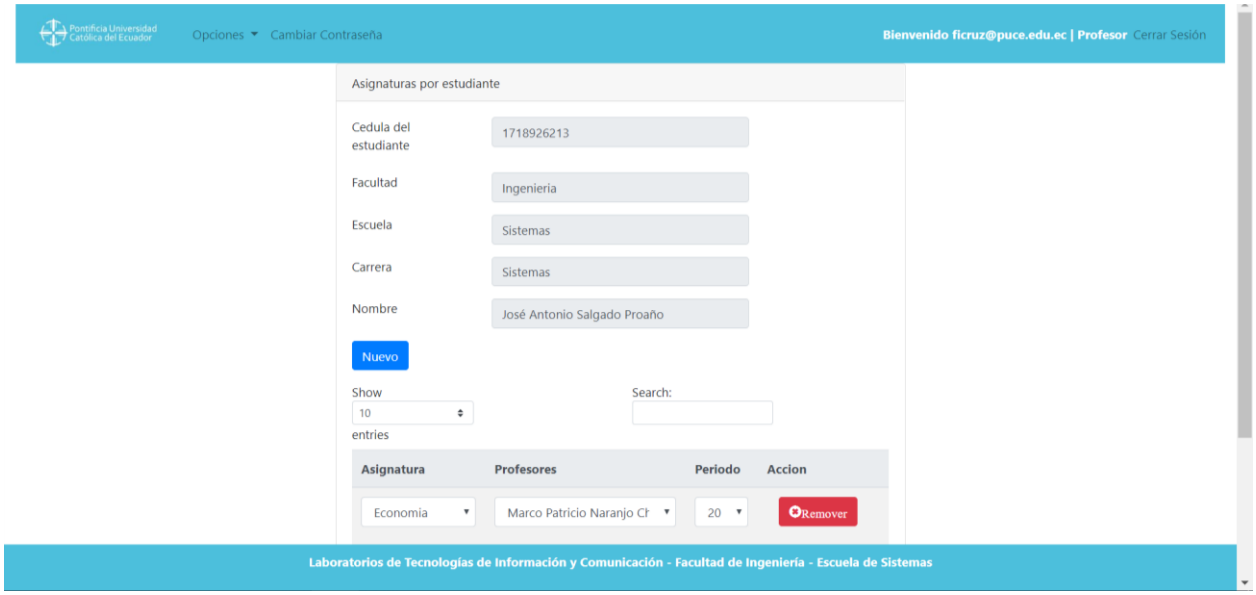


Figura 295. Página de Asignación de Estudiantes por Asignatura Editar

Como se puede observar, en la Interface de Edición de Asignación de Estudiantes por Asignatura se colocan los datos recuperados de un registro previamente ingresados, para editar y actualizar un registro ya existente de la tabla AsignaturasxEstudiantes. Al presionar guardar se modifica el registro ya en base a la cédula del Estudiante con formato no editable y se coloca en la tabla CRUD.

Requisitos, Relaciones, Procesos de Grabado, Validaciones y Estructuras que se Deben Tomar en Cuenta para la Construcción de Interfaces

Asignación de Estudiantes por Asignatura General

Sistema: Agenda Electrónica

Tabla: ASIGNATURASXESTUDIANTES

Relaciones: est_toma_asigxest, asig_tiene_asigxestu, prof_dicta_asigxest.

Requisitos Previos: Validar existencia de Estudiantes, Profesores y Asignaturas

Proceso de grabado: Tabla ASIGNATURASXESTUDIANTES.

Validaciones:

Se selecciona la descripción de Universidad en el combo box Universidad, ya que en base a la Universidad en la que se encuentre se realiza la búsqueda por cédula.

Se realiza la consulta de la Facultad a la que pertenece el Profesor y el Estudiante y se muestra su nombre en un cuadro de texto no es editable.

Se realiza la consulta de la Carrera a la que pertenece el Profesor y el Estudiante y se muestra su nombre en un cuadro de texto no es editable.

Se ingresa el número de cedula del Estudiante.

Se realiza la consulta del nombre del Estudiante en base a la cedula ingresada anteriormente y se muestra su nombre en un cuadro de texto no es editable.

Se realiza una consulta general de todas las Asignaturas dictadas por un respectivo Profesor y tomadas por un respectivo Estudiante. El resultado de la consulta se muestra en cada cuadro de texto de manera ascendente. Solo se muestran 10 registros resultantes de la consulta.

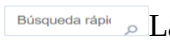
Cada fila de registros es editable.

Botones:

El botón de nuevo despliega por pantalla cuadros de texto para añadir información. Para confirmar y guardar en la base de datos se presiona el botón de visto ✓, en caso contrario, para cancelar y no almacenar los datos se presiona la equis ✕.

El lápiz permite modificar un registro ingresado previamente.

El basurero permite eliminar un registro ingresado previamente.



La barra de búsqueda permite realizar una consulta en base a cualquier atributo de la tabla.

El botón salir permite regresar a la pantalla principal.

Pruebas

Pruebas Funcionales

A continuación, se elabora un Plan de Pruebas Funcionales por cada uno de los módulos del sistema, esto permite validar que se cumplan los requerimientos Funcionales establecidos se puede y validar el correcto funcionamiento de la aplicación en base a lo definido en los diagramas de Casos de Uso. También se comprueba que el resultado de cada caso de prueba sea igual al resultado de salida esperado.

F1: Gestionar Provincia

Información General			
Nombre de caso de uso:	Gestionar Provincia		
Prerrequisitos			
Ninguno			
Descripción de Casos de Prueba			
Caso: Se realiza el proceso para la creación, modificación, eliminación y consulta de una provincia			
Escenarios de prueba		**Respuesta esperada de la aplicación**	**Coincide (Si/No)**
#	**Nombre de la prueba a ejecutar**		
1	Ingreso de Provincia	Ingreso satisfactorio	SI
2	Modificación de Provincia	Modificación satisfactoria	SI
3	Eliminación de Provincia	Eliminación satisfactoria	SI
4	Consulta General de Provincia	Consulta satisfactoria	SI

5	Consulta específica de Provincia	Consulta satisfactoria	SI
PORCENTAJE			
Back - End	100%		
Front - End	100%		

F2: Gestionar Ciudad

Información General			
Nombre de caso de uso:	Gestionar Ciudad		
Prerrequisitos			
Deben existir registros previamente ingresados de Provincias			
Descripción de Casos de Prueba			
Caso: Se realiza el proceso para la creación, modificación, eliminación y consulta de una Ciudad			
Escenarios de prueba		**Respuesta esperada de la aplicación**	**Coincide (Si/No)**
#	**Nombre de la prueba a ejecutar**		
1	Ingreso de Ciudad	Ingreso satisfactorio	SI
2	Modificación de Ciudad	Modificación satisfactoria	SI
3	Eliminación de Ciudad	Eliminación satisfactoria	SI
4	Consulta General de Ciudad	Consulta satisfactoria	SI
5	Consulta específica de Ciudad	Consulta satisfactoria	SI
PORCENTAJE			
Back - End	100%		
Front - End	100%		

F3: Gestionar Universidad

Información General	
Nombre de caso de uso:	Gestionar Universidad
Prerrequisitos	

Ninguno.			
Descripción de Casos de Prueba			
Caso: Se realiza el proceso para la creación, modificación, eliminación y consulta de una Universidad			
Escenarios de prueba		**Respuesta esperada de la aplicación**	**Coincide (Si/No)**
#	**Nombre de la prueba a ejecutar**		
1	Ingreso de Universidad	Ingreso satisfactorio	SI
2	Modificación de Universidad	Modificación satisfactoria	SI
3	Eliminación de Universidad	Eliminación satisfactoria	SI
4	Consulta General de Universidad	Consulta satisfactoria	SI
5	Consulta específica de Universidad	Consulta satisfactoria	SI
PORCENTAJE			
Back - End	100%		
Front - End	100%		

F4: Gestionar Sede

Información General			
Nombre de caso de uso:	Gestionar Sede		
Prerrequisitos			
Deben existir registros previamente ingresados de Universidades y de Ciudades			
Descripción de Casos de Prueba			
Caso: Se realiza el proceso para la creación, modificación, eliminación y consulta de una Sede			
Escenarios de prueba		**Respuesta esperada de la aplicación**	**Coincide (Si/No)**
#	**Nombre de la prueba a ejecutar**		
1	Ingreso de Sede	Ingreso satisfactorio	SI
2	Modificación de Sede	Modificación satisfactoria	SI
3	Eliminación de Sede	Eliminación satisfactoria	SI
4	Consulta General de Sede	Consulta satisfactoria	SI
5	Consulta específica de Sede	Consulta satisfactoria	SI

PORCENTAJE	
Back - End	100%
Front - End	100%

F5: Gestionar Periodo

Información General			
Nombre de caso de uso:	Gestionar Periodo		
Prerrequisitos			
Deben existir registros previamente ingresados de Sedes			
Descripción de Casos de Prueba			
Caso: Se realiza el proceso para la creación, modificación, eliminación y consulta de un Periodo			
Escenarios de prueba		**Respuesta esperada de la aplicación**	**Coincide (Si/No)**
#	**Nombre de la prueba a ejecutar**		
1	Ingreso de Periodo	Ingreso satisfactorio	SI
2	Modificación de Periodo	Modificación satisfactoria	SI
3	Eliminación de Periodo	Eliminación satisfactoria	SI
4	Consulta General de Periodo	Consulta satisfactoria	SI
5	Consulta específica de Sede de Periodo	Consulta satisfactoria	SI
PORCENTAJE			
Back - End	100%		
Front - End	100%		

F6: Gestionar Facultad

Información General	
Nombre de caso de uso:	Gestionar Facultad
Prerrequisitos	
Ninguno.	

Descripción de Casos de Prueba			
Caso: Se realiza el proceso para la creación, modificación, eliminación y consulta de una Facultad.			
Escenarios de prueba		**Respuesta esperada de la aplicación**	**Coincide (Si/No)**
#	**Nombre de la prueba a ejecutar**		
1	Ingreso de Facultad	Ingreso satisfactorio	SI
2	Modificación de Facultad	Modificación satisfactoria	SI
3	Eliminación de Facultad	Eliminación satisfactoria	SI
4	Consulta General de Facultad	Consulta satisfactoria	SI
5	Consulta específica de Facultad	Consulta satisfactoria	SI
PORCENTAJE			
Back - End	100%		
Front - End	100%		

F7: Gestionar Escuela

Información General	
Nombre de caso de uso:	Gestionar Escuela
Prerrequisitos	
Deben existir registros previamente ingresados de Facultades por Sede	
Descripción de Casos de Prueba	
Caso: Se realiza el proceso para la creación, modificación, eliminación y consulta de una Escuela	

Escenarios de prueba		**Respuesta esperada de la aplicación**	**Coincide (Si/No)**
#	**Nombre de la prueba a ejecutar**		
1	Ingreso de Escuela	Ingreso satisfactorio	SI
2	Modificación de Escuela	Modificación satisfactoria	SI
3	Eliminación de Escuela	Eliminación satisfactoria	SI
4	Consulta General de Escuela	Consulta satisfactoria	SI
5	Consulta específica de Escuela	Consulta satisfactoria	SI
PORCENTAJE			
Back - End	100%		

Front - End	100%

F8: Gestionar Carrera

Información General			
Nombre de caso de uso:	Gestionar Carrera		
Prerrequisitos			
Deben existir registros previamente ingresados de Escuelas			
Descripción de Casos de Prueba			
Caso: Se realiza el proceso para la creación, modificación, eliminación y consulta de una Carrera			
Escenarios de prueba		**Respuesta esperada de la aplicación**	**Coincide (Si/No)**
#	**Nombre de la prueba a ejecutar**		
1	Ingreso de Carrera	Ingreso satisfactorio	SI
2	Modificación de Carrera	Modificación satisfactoria	SI
3	Eliminación de Carrera	Eliminación satisfactoria	SI
4	Consulta General de Carrera	Consulta satisfactoria	SI
5	Consulta específica de Carrera	Consulta satisfactoria	SI
PORCENTAJE			
Back - End	100%		
Front - End	100%		

F9: Gestionar Estudiante

Información General			
Nombre de caso de uso:	Gestionar Estudiante		
Prerrequisitos			
Deben existir registros previamente ingresados de Carreras			
Descripción de Casos de Prueba			
Caso: Se realiza el proceso para la creación, modificación, eliminación y consulta de un Estudiante			
Escenarios de prueba		**Respuesta esperada de la aplicación**	**Coincide (Si/No)**
#	**Nombre de la prueba a ejecutar**		
1	Ingreso de Estudiante	Ingreso satisfactorio	SI

2	Modificación de Estudiante	Modificación satisfactoria	SI
3	Eliminación de Estudiante	Eliminación satisfactoria	SI
4	Consulta General de Estudiante	Consulta satisfactoria	SI
5	Consulta específica de Estudiante	Consulta satisfactoria	SI
PORCENTAJE			
Back - End	100%		
Front - End	100%		

F10: Gestionar Profesor

Información General			
Nombre de caso de uso:	Gestionar Profesor		
Prerrequisitos			
Deben existir registros previamente ingresados de Carreras			
Descripción de Casos de Prueba			
Caso: Se realiza el proceso para la creación, modificación, eliminación y consulta de un Profesor			
Escenarios de prueba		**Respuesta esperada de la aplicación**	**Coincide (Si/No)**
#	**Nombre de la prueba a ejecutar**		
1	Ingreso de Profesor	Ingreso satisfactorio	SI
2	Modificación de Profesor	Modificación satisfactoria	SI
3	Eliminación de Profesor	Eliminación satisfactoria	SI
4	Consulta General de Profesor	Consulta satisfactoria	SI
5	Consulta específica de Profesor	Consulta satisfactoria	SI
PORCENTAJE			
Back - End	100%		
Front - End	100%		

F11: Gestionar Asignatura

Información General	
Nombre de caso de uso:	Gestionar Asignatura
Prerrequisitos	

Deben existir registros previamente ingresados de Carreras			
Descripción de Casos de Prueba			
Caso: Se realiza el proceso para la creación, modificación, eliminación y consulta de una Asignatura			
Escenarios de prueba		**Respuesta esperada de la aplicación**	**Coincide (Si/No)**
#	**Nombre de la prueba a ejecutar**		
1	Ingreso de Asignatura	Ingreso satisfactorio	SI
2	Modificación de Asignatura	Modificación satisfactoria	SI
3	Eliminación de Asignatura	Eliminación satisfactoria	SI
4	Consulta General de Asignatura	Consulta satisfactoria	SI
5	Consulta específica de Asignatura	Consulta satisfactoria	SI
PORCENTAJE			
Back - End	100%		
Front - End	100%		

F12: Gestionar Unidad de Estudio

Información General	
Nombre de caso de uso:	Gestionar Unidad de Estudio
Prerrequisitos	
Deben existir registros previamente ingresados de Asignaturas	
Descripción de Casos de Prueba	
Caso: Se realiza el proceso para la creación, modificación, eliminación y consulta de unidades	

Escenarios de prueba		**Respuesta esperada de la aplicación**	**Coincide (Si/No)**
#	**Nombre de la prueba a ejecutar**		
1	Ingreso de Unidad de Estudio	Ingreso satisfactorio	SI
2	Modificación de Unidad de Estudio	Modificación satisfactoria	SI
3	Eliminación de Unidad de Estudio	Eliminación satisfactoria	SI
4	Consulta General de Unidad de Estudio	Consulta satisfactoria	SI

5	Consulta específica de Unidad de Estudio	Consulta satisfactoria	SI
PORCENTAJE			
Back - End	100%		
Front - End	100%		

F13: Gestionar Tema de Estudio

Información General			
Nombre de caso de uso:	Gestionar Tema de Estudio		
Prerrequisitos			
Deben existir registros previamente ingresados de Asignaturas y Unidades de Estudio			
Descripción de Casos de Prueba			
Caso: Se realiza el proceso para la creación, modificación, eliminación y consulta de temas			
Escenarios de prueba		**Respuesta esperada de la aplicación**	**Coincide (Si/No)**
#	**Nombre de la prueba a ejecutar**		
1	Ingreso de Tema de Estudio	Ingreso satisfactorio	SI
2	Modificación de Tema de Estudio	Modificación satisfactoria	SI
3	Eliminación de Tema de Estudio	Eliminación satisfactoria	SI
4	Consulta General de Tema de Estudio	Consulta satisfactoria	SI
5	Consulta específica de Tema de Estudio	Consulta satisfactoria	SI
PORCENTAJE			
Back - End	100%		
Front - End	100%		

F14: Gestionar Contenido

Información General	
Nombre de caso de uso:	Gestionar Contenido
Prerrequisitos	
Deben existir registros previamente ingresados de Asignaturas y Temas de Estudio	

Descripción de Casos de Prueba			
Caso: Se realiza el proceso para la creación, modificación, eliminación y consulta de Contenidos			
Escenarios de prueba		**Respuesta esperada de la aplicación**	**Coincide (Si/No)**
#	**Nombre de la prueba a ejecutar**		
1	Ingreso de Contenido	Ingreso satisfactorio	SI
2	Modificación de Contenido	Modificación satisfactoria	SI
3	Eliminación de Contenido	Eliminación satisfactoria	SI
4	Consulta General de Contenido	Consulta satisfactoria	SI
5	Consulta específica de Contenido	Consulta satisfactoria	SI
PORCENTAJE			
Back - End	100%		
Front - End	100%		

F15: Gestionar Anotación

Información General	
Nombre de caso de uso:	Gestionar Anotación
Prerrequisitos	
Deben existir registros previamente ingresados de Contenidos	
Descripción de Casos de Prueba	
Caso: Se realiza el proceso para la creación, modificación, eliminación y consulta de Contenidos	

Escenarios de prueba		**Respuesta esperada de la aplicación**	**Coincide (Si/No)**
#	**Nombre de la prueba a ejecutar**		
1	Ingreso de Anotación	Ingreso satisfactorio	SI
2	Modificación de Anotación	Modificación satisfactoria	SI
3	Consulta específica de Anotación	Consulta satisfactoria	SI
PORCENTAJE			
Back - End	100%		
Front - End	100%		

F16: Gestionar Evaluación

Información General			
Nombre de caso de uso:	Gestionar Evaluación		
Prerrequisitos			
Deben existir registros previamente ingresados de Temas de Estudio			
Descripción de Casos de Prueba			
Caso: Se realiza el proceso para la creación, modificación, eliminación y consulta de Evaluaciones			
Escenarios de prueba		**Respuesta esperada de la aplicación**	**Coincide (Si/No)**
#	**Nombre de la prueba a ejecutar**		
1	Ingreso de Evaluación	Ingreso satisfactorio	SI
2	Modificación de Evaluación	Modificación satisfactoria	SI
3	Eliminación de Evaluación	Eliminación satisfactoria	SI
4	Consulta General de Evaluación	Consulta satisfactoria	SI
5	Consulta específica de Evaluaciones	Consulta satisfactoria	SI
PORCENTAJE			
Back - End	100%		
Front - End	100%		

F17: Gestionar Glosario

Información General			
Nombre de caso de uso:	Gestionar Glosario		
Prerrequisitos			
Deben existir registros previamente ingresados de Temas de Estudio			
Descripción de Casos de Prueba			
Caso: Se realiza el proceso para la creación, modificación, eliminación y consulta de Glosarios			
Escenarios de prueba		**Respuesta esperada de la aplicación**	**Coincide (Si/No)**
#	**Nombre de la prueba a ejecutar**		
1	Ingreso de Glosario	Ingreso satisfactorio	SI
2	Modificación de Glosario	Modificación satisfactoria	SI
3	Eliminación de Glosario	Eliminación satisfactoria	SI

4	Consulta General de Glosario	Consulta satisfactoria	SI
5	Consulta específica de Glosario	Consulta satisfactoria	SI
PORCENTAJE			
Back - End	100%		
Front - End	100%		

F18: Gestionar Taller

Información General			
Nombre de caso de uso:	Gestionar Taller		
Prerrequisitos			
Deben existir registros previamente ingresados de Temas de Estudio			
Descripción de Casos de Prueba			
Caso: Se realiza el proceso para la creación, modificación, eliminación y consulta de Talleres			
Escenarios de prueba		**Respuesta esperada de la aplicación**	**Coincide (Si/No)**
#	**Nombre de la prueba a ejecutar**		
1	Ingreso de Taller	Ingreso satisfactorio	SI
2	Modificación de Taller	Modificación satisfactoria	SI
3	Eliminación de Taller	Eliminación satisfactoria	SI
4	Consulta General de Taller	Consulta satisfactoria	SI
5	Consulta específica de Taller	Consulta satisfactoria	SI
PORCENTAJE			
Back - End	100%		
Front - End	100%		

F19: Gestionar Asignación de Asignaturas por Profesor

Información General	
Nombre de caso de uso:	Gestionar Asignación de Asignaturas por Profesor
Prerrequisitos	
Deben existir registros previamente ingresados de Asignaturas, Profesores y Periodos	
Descripción de Casos de Prueba	

Caso: Se realiza el proceso para la creación, modificación, eliminación y consulta de Asignaturas por Profesor			
Escenarios de prueba		**Respuesta esperada de la aplicación**	**Coincide (Si/No)**
#	**Nombre de la prueba a ejecutar**		
1	Ingreso de Asignatura por Profesor	Ingreso satisfactorio	SI
2	Modificación de Asignatura por Profesor	Modificación satisfactoria	SI
3	Eliminación de Asignatura por Profesor	Eliminación satisfactoria	SI
4	Consulta General de Asignatura por Profesor	Consulta satisfactoria	SI
5	Consulta específica de Asignatura por Profesor	Consulta satisfactoria	SI
PORCENTAJE			
Back - End	100%		
Front - End	100%		

F20: Gestionar Asignación de Asignaturas por Estudiante

Información General	
Nombre de caso de uso:	Gestionar Asignación de Asignaturas por Estudiante
Prerrequisitos	
Deben existir registros previamente ingresados en Asignaturas, Estudiantes y Periodos	
Descripción de Casos de Prueba	
Caso: Se realiza el proceso para la creación, modificación, eliminación y consulta de Asignaturas por Estudiante	

Escenarios de prueba		Respuesta esperada de la aplicación	Coincide (Si/No)
#	**Nombre de la prueba a ejecutar**		
1	Ingreso de Asignatura por Estudiante	Ingreso satisfactorio	SI
2	Modificación de Asignatura por Estudiante	Modificación satisfactoria	SI
3	Eliminación de Asignatura por Estudiante	Eliminación satisfactoria	SI
4	Consulta General de Asignatura por Estudiante	Consulta satisfactoria	SI

5	Consulta específica de Asignatura por Estudiante	Consulta satisfactoria	SI
PORCENTAJE			
Back - End	100%		
Front - End	100%		

F21: Gestionar Asignación de Facultades por Sede

Información General			
Nombre de caso de uso:	Gestionar Asignación de Facultades por Sede		
Prerrequisitos			
Deben existir registros previamente ingresados en Sedes y en Facultades			
Descripción de Casos de Prueba			
Caso: Se realiza el proceso para la creación, eliminación y consulta de Facultades por Sede			
Escenarios de prueba		**Respuesta esperada de la aplicación**	**Coincide (Si/No)**
#	**Nombre de la prueba a ejecutar**		
1	Ingreso de Sede	Ingreso satisfactorio	SI
2	Eliminación de Sede	Eliminación satisfactoria	SI
3	Consulta General de Sede	Consulta satisfactoria	SI
4	Consulta específica de Sede	Consulta satisfactoria	SI
PORCENTAJE			
Back - End	100%		
Front - End	100%		

F22: Gestionar Estudiante Excel

Información General		
Nombre de caso de uso:	Gestionar Estudiantes Excel	
Prerrequisitos		
Deben existir registros previamente ingresados en Carreras		
Descripción de Casos de Prueba		
Caso: Se realiza el proceso para la creación, modificación, eliminación y consulta de un Estudiante		
Escenarios de prueba		**Coincide (Si/No)**

#	Nombre de la prueba a ejecutar	Respuesta esperada de la aplicación	
1	Ingreso de Estudiante	Ingreso satisfactorio	SI
PORCENTAJE			
Back - End	100%		
Front - End	100%		

Capítulo 4

Conclusiones y Recomendaciones

Conclusiones

Al finalizar el Trabajo de Titulación y con el objetivo de demostrar los logros adquiridos en el desarrollo del documento, se obtuvieron los siguientes resultados y conclusiones:

1. Se realizaron las Fases de Análisis y Diseño del Sistema de Agenda Electrónica del Estudiante, con el fin de que el mismo pueda interactuar con la información del syllabus de la Asignatura y realizar las Anotaciones pertinentes que requiera y que incrementen su adquisición de conocimiento significativo por cada Tema de Estudio de una o varias Asignaturas. Para la construcción del sistema, se siguieron los estándares especificados en la Metodología Orientada a Objetos, que permitieron definir objetos de Clases con atributos y métodos propios en base a Controladores. Además, mediante el ciclo de vida por Fases para el desarrollo de software especificado por el Método de Cascada Sinergy, se recabaron las necesidades del Usuario (Estudiante). Y mediante la traducción de estas necesidades a Requerimientos Funcionales, se elaboraron diagramas tales como: Diagramas de Casos de Uso, Diagramas de Paquetes, Diagramas de Actividades, Diagramas de Clases, Diagramas de Secuencia y Diagramas de Entidad-Relación. Lo mencionado permitió entender de mejor manera el propósito del Sistema, y así se particularizó los objetos y procesos observados en la vida real dentro del Sistema Web.

2. Se implementó y documentó el Sistema Web en función al Análisis de las estructuras definidas por capas dentro de cada carpeta de Paquetes por módulo y mediante el uso del archivo web.php que permite manejar el canal de comunicación por rutas. En general, cada módulo utiliza 3 capas definidas como: Vistas/Blade (GUI), Controladores (DP) y Modelos (MD) que se comunican entre sí para redireccionar, usar y enviar datos entre sí; los cuales se presentan al Usuario tomando los principios del Diseño Universal para facilitar la interacción y potenciar el entendimiento. En base a esta lógica, se desglosaron 4 módulos base dentro del Sistema de Agenda Electrónica del Estudiante, de los cuales se explicaron su propósito y funcionamiento a detalle. Finalmente, se debe recalcar que, para la construcción del Sistema, se utilizó el Framework de código abierto Laravel que utiliza un Lenguaje de Programación Web PHP 5 y PHP 7 para elaborar Sistemas en la Nube.
3. Se elaboró las Pruebas Funcionales de los módulos del Sistema de Agenda Electrónica del Estudiante, en base al estándar de Pruebas definido por el Método de Cascada Sinergy. Aquí, se establece que para realizar un análisis de Pruebas Funcionales se necesita: especificar el nombre del Caso de Uso al cual el módulo hace referencia, analizar los prerrequisitos o relaciones declaradas en el segmento de validaciones por módulo, describir el propósito del módulo y el objetivo que debe cumplir, analizar cada funcionalidad que utiliza los métodos detallados en cada controlador por módulo (para ver si realiza lo que debe hacer y si coincide con lo deseado), y por último, determinar el porcentaje de compleción del Back-

End y del Front-End, en base a Vistas y Controladores de cada módulo mediante pruebas directas en tiempo real.

4. Durante el Ciclo de Vida de Desarrollo de Software se siguió una Metodología Tradicional por Fases, en base al Método de Cascada Modificada (Sinergy). El Método permitió iterar durante la Fase de Implementación del Sistema, con el objetivo de evidenciar el cumplimiento de los requerimientos obtenidos en la Fase de Análisis y la funcionalidad establecida en base a diagramas realizados en la Fase de Diseño; garantizando un nivel de calidad óptimo (funcionalidad total en base a los requerimientos definidos, tiempos de respuesta adecuados, concurrencia de usuarios ideal, etc). Finalmente, se realizó una Documentación adecuada de cada Fase definida dentro de cada capítulo del Trabajo de Titulación y se Implantó el Sistema de Agenda Electrónica del Estudiante usando un Servidor Local levantado en las computadoras personales de los Autores del Trabajo de Titulación.

5. Se utilizaron diferentes librerías para Gestionar Usuarios y Roles, para Construir Contenido Digital del Sistema (Imágenes, Videos, Audios, PDF, Ventanas Emergentes) y para Gestionar el Ingreso de Diversos Usuarios a través de Excel. De estas se pueden recalcar principalmente: Mathlib, que permite gestionar procesos de carga de documentos con fundamentos matemáticos formalizados para leer archivos planos .xls, .xlms, etc, Video.js y Carousel.js, que permiten manejar y presentar diferentes Imágenes, Videos y Audios usando protocolos de transmisión HTTP, Dash y HLS, Migraciones con Semillas de Datos presentadas

dentro del Framework Laravel para Generar Interfaces que permitan manejar Módulos de Inicio de Sesión al ser Implementadas, y ModalBox.js, que permite levantar ventanas emergentes que conservan la estructura única de un archivo HTML y cuya funcionalidad es 100% compatible con JavaScript y JQuery; por lo que además facilitaban la carga y presentación de Archivos PDF.

Recomendaciones

A continuación, se mencionan algunas sugerencias en base a escenarios que se presentaron durante la elaboración del presente Trabajo de Titulación:

1. Para el modelado de los Diagramas UML, se sugiere el uso de herramientas avanzadas, como es el caso de PowerDesigner, la cual permite visualizar, analizar y diseñar de manera sencilla. Esto es importante debido a que las Fases de Análisis y Diseño son la base previo a la elaboración del Sistema, es decir, si se las realiza bien, posteriormente en la Fase de Implementación, el desarrollo se lo realizará de la mejor forma posible; y por lo tanto, se presentarán menor cantidad de defectos durante el transcurso de esta Fase. Los Diagramas que se pueden modelar con esta herramienta son los Diagramas de Casos de Uso, Diagramas de Actividades, Diagramas de Secuencia, Diagramas de Clases, Diagramas de Paquetes y el Modelo Entidad-Relación para la creación de la Base de Datos del Sistema.

2. Se sugiere realizar pruebas y encuestas a los estudiantes que utilicen este sistema, con el fin de verificar si las nuevas herramientas tecnológicas, como un cuaderno virtual, mejoran las capacidades de entendimiento y aplica las Metodologías de Aprendizaje que se requiere para adquirir conocimiento significativo de mejor

manera por parte del Estudiante. Realizando este estudio, se puede saber si algunas clases o Asignaturas se pueden recibir virtualmente.

3. Se recomienda diseñar el proceso de migración de la información del sistema a formatos digitales administrables por el Estudiante. Esto permite generar un archivo descargable para que el Estudiante pueda acceder a sus Anotaciones de su Agenda Estudiantil, sin necesidad de tener acceso a Internet.

4. Promover el uso de herramientas y Frameworks como Laravel en las nuevas carreras de Ingeniería en Sistemas de Información e Ingeniería en Tecnologías de la Información, debido a que, posee muchas características que brindan un buen soporte al desarrollador durante la elaboración y desarrollo de páginas web, como por ejemplo la modularidad, lo cual permite extender la aplicación mediante la creación de nuevos módulos sin alterar y estropear módulos elaborados previamente, brindando un fácil mantenimiento al sistema. Además, posee una extensa documentación en su página, la cual permite comprender todos los elementos y características que ofrece.

5. Se recomienda el uso de AJAX en aplicaciones que requieren cargar recursos sin necesidad de refrescar la página. Por ejemplo, para cargar datos en base a selecciones previas sin recargar la página, como sucedió en los CRUDS básicos de este sistema y en la interface principal, para colorear las palabras del Glosario, cargar las preguntas de las Evaluaciones, cargar las Anotaciones del Estudiante, entre otras cosas.

Bibliografía

Castro, E., & Fernández, I. (2013). *El significado de innovar.* Madrid: CSIC.

De la Cruz, F. (s.f.). *Ingeniería de Software I.* Quito. Obtenido de ngeniería de Software I.

Gallego, A. (2019). *Laravel 5*. Obtenido de Gitbooks.io: https://ajgallego.gitbooks.io/laravel-5/content/capitulo_1_funcionamiento_basico.html

Gutierrez, D. (Septiembre de 2009). *UML - Diagramas de Paquetes*. Obtenido de Code Compiling: http://www.codecompiling.net/files/slides/UML_clase_05_UML_paquetes.pdf

Hernández, E. (2019). *Panorama del mercado laboral de profesionistas en México*. Obtenido de Universidad Autónoma Metropolitana: http://www.ejournal.unam.mx/ecu/ecunam2/ecunam0208.pdf

Hernández, R., & Greguas, D. (2010). Estándares de Diseño Web. *Ciencias de la Información*, 69-71.

Lozada, J. (2014). Investigación Aplicada: Definición, Propiedad Intelectual e Industria. *CienciAmérica*, 47-50.

Mace, R. L. (1941-1998). About the center: Ronald L. Mace.

MACE, R. L. (24 de Junio de 2018). *ABOUT THE CENTER: RONALD L. MACE*. Obtenido de Las Ideas Fundamentales del Diseño Universal: http://www.webmati.es/index.php?option=com_content&view=article&id=54:las-ideas-fundamentales-del-diseno-universal&catid=15&Itemid=160

Martí, E. (2013). *Representar el mundo externamente. La construcción infantil de los sistemas externos de representación.* Madrid: Antonio Machado Libros.

Otwell, T. (2019). *Introduction*. Obtenido de Laravel: https://laravel.com/docs/4.2/introduction

PostgreSQL Global Development Group. (2019). *PostgreSQL*. Obtenido de What is PostgreSQL?: https://www.postgresql.org/about/

Pressman, R. (2010). Ingeniería del software - Un enfoque práctico. En R. Pressman, *Ingeniería del software - Un enfoque práctico* (pág. 75). México: McGraw-Hill.

Quintanilla García, A. A., Peña Toledo, L. A., Montes Guillermo, E., & Nuñez Huamani, J. (2015). *Que es la Metodología orientada a objetos - Introducción a UML*. Obtenido de Academia: https://www.academia.edu/15943563/Que_es_la_Metodolog%C3%ADa_orientada_a_objetos_-_Introducci%C3%B3n_a_UML

Rumbaugh, J., Jacobson, I., & Booch, G. (2007). *El Lenguaje Unificado de Modelado - Manual de Referencia.* Madrid: Addison-Wesley.

Segovia, J. (2017). *Todo PostgreSQL*. Obtenido de Diferencias entre DDL, DML y DCL: https://todopostgresql.com/diferencias-entre-ddl-dml-y-dcl/

Viñas, M. (2013). *Adiós al papel: Escribir a mano en el iPad*. Obtenido de Totemguard.com: https://www.totemguard.com/aulatotem/2013/02/adios-al-papel-tomar-apuntes-y-hacer-ejercicios-en-el-ipad/

Zorrilla Gamboa, J. A., & Garzón Luces, J. F. (2016). *Sistema de evaluación para el aprendizaje continuo de los estudiantes de la Universidad de Guayaquil*. Obtenido de Repositorio.ug.edu.ec: http://repositorio.ug.edu.ec/bitstream/redug/17923/1/UG-FCMF-B-CISC-PTG.1177.pdf