


**ESCUELA DE INGENIERÍA EN SISTEMAS**

**Tema:**

# "DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN WEB PARA LA GESTIÓN AUTOMATIZADA DE PROCESOS ADMINISTRATIVOS EN LA ESCUELA DE EDUCACIÓN BÁSICA JERUSALÉN"

**Proyecto de investigación y desarrollo previo a la obtención del título de Ingeniero en Sistemas y Computación**

**Línea de investigación:**

INGENIERÍA DE SOFTWARE Y/O PLATAFORMAS EDUCATIVAS

**Autor:**

JUAN MARCELO VILLAGRÁN FIALLOS

**Directora:**

ING. MG. LILIANA DEL ROCÍO MENA HERNÁNDEZ

**AMBATO – ECUADOR**

**ENERO – 2019**

**PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR**

**SEDE AMBATO**

**HOJA DE APROBACIÓN**

**Tema:**

"DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN WEB PARA LA GESTIÓN AUTOMATIZADA DE PROCESOS ADMINISTRATIVOS EN LA ESCUELA DE EDUCACIÓN BÁSICA JERUSALÉN"

**Línea de investigación:**

INGENIERÍA DE SOFTWARE Y/O PLATAFORMAS EDUCATIVAS

**Autor:**

Pontificia Universidad Católica del Ecuador
BIBLIOTECA

JUAN MARCELO VILLAGRÁN FIALLOS

Liliana del Rocío Mena Hernández, Ing. Mg. f._____

**CALIFICADORA**

Enrique Xavier Garcés Freire, Ing. Mg. f._____

**CALIFICADOR**

Teresa Milena Freire Aillón, Ing. Mg. f._____

**CALIFICADORA**

Mónica Patricia Mena Moreno, Ing. Mg. f._____

**DIRECTORA ESCUELA DE INGENIERIA EN SISTEMAS**

Hugo Rogelio Altamirano Villarroel, Dr. f._____

Pontificia Universidad Católica del Ecuador
SECRETARÍA GENERAL
PROCURADURÍA

**SECRETARIO GENERAL DE LA PUCESA**

**AMBATO – ECUADOR**

**ENERO – 2019**

# DECLARACIÓN Y AUTORIZACIÓN

Yo: **JUAN MARCELO VILLAGRÁN FIALLOS,** con **CC. 180328611-9,** autor del trabajo de graduación intitulado: "DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN WEB PARA LA GESTIÓN AUTOMATIZADA DE PROCESOS ADMINISTRATIVOS EN LA ESCUELA DE EDUCACIÓN BÁSICA JERUSALÉN", previa a la obtención del título profesional de **INGENIERO EN SISTEMAS Y COMPUTACIÓN,** en la escuela de **INGENIERÍA EN SISTEMAS.**

1.- Declaro tener pleno conocimiento de la obligación que tiene la Pontificia Universidad Católica del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la Pontificia Universidad Católica del Ecuador a difundir a través de sitio web de la Biblioteca de la PUCE Ambato, el referido trabajo de graduación, respetando las políticas de propiedad intelectual de Universidad

Ambato, enero 2019



**JUAN MARCELO VILLAGRÁN FIALLOS**

**CC. 180328611-9**

Pontificia Universidad Católica del Ecuador

BIBLIOTECA

# AGRADECIMIENTO

Agradezco a Dios porque nunca me ha abandonado, y porque me ha dado la sabiduría necesaria para seguir adelante y cumplir una de las metas más importantes de mi vida.

Agradezco a toda mi familia, en especial a mis papás Marcelo y Rocío quienes han sido mis ángeles a lo largo de mi vida estudiantil y me han sabido alentar en los momentos más difíciles regalándome una palabra de amor cuando más lo necesitaba.

Agradezco también a mis suegros Noé y Laura por ayudarme a cumplir esta meta brindándome su apoyo, a la Lic. Fanny Hernández por permitirme realizar mi proyecto de investigación en su Escuela y a la Ing. Mg. Liliana Mena por su valiosa ayuda.

Agradezco finalmente a mi alma gemela, mi esposita Gardenia Gavilanes, gracias mi amor por todo el inconmensurable apoyo que me ha dado a lo largo de todo este proceso, y por todos los innumerables sacrificios que ha hecho para ayudarme a llegar hasta este punto, le amo mucho mi esposita.

# DEDICATORIA

Dedicado a mi familia, en especial a mis papás que gracias a su trabajo y esfuerzo nos han podido dar una educación de calidad, los amo mucho.

Dedicado a mi esposita Gardenia Gavilanes que nunca bajó los brazos y siempre confió en mí, le amo mucho mi bebé.

# RESUMEN

El presente proyecto de investigación es motivado por la necesidad de la Escuela de Educación Básica Jerusalén de contar con un sistema que facilite al personal administrativo su labor, es así que el Diseño e Implementación de una Aplicación web para la Gestión Automatizada de Procesos Administrativos era necesaria, con el objetivo principal de registrar las matrículas y pensiones de los 330 estudiantes pertenecientes a la Institución Educativa, misma que permita al personal administrativo optimizar el tiempo y los recursos por medio de procesos automáticos. En el proceso de desarrollo de la investigación se utilizó una metodología ágil que permitiera documentar de forma correcta cada una de sus fases, la metodología de desarrollo adaptativo de software es la ideal para el proyecto ya que su principal característica es el poder adaptarse a los cambios en los requerimientos y promover el trabajo en equipo con el propósito de que cada decisión sea tomada rápidamente, el DAS tiene fases iterativas que por medio de la recolección de requerimientos permite planificar adecuadamente cada ciclo de desarrollo, por lo cual apoyada por el instrumento de la entrevista ejecutada a las autoridades del establecimiento educativo se pudo determinar cuáles eran estos, mismos que sirvieron para desarrollar la aplicación. Como resultado de la implementación, la Escuela dispone de un sistema de información que optimiza el tiempo al momento de realizar sus procesos administrativos sin mencionar el hecho de que se puede consultar información de los estudiantes rápidamente y reducir de esta forma el consumo de papel.

**Palabras clave:** aplicación, matrículas, pensiones, desarrollo adaptativo.

# ABSTRACT

This research Project was motivated by the need of the Jerusalen Elementary School to rely on a system that facilitates the work of administrative staff. Therefore, the design and implementation of a web application for the automated management of administrative processes was needed. Its main aim is to register the enrollments and payments of all 330 students of the school, thereby allowing the administrative staff to optimize their time and resources through automatic processes. In the development process of the study, an agile methodology was used that helped to document each one of its phases correctly. The Adaptive Software Development methodology was ideal for the Project since its main feature is that it can quickly adapt to requirement changes and promote teamwork in order to improve quick decision making. Adaptive Software Development has iterative phases that make it possible to adequately plan each development cycle by the gathering of requirements. With the support of an interview with the school authorities, it was possible to determine what they were and the helped to develop the application. As a result of the implementation, the school now has a computer system that successfully streamlines administrative processes and allows for quick access to student data, not to mention that the school's use of paper has reduced

**Keywords:** application, enrollments, payments, adaptive development.

# TABLA DE CONTENIDOS

DECLARACIÓN Y AUTORIZACIÓN.....iii

AGRADECIMIENTO.....iv

DEDICATORIA.....v

RESUMEN.....vi

ABSTRACT.....vii

1. CAPÍTULO I..... 1

1.1. Antecedentes ..... 1

1.2. Problema ..... 3

1.3. Descripción del problema ..... 4

1.4. Preguntas básicas ..... 7

1.5. Justificación..... 8

1.6. Objetivos ..... 9

1.7. Meta..... 9

1.8. Fundamentos teóricos ..... 10

1.8.1. Aplicación ..... 10

1.8.2. Tipos de aplicaciones ..... 11

1.8.2.1. Aplicaciones nativas..... 11

1.8.2.2. Aplicaciones web ..... 12

1.8.2.3. Aplicaciones híbridas ..... 12

1.8.2.4. Desarrollo de aplicaciones web..... 15

1.8.2.5. Aplicaciones de Código Abierto ..... 16

1.8.2.6. HTML ..... 17

1.8.2.7. CSS ..... 19

1.8.2.8. JavaScript ..... 20

1.8.3. Arquitectura Cliente/Servidor ..... 21

1.8.3.1. Control de Versiones ..... 23

1.8.3.2. Características de la Arquitectura Cliente/Servidor ..... 26

1.8.3.3. Cliente ..... 28

1.8.3.4. Servidor ..... 30

1.8.4. Administración de Procesos ..... 32

1.8.4.1. Principios fundamentales de la Administración ..... 33

1.8.4.2. Definición de Proceso ..... 34

1.8.4.3. Proceso informático ..... 35

1.8.5. Herramientas de desarrollo ..... 36

1.8.5.1. Laravel ..... 37

1.8.5.2. Atom ..... 41

1.8.5.3. HeidiSQL ..... 41

1.8.5.4. Laragon ..... 42

1.8.5.5. Composer ..... 42

1.8.5.6. JavaScript ..... 42

2. CAPÍTULO II ..... 43

2.1. Metodología de la investigación ..... 43

2.1.1. Tipos de investigación ..... 44

2.1.1.1. Investigación bibliográfica ..... 44

2.1.1.2. Investigación de campo ..... 45

2.1.2. Técnicas de recolección ..... 45

2.1.2.1. Entrevista ..... 45

2.1.3. Instrumento de recolección de información ..... 45

2.1.4. Población ..... 46

2.2. Metodología de desarrollo ..... 46

2.2.1. Desarrollo adaptativo de software ..... 47

2.2.1.1. Especulación ..... 48

2.2.1.2. Colaboración ..... 49

2.2.1.3. Aprendizaje ..... 49

2.3. Diagnóstico de la gestión de procesos académicos de la Institución ..... 50

2.3.1. Lista de procesos académicos ..... 51

3. CAPÍTULO III ..... 54

3.1. Primera iteración ..... 54

3.1.1. Especulación ..... 54

3.1.1.1. Obtención de requerimientos ..... 54

3.1.1.2. Casos de uso ..... 55

3.1.1.3. Diagrama de datos ..... 67

3.1.1.4. Diccionario de datos ..... 68

3.1.1.5. Diseño ..... 71

3.1.2. Colaboración ..... 74

3.1.2.1. Programación ..... 74

3.1.3. Aprendizaje ..... 78

3.1.3.1. Control de calidad ..... 78

3.2. Segunda iteración ..... 79

3.2.1. Especulación ..... 79

3.2.1.1. Casos de uso ..... 79

3.2.1.2. Diagrama de datos ..... 109

3.2.1.3. Diccionario de datos ..... 110

3.2.1.4. Diseño ..... 115

3.2.2. Colaboración ..... 118

3.2.2.1. Programación ..... 118

3.2.3. Aprendizaje ..... 123

3.2.3.1. Control de calidad ..... 123

3.3. Tercera iteración ..... 124

3.3.1. Especulación ..... 124

3.3.1.1. Casos de uso ..... 124

3.3.1.2. Diagrama de datos ..... 137

3.3.1.3. Diccionario de datos ..... 138

3.3.1.4. Diseño ..... 139

3.3.2. Colaboración ..... 141

3.3.2.1. Programación ..... 141

3.3.3. Aprendizaje ..... 149

3.3.3.1. Control de calidad ..... 149

3.3.3.2. Implementación ..... 150

4. CAPÍTULO IV ..... 153

4.1. Análisis de resultados ..... 153

4.2. Validación de resultados ..... 176

4.3. Capacitación ..... 177

5. CAPÍTULO V ..... 178

5.1. Conclusiones ..... 178

5.2. Recomendaciones ..... 180

6. BIBLIOGRAFÍA ..... 181

7. ANEXOS ..... 186

# TABLA DE GRÁFICOS

## Imágenes

Ilustración 1.1. Elementos de una aplicación web ..... 16

Ilustración 1.2. Hojas de estilo minimizadas y comprimidas..... 19

Ilustración 1.3. Arquitectura Cliente/Servidor ..... 22

Ilustración 1.4. Control de Versiones Centralizado ..... 24

Ilustración 1.5. Elementos de un proceso informático ..... 34

Ilustración 1.6. Estructura de un proyecto en Laravel ..... 39

Ilustración 1.7. Arquitectura MVC ..... 39

Ilustración 2.1. Fases del desarrollo adaptativo de software..... 47

Ilustración 3.1. Diagrama de casos de uso para el inicio de sesión ..... 55

Ilustración 3.2. Diagrama de casos de uso del módulo de usuarios ..... 55

Ilustración 3.3. Diagrama de datos del módulo de usuarios ..... 67

Ilustración 3.4. Wireframe inicio de sesión ..... 71

Ilustración 3.5. Wireframe recuperar contraseña ..... 72

Ilustración 3.6. Wireframe reiniciar contraeña ..... 72

Ilustración 3.7. Wireframe dashboard..... 73

Ilustración 3.8. Wireframe gestion de usuarios..... 73

Ilustración 3.9. Wireframe gestion de roles ..... 74

Ilustración 3.10. Permiso para ver los usuarios..... 75

Ilustración 3.11. Permiso para crear un usuario ..... 75

Ilustración 3.12. Permisos para modificar y eliminar un usuario ..... 75

Ilustración 3.13. Método para crear un usuario..... 76

Ilustración 3.14. Método para modificar un usuario ..... 76

Ilustración 3.15. Método para crear un rol..... 77

Ilustración 3.16. Método para modificar un rol ..... 77

Ilustración 3.17. Diagrama de casos de uso matrículas (cursos y paralelos) ..... 79

Ilustración 3.18. Diagrama de casos de uso del módulo de matrículas..... 88

Ilustración 3.19. Diagrama de datos del módulo de matrículas ..... 109

Ilustración 3.20. Wireframe gestión de cursos..... 115

Ilustración 3.21. Wireframe gestión de paralelos..... 115

Ilustración 3.22. Wireframe gestión de docentes ..... 116

Ilustración 3.23. Wireframe gestión de ofertas ..... 116

Ilustración 3.24. Wireframe gestión de estudiantes ..... 117

Ilustración 3.25. Wireframe gestión de matrículas ..... 117

Ilustración 3.26. Wireframe gestión de pensiones ..... 118

Ilustración 3.27. Relaciónes de las ofertas ..... 119

Ilustración 3.28. Relaciones inversas de las ofertas..... 119

Ilustración 3.29. Método para crear una oferta ..... 119

Ilustración 3.30. Relaciones de los estudiantes..... 120

Ilustración 3.31. Método para crear un estudiante ..... 120

Ilustración 3.32. Código para elegir la fecha de nacimiento del estudiante..... 121

Ilustración 3.33. Método para actualizar las configuraciones..... 121

Ilustración 3.34. Relaciones de las matrículas ..... 122

Ilustración 3.35. Componente datatables para consultar y exportar datos..... 122

Ilustración 3.36. Diagrama de casos de uso del módulo de adelantos ..... 124

Ilustración 3.37. Diagrama de datos del módulo de adelantos..... 137

Ilustración 3.38. Wireframe gestión de cargos..... 139

Ilustración 3.39. Wireframe gestión de trabajadores ..... 140

Ilustración 3.40. Wireframe gestión de adelantos..... 140

Ilustración 3.41. Método para crear un cargo ..... 141

Ilustración 3.42. Método para cargar la tabla de cargos ..... 141

Ilustración 3.43. Rutas para el módulo de adelantos..... 142

Ilustración 3.44. Función para cargar la tabla de cargos..... 142

Ilustración 3.45. Datos de la tabla de empleados ..... 143

Ilustración 3.46. Relacions del módelo de rabajadores..... 143

Ilustración 3.47 . Método para crear un trabajador ..... 143

Ilustración 3.48. Función para crear un adelanto en la vista..... 144

Ilustración 3.49. Mótodo para registrar un adelanto ..... 144

Ilustración 3.50. Función para mostrar los adelantos registrados ..... 145

Ilustración 3.51. Función para la tabla de datos de los adelantos ..... 145

Ilustración 3.52. Plantilla para el desarrollo de la página web..... 146

Ilustración 3.53. Sección de cabecera de la plantilla ..... 147

Ilustración 3.54. Barra de navegación de la plantilla..... 147

Ilustración 3.55. Pié de página de la plantilla ..... 148

Ilustración 3.56. Función para contar estudiantes, docentes y ofertas ..... 148

Ilustración 3.57. Seleccionar versión de PHP ..... 150

Ilustración 3.58. Comandos para desplegar la aplicación..... 151

Ilustración 3.59. Administrador de archivos de Cpanel..... 151

Ilustración 3.60. Aplicación subida al servidor de Cpanel..... 152

Ilustración 4.1. Resultados de la pantalla de inicio de sesión ..... 153

Ilustración 4.2. Resultados de la pantalla principal de la aplicación ..... 154

Ilustración 4.3. Resultados de la pantalla de gestión de usuarios ..... 154

Ilustración 4.4. Resultados de la opción para crear un usuario..... 155

Ilustración 4.5. Resultados de la pantalla de gestión de roles..... 155

Ilustración 4.6. Resultados de la opción para crear un rol ..... 156

Ilustración 4.7. Resultados de la implementación de roles y permisos..... 156

Ilustración 4.8.Resultados de la pantalla de gestión de cursos ..... 157

Ilustración 4.9. Resultados de la pantalla de gestión de paralelos ..... 157

Ilustración 4.10. Resultados de la pantalla de gestión de docentes..... 158

Ilustración 4.11. Resultados de la pantalla de gestión de ofertas..... 158

Ilustración 4.12. Resultados de la opción para crear una oferta académica..... 159

Ilustración 4.13. Resultados de la pantalla de gestión de estudiantes..... 159

Ilustración 4.14. Resultados de la pantalla de gestión de matrículas..... 160

Ilustración 4.15. Resultados de la opción de matricular estudiante ..... 160

Ilustración 4.16. Resultados de la opción de consultar matrícula..... 161

Ilustración 4.17. Resultados del reporte de estudiantes matriculados..... 161

Ilustración 4.18. Resultados de la opción imprimir comprobante matrícula ..... 162

Ilustración 4.19. Resultados de la pantalla de gestión de pensiones..... 162

Ilustración 4.20. Resultados de la opción de crear pensión ..... 163

Ilustración 4.21. Resultados de la opción de consultar pensiones ..... 163

Ilustración 4.22. Resultados del reporte de pensiones ..... 164

Ilustración 4.23. Resultados de la opcion imprimir comprobante pensión..... 164

Ilustración 4.24. Resultados de la opción de ver estudiante ..... 165

Ilustración 4.25. Resultados de la opción de enviar recordatorios..... 165

Ilustración 4.26. Resultados del correo enviado como recordatorio..... 166

Ilustración 4.27. Resultados de la pantalla de gestión de cargos ..... 166

Ilustración 4.28. Resultados de la opción para crear un cargo..... 167

Ilustración 4.29. Resultados de la pantalla de gestión de trabajadores ..... 167

Ilustración 4.30. Resultados de la opción para crear un trabajador ..... 168

Ilustración 4.31. Resultados de la pantalla de grestión de adelantos ..... 168

Ilustración 4.32. Resultados de la opción para crear un adelanto ..... 169

Ilustración 4.33. Resultados de la opción para consultar un adelanto ..... 169

Ilustración 4.34. Resultados del reporte de adelantos ..... 170

Ilustración 4.35. Resultados de la opcion imprimir comprobante adelanto ..... 170

Ilustración 4.36. Resultados de la página de bienvenida del sitio web ..... 171

Ilustración 4.37. Resultados del conteo de estudiantes, docentes y ofertas ..... 171

Ilustración 4.38. Resultados de la página “Quienes Somos” del sitio web ..... 172

Ilustración 4.39. Resultados de la página “Docentes” del sitio web ..... 172

Ilustración 4.40. Resultados de la página “Servicios” del sitio web ..... 173

Ilustración 4.41. Resultados de la página de bienvenida con cache ..... 174

Ilustración 4.42. Resultados de la página principal de la aplicación con cache..... 174

Ilustración 4.43. Certificado de validación de resultados ..... 176

Ilustración 4.44. Capacitación a la Secretaria. ..... 177

**Tablas**

Tabla 1.1. Tipos de aplicaciones ..... 13

Tabla 1.2. Diferencias entre sitio web y aplicación web..... 18

Tabla 1.3. Ventajas y desventajas de la arquitectura cliente/servidor..... 27

Tabla 1.4. Ataques cibernéticos del lado del cliente. ..... 29

Tabla 1.5. Ataques cibernéticos del lado del servidor. ..... 30

Tabla 1.6. Principios fundamentales de la administración..... 33

Tabla 2.1. Tipos de investigación ..... 43

Tabla 2.2. Lista de procesos académicos ..... 51

Tabla 2.3. Proceso registro de la matrícula ..... 51

Tabla 2.4. Proceso actualización de datos de la matrícula ..... 52

Tabla 2.5. Proceso anulación de la matrícula ..... 52

Tabla 3.1. Caso de uso - Iniciar sesión ..... 56

Tabla 3.2. Caso de uso - Reiniciar contraseña ..... 57

Tabla 3.3. Caso de uso – Cerrar sesión ..... 58

Tabla 3.4. Caso de uso - Consultar usuario ..... 59

Tabla 3.5. Caso de uso - Crear usuario ..... 60

Tabla 3.6. Caso de uso - Modificar usuario ..... 61

Tabla 3.7. Caso de uso - Eliminar usuario ..... 62

Tabla 3.8. Caso de uso - Consultar rol ..... 63

Tabla 3.9. Caso de uso - Crear rol ..... 64

Tabla 3.10. Caso de uso - Modificar rol ..... 65

Tabla 3.11. Caso de uso - Eliminar rol ..... 66

Tabla 3.12. Tabla permissions ..... 68

Tabla 3.13. Tabla roles ..... 68

Tabla 3.14. Tabla users ..... 69

Tabla 3.15. Tabla migrations ..... 69

Tabla 3.16. Tabla permission_user ..... 70

Tabla 3.17. Tabla permission_role ..... 70

Tabla 3.18. Tabla role_user ..... 70

Tabla 3.19. Tabla password resets ..... 71

Tabla 3.20. Prueba de aceptación de módulo de usuarios ..... 78

Tabla 3.21. Caso de uso - Consultar curso ..... 80

Tabla 3.22. Caso de uso - Crear curso ..... 81

Tabla 3.23. Caso de uso - Modificar curso ..... 82

Tabla 3.24. Caso de uso - Eliminar curso ..... 83

Tabla 3.25. Caso de uso - Consultar paralelo..... 84

Tabla 3.26. Caso de uso - Crear paralelo ..... 85

Tabla 3.27. Caso de uso - Modificar paralelo ..... 86

Tabla 3.28. Caso de uso - Eliminar paralelo ..... 87

Tabla 3.29. Caso de uso - Consultar docente ..... 89

Tabla 3.30. Caso de uso - Crear docente..... 90

Tabla 3.31. Caso de uso - Modificar docente..... 91

Tabla 3.32. Caso de uso - Eliminar docente..... 92

Tabla 3.33. Caso de uso - Consultar oferta ..... 93

Tabla 3.34. Caso de uso - Crear oferta..... 94

Tabla 3.35. Caso de uso - Modificar oferta..... 95

Tabla 3.36. Caso de uso - Eliminar oferta..... 96

Tabla 3.37. Caso de uso - Consultar estudiante ..... 97

Tabla 3.38. Caso de uso - Crear estudiante..... 98

Tabla 3.39. Caso de uso - Modificar estudiante..... 99

Tabla 3.40. Caso de uso - Eliminar estudiante..... 100

Tabla 3.41. Caso de uso - Consultar matrícula ..... 101

Tabla 3.42. Caso de uso - Crear matrícula..... 102

Tabla 3.43. Caso de uso - Modificar matrícula..... 103

Tabla 3.44. Caso de uso - Eliminar matrícula..... 104

Tabla 3.45. Caso de uso - Consultar pensión..... 105

Tabla 3.46. Caso de uso - Crear pensión..... 106

Tabla 3.47. Caso de uso - Modificar pensión..... 107

Tabla 3.48. Caso de uso - Eliminar pensión..... 108

Tabla 3.49. Tabla settings ..... 110
Tabla 3.50. Tabla courses..... 110
Tabla 3.51. Tabla offers ..... 111
Tabla 3.52. Tabla teachers ..... 111
Tabla 3.53. Tabla parallels..... 112
Tabla 3.54. Tabla extras..... 112
Tabla 3.55. Tabla pensions..... 113
Tabla 3.56. Tabla registers..... 113
Tabla 3.57. Tabla students ..... 114
Tabla 3.58. Prueba de aceptación de módulo de matrículas ..... 123
Tabla 3.59. Caso de uso - Consultar cargo..... 125
Tabla 3.60. Caso de uso - Crear cargo ..... 126
Tabla 3.61. Caso de uso - Modificar cargo ..... 127
Tabla 3.62. Caso de uso - Eliminar cargo ..... 128
Tabla 3.63. Caso de uso - Consultar trabajador ..... 129
Tabla 3.64. Caso de uso - Crear trabajador..... 130
Tabla 3.65. Caso de uso - Modificar trabajador..... 131
Tabla 3.66. Caso de uso - Eliminar trabajador..... 132
Tabla 3.67. Caso de uso - Consultar adelanto..... 133
Tabla 3.68. Caso de uso - Crear adelanto..... 134
Tabla 3.69. Caso de uso - Modificar adelanto ..... 135
Tabla 3.70. Caso de uso - Eliminar adelanto ..... 136
Tabla 3.71. Tabla advances..... 138
Tabla 3.72. Tabla employees ..... 138
Tabla 3.73. Tabla positions..... 139

Tabla 3.74. Prueba de aceptación de módulo de adelantos..... 149

Tabla 4.1. Resultados del tiempo de carga con cache..... 175

Tabla 4.2. Resultados del tiempo de carga sin cache..... 175

# CAPÍTULO I

# FUNDAMENTOS TEÓRICOS

## 1.1.Antecedentes

La tecnología ha evolucionado de tal manera que las instituciones educativas deben adaptarse a la misma para poder proporcionar al cliente un servicio que esté acorde al mercado actual, su objetivo principal es alcanzar la satisfacción del cliente mediante los servicios prestados y para lograr este propósito se pretende otorgar servicios que tengan un valor añadido, es decir que sean convenientes, deseables y que cubran una necesidad de manera sencilla, al hacerlo se genera una ventaja competitiva y leal.

El poder apoyarse en la tecnología para cumplir funciones administrativas abarca mucho más que simplemente utilizar un computador para revisar el correo o para crear documentos de texto, aquello se refiere a que adicionalmente se puede instalar un *software* de computador, capacitarse en el funcionamiento de este y manejarlo diariamente con la finalidad de simplificar el trabajo realizado por una persona dentro de una institución educativa, independiente de la labor que se realice en la misma.

En concreto, si se maneja un *software* de computador que reúne todos los requisitos necesarios para otorgar al cliente un servicio eficaz y de muy buena calidad se genera la satisfacción de este que es la principal motivación para el ofertante y es la recompensa para el Departamento administrativo, es por esto por lo que el tener el apoyo de un *software* de computador hecho a la medida de las necesidades de la institución educativa es en retrospectiva brindar un servicio añadido.

La Institución educativa Jerusalén es una Escuela de educación básica que fue fundada en el año de 1986 y empieza sus actividades en el mismo año con un grupo selecto de docentes que día a día tendrían como meta el desarrollo integral del estudiante, para lograrlo la Escuela implementa técnicas de enseñanza modernas como lo es aprender por medio de juegos, estas técnicas permiten al estudiante desarrollarse en todos los ámbitos educativos por medio de juegos interactivos, los mismos que desarrollan plenamente su motricidad y su capacidad de raciocinio desde muy temprana edad.

Al obtener buenos resultados con los estudiantes y su metodología de enseñanza, la escuela se consolida como una Institución educativa de calidad en la ciudad de Ambato, como consecuencia de su buena práctica educativa la Escuela busca innovar su modo de operación y se plantea la idea de contar con el apoyo tecnológico en forma de un *software* de computador, por lo tanto se analizó la opción de desarrollar una aplicación web que permita gestionar de mejor manera sus procesos administrativos actuales, bajo esta premisa se busca una opción que facilite el ingreso y consulta de información para el personal que trabaja en la Escuela de educación básica.

A partir de la implementación de una aplicación para la gestión automatizada de procesos administrativos se pudo monitorear de manera exitosa el periodo lectivo, el mismo que contiene cursos, paralelos, docentes, estudiantes, matrículas y pensiones de los mismos, controlar el salario de los trabajadores mediante un simple rol de pagos y respaldar la información en una base de datos segura alojada en un servidor web, en resumidas cuentas, la institución educativa contó con una aplicación que le permitió estar a la vanguardia tecnológica dentro del demandante mercado de la educación y brindar un servicio que se encuentre a la altura de la Institución educativa.

## 1.2.Problema

En la actualidad es imprescindible apoyarse en la tecnología de la información para poder alcanzar los diferentes objetivos que tiene la Institución educativa y que la administración propone de una manera fácil y organizada, el hecho de poder contar con métodos de operación previamente definidos permite que los procesos de trabajo sean eficientes y fluyan de forma normal sin contratiempos o demoras no programadas.

Si se recibe cientos de solicitudes por día ya sea de consulta y entrada o salida de información, el tratarlas de manera manual provoca una demora no esperada puesto que es la práctica que se acostumbra en la Institución, lo ideal sería contar con un registro digital de la información que esté listo y siempre disponible para cualquier solicitud y que devuelva una respuesta detallada de la información solicitada.

Para la Escuela de educación básica Jerusalén la incapacidad de adaptarse a un cambio tecnológico resulta en el problema, el creer que la misma practica que funcionaba anteriormente va a tener el mismo impacto en la actualidad es conformarse con lo que se tiene, y es por esta razón que la Escuela busca a través de una aplicación mejorar sus procesos administrativos y respaldar de mejor manera los datos de esta.

Cuando existe una demora en atender las solicitudes existe una cola, las solicitudes en cola se van acumulando una tras otra lo que perjudica la dinámica de la Escuela, mientras que un atraso en los procesos de esta conlleva una pérdida de tiempo y recursos valiosos para la Institución educativa, con la implementación de una aplicación automática se busca resolver este inconveniente y optimizar el tiempo.

## 1.3.Descripción del problema

El Departamento administrativo de la Institución educativa es el área que más procesos tenía y por lo cual existía la necesidad de automatizarlos mediante una herramienta informática, al haber analizado los procesos de dicho departamento se pudo identificar cual era el problema principal que tenía la Escuela, es decir se observaron todas las tareas individuales que se cumplían y que al final conformaban un proceso exitoso.

Al haber observado los procesos de matriculación, cobro de aranceles y gestión del rol de pagos que la Escuela manejaba permitió conocer cómo eran en realidad y como se los podía mejorar con el uso de la tecnología para que en el futuro se los pueda realizar metódicamente y que sirvan para que la Institución educativa logre esa ventaja competitiva hacia el resto de los Establecimientos educativos de la ciudad de Ambato.

En época de matrículas al inicio del año lectivo se toma un grupo de Docentes al azar para que ayuden a la Secretaria a matricular a los estudiantes, la Secretaria o Docente entonces procede a tomar los datos personales de cada estudiante, así como también los datos personales de los padres de familia y los guarda en un folder en donde se tiene un número de hojas correspondiente al número de estudiantes de cada grado.

Posteriormente al momento de registrar el pago de la matrícula la persona a cargo lo realiza de forma manual, es decir recibe el dinero de la matrícula lo cuenta y es ahí cuando se registra el pago correspondiente en la hoja de cada estudiante almacenada en el folder de las matrículas, entonces hecho esto se entrega un comprobante de la transacción realizada con la fecha, el monto y la persona que realiza la transacción.

De la misma manera para registrar el pago de la pensión mensual el proceso actual consiste en buscar en el folder de las matrículas al estudiante por el apellido, al encontrarlo se saca la ficha del estudiante y se registra el pago junto con el valor de este, la fecha y la persona que lo hace, luego se desprende un recibo que no tiene valor tributario para entregar al representante legal como evidencia del pago realizado.

Si en un día se acercan a efectuar el pago 5 padres de familia, la primera persona se acerca a pagar la pensión mientras que las otras 4 esperan, en un escenario ideal habría más de una persona atendiendo para que la espera en la cola sea corta, sin embargo, en la escuela solo existe una Secretaria que es la que registra las pensiones de los estudiantes, es por esto por lo que se busca agilizar los procesos administrativos.

Así mismo con el rol de pagos, cuando los trabajadores desean un adelanto se debe llenar una solicitud, si la misma es aceptada se registra el préstamo de manera manual en un folder correspondiente a cada trabajador junto con la fecha, el monto y la firma del mismo como evidencia para ambas partes, luego a fin de mes se consulta en el mismo folder a ver si el trabajador ha hecho o no un adelanto, si lo ha hecho se verifica el monto y se realiza el descuento respectivo para luego entregar el sueldo.

Hay que mencionar también la ausencia de un portal web informativo, en la actualidad el tener un sitio web no es un lujo sino más bien se ha convertido en una necesidad, es una forma de llegar a mucha más población y para una institución educativa es importante ofertar los servicios de los que se pone a disposición al público en general, además de mantener a los padres de familia y partes interesadas informados de cualquier boletín o nota que publique la administración de la Institución educativa.

Ahora bien, la solución al problema inicia desde el momento en el que se matricula a un estudiante, para empezar, se debe crear un curso, un paralelo, y un docente, teniendo en cuenta que para cada curso habrá un docente, luego se debe crear un año lectivo con todos los cursos asociados y una vez hecho esto se procede a matricular a los estudiantes en cada curso teniendo en cuenta la oferta y los cupos disponibles.

A continuación, son las pensiones mensuales, en la aplicación se busca en una tabla de datos el nombre, apellido o número de cedula del estudiante y al encontrarlo se procede a registrar la pensión del mes para que de esta forma el arancel de esta se quede registrado en la aplicación y se pueda imprimir un comprobante, caso contario se puede enviar un correo electrónico recordando el pago de esta al representante legal.

Posteriormente es posible para el usuario generar un rol de pagos de los trabajadores de la Institución educativa: Docentes, Conserjes, Personal administrativo y Cocineros si los hubiere, para empezar, se registran los datos de cada uno de ellos en la base de datos junto con su cargo y sueldo mensual para que al momento de generar el rol de pagos la aplicación pueda verificar si el trabajador x ha realizado algún adelanto, la cantidad y la fecha del mismo y proceder a descontar del sueldo final del trabajador.

Finalmente, la Escuela no cuenta con un sitio web informativo que sirva como medio de comunicación para la Institución educativa, por todo esto nace el propósito de implementar una aplicación web para gestionar de forma automatizada los procesos administrativos de acuerdo con los estándares y normas educativas actuales, en la solución se despliega un sitio web de la Escuela de educación básica Jerusalén que aloje una pestaña para acceder a la parte administrativa de la aplicación.

## 1.4.Preguntas básicas

**¿Cómo aparece el problema que se pretende solucionar?**

- El problema surge como producto de una revisión de procesos administrativos realizado en el año 2016.

**¿Por qué se origina?**

- Se origina al no disponer de una aplicación para la gestión de procesos administrativos.

**¿Qué lo origina?**

- La falta de implementación de la tecnología en la lógica del negocio.

**¿Dónde se origina?**

- En la Escuela de educación básica Jerusalén.

**¿Dónde se detecta?**

- En los procesos administrativos de registro de matrículas y pensiones y generación de rol de pagos del Establecimiento educativo.

## 1.5.Justificación

El análisis de la investigación está motivado por la necesidad de la Institución educativa de agilizar y modificar sus procesos actuales usando una herramienta tecnológica que sea capaz de replicarlos satisfactoriamente y de esta forma comprobar que la mejor opción disponible para optimizar recursos es usar *software* distribuido libremente o de código abierto y que el mismo es útil para cualquier propósito.

De acuerdo con [1], en nuestro país existen normas laborales que dictan que el empleador debe remunerar al trabajador de acuerdo a horas trabajadas por mes, esto quiere decir que se debe tener un registro de las horas trabajadas del mismo para poder calcular el sueldo del trabajador, adicionalmente si el trabajador tuviese un atraso o inasistencia es deber del empleador descontar la cantidad respectiva del sueldo final.

Para reducir el riesgo de que la propuesta funcione en teoría, pero no en la práctica, el estudio de la implementación a realizar incorpora etapas de desarrollo que promueven la documentación y el análisis de requisitos, por lo tanto, se ha resuelto que el planteamiento de una aplicación que gestione los procesos administrativos de la Escuela servirá para cumplir las condiciones del negocio y alcanzar el objetivo.

Con la implementación de la aplicación informática se pretende marcar una diferencia en cuanto al servicio se refiere, al mismo tiempo que se promueve la seguridad informática de tener todos los datos respaldados, un mejor servicio permite a la Escuela de educación básica Jerusalén y a los padres de familia disfrutar de la calidad de un valor agregado y al Personal administrativo la comodidad de la tecnología.

## 1.6.Objetivos

### General

- Desarrollar una aplicación web para la gestión automatizada de procesos administrativos en la escuela de educación básica Jerusalén.

### Específicos

- Elaborar un marco conceptual sobre aplicaciones web y procesos administrativos.

- Diagnosticar la situación actual en la gestión de procesos académicos en la escuela de educación básica Jerusalén.

- Programar los módulos necesarios para la integración de ellos en la aplicación web.

- Capacitar a los usuarios finales de la aplicación.

## 1.7.Meta

Diseñar e implementar una aplicación web en la Escuela de educación básica Jerusalén para facilitar el manejo de pensiones, matrículas y rol de pagos de los estudiantes y los trabajadores respectivamente.

## 1.8.Fundamentos teóricos

### 1.8.1.Aplicación

De acuerdo con [2], la palabra aplicación proviene del latín *applicatio* que significa acción o efecto de aplicar algo, dentro de la informática [3] sugiere que una aplicación está diseñada para resolver un problema de una manera metódica, como, por ejemplo: una aplicación que sincroniza las notas del usuario en la nube. Si bien es cierto la idea no parece tan innovadora, la verdad es que su uso es altamente adaptable, ya que tener todas las notas respaldadas en un almacenamiento seguro es una idea muy rentable.

De acuerdo con [4], las aplicaciones nacen de la necesidad de poder desarrollar *software* informático de forma rápida y sencilla, una aplicación es ligera ya que se ejecuta por encima del sistema operativo, de esta manera el impacto de una aplicación en el sistema operativo es mínimo, el autor añade también que la flexibilidad que una aplicación ofrece al usuario al momento de utilizarla es lo que las hace tan populares hoy en día y es por lo cual el desarrollo de aplicaciones se encuentra en auge.

El termino aplicación según [5], se volvió popular con la aparición de los primeros dispositivos móviles de pantalla táctil, el *iPhone* fue el pionero de las aplicaciones ya que el usuario entraba en una tienda virtual, elegía la aplicación que necesitaba, se la descargaba y esta se instalaba automáticamente, posteriormente si el usuario la consideraba innecesaria simplemente la desinstalaba, es la sencillez con la que se manejan el motivo por el cual se prefiere una aplicación a todo un sistema informático debido a que en la actualidad el usuario prefiere que las cosas sean simples y eficientes.

### 1.8.2.Tipos de aplicaciones

Según [6], existen tres tipos de aplicaciones: aplicaciones nativas, aplicaciones web y aplicaciones híbridas, cada una de ellas está diseñada específicamente para desenvolverse mejor en un entorno especifico y cada una tiene sus propias ventajas y desventajas, para conocer mejor los tipos de aplicaciones es necesario investigar sobre las mismas para conocer más sobre cada una de sus características.

Basándose en términos de funcionalidad, rendimiento, experiencia de usuario (UX) y costos generales de desarrollo de cada tipo de aplicación se puede llegar a obtener una mejor interpretación sobre cuál es la mejor opción de desarrollo para cada proyecto enfocándose en el problema que se pretende resolver por medio de la aplicación.

#### 1.8.2.1.Aplicaciones nativas

A juicio de [7], una aplicación nativa es aquella que se ejecuta en la plataforma principal y depende de las características del *hardware* para funcionar, es como cualquier programa de computador que se instala en la memoria del dispositivo y se ejecuta en la máquina cada vez que el usuario así lo considere necesario.

Una aplicación nativa correctamente diseñada tiene muy buen rendimiento ya que los desarrolladores utilizan las características propias del dispositivo para obtener más beneficios, por ejemplo, una aplicación para poder visualizar fotos de alta definición, si el dispositivo no dispone de un controlador para visualizar imágenes de alta resolución entonces la aplicación no funciona adecuadamente y no se puede ejecutar.

#### 1.8.2.2.Aplicaciones web

Como señala [7], una aplicación web está compuesta por tres lenguajes de programación: *HyperText Markup Language* (HTML)*, Cascading Style Sheets* (CSS) y *JavaScript (JS),* con el uso de estos tres componentes se puede diseñar y desarrollar aplicaciones web dinámicas que sean responsivas al contenido de la página.

Las aplicaciones basadas en web se ejecutan en el navegador del dispositivo, como la red social *Facebook* que funciona en cualquier navegador web y que al mismo tiempo es compatible con cualquier dispositivo móvil, su diseño es bastante similar a la aplicación nativa que lleva el mismo nombre, pero esta no ocupa la misma memoria ni los mismos recursos que la aplicación nativa necesita.

#### 1.8.2.3.Aplicaciones híbridas

Según investigaciones de [7], las aplicaciones híbridas son desarrolladas usando librerías de las aplicaciones nativas, así como también de las aplicaciones web, esto para poder rescatar lo mejor de ambas aplicaciones y unirlas en una sola aplicación, una librería es un conjunto de funciones y una función es un fragmento de código.

Para ilustrar el concepto de una aplicación híbrida se toma, por ejemplo, una aplicación de geolocalización que funciona tanto para un terminal nativo como para un terminal móvil a la cual se puede acceder por medio del navegador web, entonces una aplicación híbrida puede funcionar con o sin conexión a internet y tener las mismas características que sobresalen de una aplicación nativa y de una aplicación web.

En la siguiente tabla se presenta un resumen de los tipos de aplicaciones tomando en cuenta la funcionalidad, el rendimiento, la experiencia de usuario (UX) y el costo de desarrollo de la aplicación en donde se detallan cuáles son las ventajas y desventajas de cada tipo de aplicación que existe en la actualidad:

Tabla 1.1. Tipos de aplicaciones

| | **Ventajas** | **Desventajas** |
|---|---|---|
| **Aplicación Nativa** | • Funcionalidad<br>• Rendimiento<br>• UX | • Costo |
| **Aplicación Web** | • Funcionalidad<br>• Rendimiento<br>• Costo | • UX |
| **Aplicación Híbrida** | • Funcionalidad<br>• UX | • Rendimiento<br>• Costo |

Fuente: [7]

En términos de funcionalidad la aplicación nativa supera al resto ya que la misma tiene acceso a todas las funciones del dispositivo mientras que la aplicación web y la híbrida ambas dependen de una conexión a internet, en términos de rendimiento la aplicación nativa igualmente supera al resto puesto que su interfaz, su estructura y sus datos están almacenados en la memoria del dispositivo y se encuentran listos para uso inmediato, estas están diseñadas específicamente para el sistema operativo en el cual se ejecuta, sin embargo depende ampliamente del desarrollador para mantenerse a la vanguardia.

En cuanto a la experiencia de usuario respecta, la aplicación web queda muy por detrás del resto ya que depende en su totalidad de la compatibilidad con el navegador web del dispositivo, si un dispositivo tiene un navegador web anticuado es posible que la aplicación web no se ejecute apropiadamente y algunas características no funcionen.

Sin embargo, es necesario resaltar que durante el desarrollo de una aplicación web se le puede añadir características para que la hagan responsiva al navegador, en cuanto al costo se refiere, la aplicación web sobresale por el resto, su costo de desarrollo e implementación es mínimo y su costo de mantenimiento no es tan elevado en comparación al costo de mantenimiento de una aplicación nativa o híbrida.

En definitiva, se pueden obtener varias razones por las cuales el enfoque de desarrollo de aplicaciones web prevalece sobre el resto de los tipos de aplicaciones:

- Orientado a proyectos con recursos limitados.
- Fácil control de versiones de la aplicación.
- Facilidad de mantenimiento del código base.
- No necesita de un asistente de configuración e instalación.
- Información centralizada, lista para ser respaldada.
- Requerimientos mínimos, solo necesita un navegador web.

#### 1.8.2.4.Desarrollo de aplicaciones web

El desarrollo de aplicaciones web nace de la necesidad de crear aplicaciones con funciones específicas que sean capaces de ser accedidas por los usuarios necesarios en cualquier momento y desde cualquier dispositivo de entrada, se puede utilizar herramientas de código abierto para el desarrollo y una vez que el código fuente ha sido escrito la aplicación puede ser ejecutada en cualquier plataforma que tenga un navegador web y para funciones de mantenimiento únicamente se debe revisar el código almacenado en el servidor web para que se actualicen todas las aplicaciones abiertas en los navegadores, esto hace que las aplicaciones web sean fáciles de desarrollar y por ende fáciles de mantener y de agregar nuevas funciones a la misma.

Como afirmara [8] en el año 2010, con la creciente popularidad de aplicaciones de código abierto, los desarrolladores han adoptado la idea de el desarrollo de aplicaciones para plataformas web sin costo, toda aplicación web empieza con tres componentes, HTML, CSS y JavaScript, cada una con un archivo correspondiente al lenguaje de programación al cual hace referencia, de acuerdo con ello se puede empezar el desarrollo de una aplicación robusta que tenga un buen rendimiento sin tomar en cuenta el pago de licencias corporativas para el uso de programas licenciados.

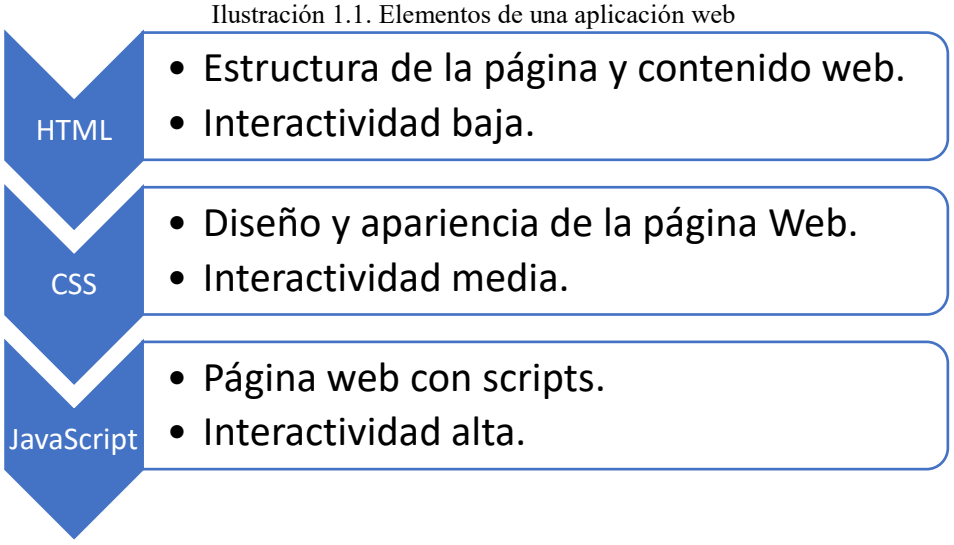
En sí, toda aplicación web necesita de archivos HTML para que el motor procesador de texto del navegador web despliegue la información de manera correcta, archivos CSS para darles estilo y diseño a las páginas de esta, así como también añadir colores y botones y por último archivos *JavaScript* para poder controlar el texto HTML mediante etiquetas y añadir funciones únicas a la aplicación web en su totalidad.

### 1.8.2.5.Aplicaciones de Código Abierto

Los programas de código abierto o también llamados *open source* son programas de computador que no tienen costo y se pueden utilizar abiertamente sin repercusiones legales por parte del creador de las mismas, [9] expresa claramente que es importante entender las obligaciones y los derechos que conlleva utilizar programas de código abierto ya que el mismo está disponible para todos y por lo tanto es modificable por todos y la libre distribución debe incluir de manera obligatoria en un repositorio web el código fuente del mismo para evitar código malicioso dentro de la aplicación.

Cabe resaltar la importancia de desarrollar aplicaciones con programas *open source* ya que esto para la Institución educativa significó una inversión ya que se obtuvo un producto que fue desarrollado y probado exitosamente sin haber realizado ningún gasto económico, sin embargo, ya que fue un proyecto universitario el mismo no puede ser distribuido libremente y por lo tanto el único beneficiario es la Escuela Jerusalén.

Ilustración 1.1. Elementos de una aplicación web




Fuente: Elaboración propia

### 1.8.2.6.HTML

HTML es el lenguaje de procesamiento de texto usado para el diseño de páginas web, [10] indica que HTML es usado para crear documentos electrónicos llamados páginas que se muestran en la web, cada página contiene una serie de conexiones a otras páginas llamadas hipervínculos, un hipervínculo es un vínculo que asocia un documento de hipertexto a otro similar. EL código HTML se asegura de que el texto esté correctamente formateado para que la información y las imágenes se muestren de manera correcta y el navegador web despliegue las mismas de la forma adecuada.

Según indica [11], HTML fue creado por Tim Berners-Lee en 1991, el creador oficial y fundador de lo que actualmente se conoce como W*orld Wide Web,* él tuvo la idea de crear una biblioteca virtual para controlar los documentos que en ella se alojan y compartir información entre investigadores, los sitios web inicialmente eran un grupo de páginas conectadas entre sí por medio de los hipervínculos, un conjunto de páginas web conforma lo que actualmente se llama sitio web con contenido dinámico.

Es necesario resaltar el hecho de que existe una gran diferencia entre página web y sitio web, un sitio web es más que solamente un grupo de páginas web, al nivel más bajo todo el contenido del sitio web está conectado por medio de estilos que mantienen una consistencia temática que le da un cierto grado de identidad al sitio web. De esta forma se tiene un lenguaje de programación basado en motores procesadores de texto que conforman una estructura basada en etiquetas las mismas que correctamente formateadas forman una página web, un conjunto de páginas web correctamente diseñadas y con contenido relacionado a su propósito es un sitio web informativo.

Además de lo anterior existe también una evolución de los sitios web a lo que ahora es una aplicación web, es válido resaltar la diferencia entre lo que es un sitio web y lo que es una aplicación web, a continuación, las diferencias más claras y sobresalientes:

Tabla 1.2. Diferencias entre sitio web y aplicación web

| **Sitio Web** | **Aplicación Web** |
|---|---|
| • Es informativo | • Es interactiva |
| • No depende de una base de datos | • Funciona siempre con una base de datos |
| • El costo de desarrollo es generalmente bajo | • El costo de desarrollo depende de las funciones de la aplicación |
| • No recibe solicitudes del usuario | • Recibe solicitudes del usuario y devuelve una respuesta |

Fuente: Elaboración propia

En consecuencia de lo expuesto, se puede conocer que inicialmente la información que las personas compartían por internet era estática y muy pocos sitios web eran los que tenían contenido dinámico para que el usuario interactúe con ellos, una aplicación web utiliza la metodología cliente/servidor en donde el navegador web es el programa cliente y se realiza cualquier servicio interactivo del lado del servidor por medio de una conexión a internet, entonces una aplicación web presenta contenido adaptado dinámicamente basado en parámetros de solicitud y consideraciones de seguridad importantes que hacen que la aplicación web cumpla con los estándares de seguridad permitiendo que el usuario navegue por la misma de forma segura y eficiente.

### 1.8.2.7.CSS

Al diseñar una aplicación web se toma en cuenta principalmente los requerimientos que tendrá la misma, sin embargo no se puede dejar de lado el comportamiento estético ya que una parte importante de la satisfacción del cliente es que el usuario tenga una buena experiencia al utilizar la aplicación, de acuerdo con [12], CSS es una abreviación de *Cascading Style Sheets* que significa hojas de estilo en cascadas y que trabaja junto con HTML para controlar la forma en la que el contenido web es presentado del lado del cliente sin importar el navegador web que utilice el usuario.

Ahora bien, el utilizar hojas de estilos al desarrollar la aplicación significa que el servidor web tiene que cargar un archivo más para que el navegador web lo presente al usuario, si una aplicación web tiene varias hojas de estilos de gran tamaño el servidor web tomaría más tiempo de lo normal para cargar y desplegar las hojas de estilos junto con el maquetado de la aplicación web, es por esta razón que se acostumbra a minimizar y comprimir las hojas de estilos para que su tamaño sea menor al original, algunas hojas de estilos tienen la extensión *.css.min* que significa que esas hojas de estilos se encuentran en un estado minimizado es decir su tamaño es menor al tamaño de la hoja de estilo original, las mismas permiten que el rendimiento de la aplicación web no se vea afectado por el número de hojas de estilos que ocupa la aplicación.

Ilustración 1.2. Hojas de estilo minimizadas y comprimidas

```
<!-- Bootstrap Core CSS -->
<link href="css/bootstrap.min.css" rel="stylesheet">
<!-- Custom CSS -->
<link href="css/sb-admin.css" rel="stylesheet">
<!-- Morris Charts CSS -->
<link href="css/plugins/morris.css" rel="stylesheet">
<!-- Custom Fonts -->
<link href="font-awesome/css/font-awesome.min.css" rel="stylesheet" type="text/css">
```

Fuente: Elaboración propia

Casi todas las páginas web o sitios web o aplicaciones web en la actualidad contienen hojas de estilos personalizadas con el color de la organización a la que representa junto con imágenes de la misma, si un sitio web empieza a generar un tráfico considerable en su servidor es posible que las hojas de estilos se demoren en cargar y en ocasiones que el sitio web no se visualice de la forma que pretende el desarrollador, es aquí donde la tecnología se pone al servicio del mismo y se puede optar por utilizar una *Content Delivery Network* (CDN) que significa en español una red de entrega de contenidos, [13] propone un servidor globalizado que aloja las librerías más populares de desarrollo y los guarda en cache en su servidor y únicamente se coloca un enlace donde se encuentra alojada la librería para poder hacer uso de ella en el diseño.

#### 1.8.2.8.JavaScript

*JavaScript* es un lenguaje de programación que utiliza *scripts* para ejecutar una función, es orientado a objetos y se utiliza principalmente para crear páginas web dinámicas, es un lenguaje para ejecutar comandos del lado del cliente lo que significa que su código fuente es procesado por el navegador web y por lo tanto sus funciones se pueden ejecutar una vez que se ha terminado de cargar la página web.

En este sentido [14] declara que *JavaScript* es utilizado para que la página web pueda realizar una mini función sin realizar una petición al servidor web y es por esto que los comandos ejecutados del lado del servidor son orientados a objetos y no generan tráfico innecesario ni solicitudes innecesarias al servidor web, es decir se identifica un elemento mediante una etiqueta y se trabaja en relación a dicho elemento encapsulado mediante la etiqueta para que pueda cumplir su función y ser ejecutado.

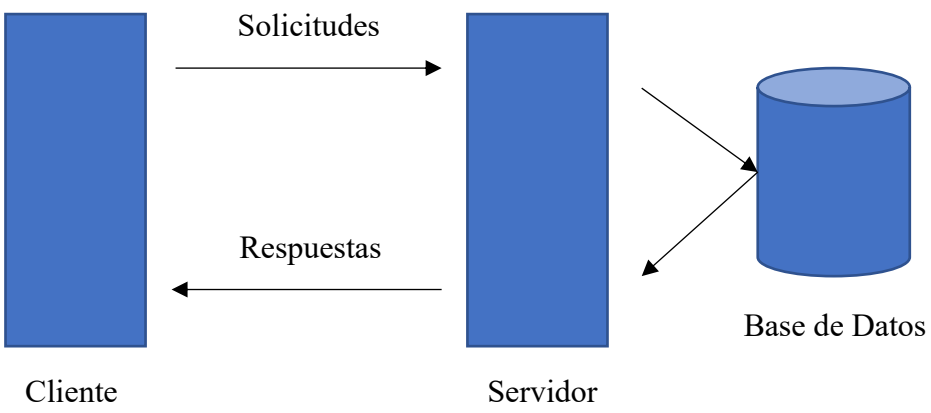
Una vez que se conoce cuáles son los elementos principales del desarrollo de aplicaciones web y se ha conceptuado cada uno de ellos se puede saber que función cumplen cada uno en el proyecto planteado, una aplicación web funciona con una metodología cliente/servidor, por lo tanto, es necesario revisar lo que significa un cliente y un servidor y cómo funcionan ambas en el desarrollo de aplicaciones web.

### 1.8.3.Arquitectura Cliente/Servidor

La arquitectura cliente/servidor está diseñada para que el servidor sea capaz de soportar varios clientes conectados al mismo tiempo y enviar varias respuestas sin que el rendimiento del cliente se vea afectado. De acuerdo con [15], esta arquitectura es muy común en proyectos pequeños que están basados en web, y describe la relación como dos programas de computadora en el cual el cliente realiza una solicitud de servicio y el servidor es el encargado de satisfacer dicha solicitud, ahora bien, se pude entender que el *software* es un servicio por lo tanto todo el desarrollo de la aplicación, actualizaciones, respaldo y mantenimiento se lo realiza únicamente en el servidor, de esta manera el *software* servicio se encuentra siempre actualizado y al día.

De acuerdo con lo anteriormente mencionado, para alcanzar el objetivo del proyecto planteado es necesario usar esta arquitectura para que la aplicación se pueda acceder en cualquier momento y en cualquier lugar siempre y cuando el Establecimiento disponga de una conexión a internet, la arquitectura cliente servidor se utiliza en la actualidad de manera muy frecuente, por ejemplo una aplicación web para revisar el correo o una aplicación web para control de inventario en un centro comercial, son algunos de los ejemplos en donde esta arquitectura se implementa de manera exitosa.

Ilustración 1.3. Arquitectura Cliente/Servidor




Fuente: Elaboración propia a partir de [15]

Es algo muy importante el poder tener el control de una sola versión en cuanto a una aplicación se refiere, el hecho de tener una sola versión significa que todos los clientes están actualizados y tienen todas las funciones implementadas dentro del programa de computador, al contrario si se mantienen distintas versiones de los clientes es posible que exista incompatibilidad en cuanto a funciones o nuevas características se refiere, dicha arquitectura hace que sea muy fácil controlar el historial de versiones de la misma ya que simplemente se mantienen dos aplicaciones, la primera que está alojada en el servidor y es la que el computador muestra al usuario por medio del navegador web y la segunda que es en la cual se desarrolla y se prueban nuevas funciones.

En la segunda el desarrollador tiene una versión exactamente igual a la versión original, desarrolla, realiza pruebas y aplica los cambios para implementar las nuevas funciones al cliente, [16] analiza el control de versiones y lo expone como una forma de controlar que cambios se han hecho de forma detallada permitiendo al grupo de desarrolladores revertir un cambio no deseado o revertir la aplicación a un estado anterior, esto se lo realiza para poder tener un control total del desarrollo.

#### 1.8.3.1.Control de Versiones

El control de versiones permite al desarrollador manejar de manera sistemática los cambios que se van a realizar al proyecto en desarrollo a la vez que protege el código fuente de consecuencias no planificadas ni tampoco deseadas, es un proceso esencial para el correcto desarrollo de *software*, de acuerdo con [17], las ventajas que tiene el controlar las versiones son varias, a continuación, las más importantes:

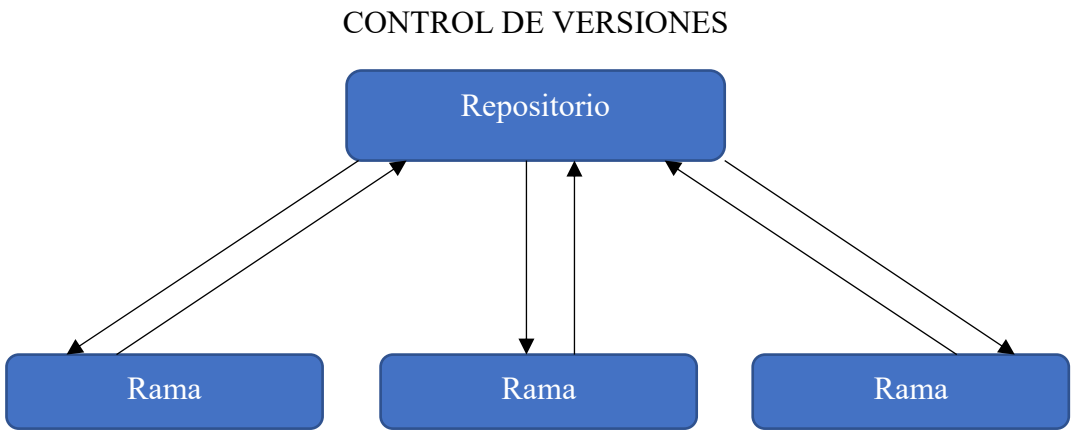
- Gestión organizada y coordinada de los cambios en los activos de la aplicación a lo largo del desarrollo, los cambios incluyen la creación y la eliminación de archivos, así como también la edición de sus contenidos.

- La creación de una "rama" brinda la posibilidad al equipo de desarrolladores de crear varios entornos de trabajo y que estos sean independientes entre sí, aquello permite crear una ramificación de la aplicación para desarrollar una nueva característica y fusionarla si es que no existen conflictos.

- La facilidad para el equipo de desarrollo de poder reemplazar a un miembro y que el nuevo miembro reciba un historial completo de cambios realizados promueve el diseño a largo plazo de la aplicación.

- El tener un respaldo de los archivos permite que el equipo de desarrollo se sienta seguro de alterar el código fuente de la aplicación, si el cambio no satisface las expectativas siempre se pueden revertir los mismos para volver a un estado previo a los cambios realizados.

Ilustración 1.4. Control de Versiones Centralizado




Fuente: Elaboración propia a partir de [17]

Como se muestra en la imagen anterior un control de versiones cuenta con un repositorio centralizado alojado en el servidor del cual emergen varias ramificaciones, de acuerdo con [18], las operaciones más comunes que se pueden realizar en un repositorio para controlar las versiones son las siguientes:

- **Crear:** Crea un nuevo repositorio vacío que es el lugar donde se empieza a versionar la aplicación, el repositorio creado aloja un directorio con varios subdirectorios para controles, modelos, vistas, y archivos de estilo, esto se lo puede realizar desde una ventana de comandos con el comando *git-init* que inicia un repositorio *git* para ser usado con el directorio de la aplicación.

- **Ramificación:** Crea una rama para trabajar en ella, esta duplica la versión original de la aplicación almacenada en el repositorio y genera una copia exacta utilizada para el desarrollo, lo ideal es crear una rama, modificarla e incorporar al repositorio las modificaciones realizadas y empezar el ciclo mencionado cuantas veces sea necesario durante la fase de desarrollo.

- **Combinar:** Esta operación combina los cambios realizados en la rama de desarrollo al repositorio, es recomendable siempre probar de forma exhaustiva las funciones implementadas antes de realizar esta operación.

- **Actualizar:** Actualiza la versión para que siempre se pueda desarrollar con la versión actual del repositorio, esto evita cambios que ya se han realizado en la aplicación y permite al desarrollador trabajar con la versión actual.

- **Revertir:** Esta operación como su nombre lo indica permite revertir la aplicación a su estado original cuando ya se aplicó un cambio defectuoso o un cambio innecesario para el objetivo de la aplicación.

- **Historial de cambios:** Esta operación o característica muestra un historial de los cambios realizados y sigue la pista de todas las versiones existentes en el repositorio, muestra detalles tales como quien hizo algún cambio, que fue lo que cambió y si el cambio tenía conflictos o no.

Estas son las operaciones más usadas cuando se utiliza un control de versiones para el desarrollo de aplicaciones web, adicionalmente se tienen operaciones tales como editar, eliminar, reiniciar que realizan la operación que su nombre indica.

Como se puede evidenciar usar un control de versiones al desarrollar aplicaciones web trae consigo muchos beneficios para el equipo de trabajo, todo con el objetivo de facilitar el trabajo realizado y de que todos los integrantes de este puedan hacer cambios al repositorio y trabajar en grupo de manera coordinada y ordenada.

### 1.8.3.2.Características de la Arquitectura Cliente/Servidor

La arquitectura cliente/servidor se usa en el desarrollo de aplicaciones web ya que es un método habitual el separar las dos funciones implicadas, es decir la función del cliente de utilizar los servicios y la función del servidor de proporcionar dichos servicios, dicha arquitectura trae consigo muchos beneficios que son:

- **Velocidad:** La velocidad es una característica que muy a menudo se pasa por alto, el usuario quiere que la aplicación responda de inmediato, si la aplicación es lenta la experiencia de usuario se deteriora, en la arquitectura cliente/servidor la velocidad depende de la conexión a internet.

- **Sencillez:** La arquitectura cliente/servidor resalta por su sencillez al momento de desarrollar e implementar, maneja una versión única alojada en un servidor web que se presenta al usuario por medio de un navegador web, es responsiva lo que significa que se adapta al dispositivo en el cual se la acceda.

- **Persistencia:** La persistencia significa que el servicio es persistente y puede ser accedido en cualquier momento, si un cliente falla o se desconecta, el resto de los clientes no se ve afectado por lo que el servicio es ininterrumpido.

- **Eficiencia:** El tráfico causado en el servidor durante el proceso de envío de mensajes se ve reducido por el hecho de contar con un servidor único centralizado, ya que el navegador web procesa los datos antes de enviarlos lo que resulta en una solicitud de servicio previamente definida.

En su trabajo de investigación [19] expone que el diseñar una aplicación que sea intuitiva es fundamental para determinar si el producto final es exitoso o no, si una aplicación cumple con los requerimientos iniciales pero su uso es complicado la experiencia del usuario se ve afectada, mientras que si una aplicación es fácil de usar la experiencia del usuario es positiva, el estudio citado se concentra en demostrar cómo usar las características mencionadas anteriormente para desarrollar una aplicación intuitiva que sea predictible y que tenga un grado de aprendizaje normal, de acuerdo con ello [20] añade que la arquitectura cliente/servidor tiene ventajas y desventajas que la hacen única, a continuación, se toman en cuenta las más importantes:

Tabla 1.3. Ventajas y desventajas de la arquitectura cliente/servidor.

| **Ventajas** | **Desventajas** |
|---|---|
| • El uso de *Structured Query Language* (SQL) soporta el acceso y facilita la consulta de información. | • El riesgo de que el servidor colapse por una congestión es una constante cuando se utiliza esta arquitectura. |
| • Todos los clientes tienen la oportunidad de acceder por medio de un navegador web. | • Cuando hay una mala conexión a internet las solicitudes llegan al servidor de manera incompleta. |
| • El soporte o mantenimiento de la aplicación es sencillo y el cliente no se ve afectado, esto se llama encapsulamiento. | • Se requiere de un análisis previo de compatibilidad para los clientes o navegadores web. |

Fuente: [20]

Una aplicación web que utiliza una arquitectura cliente/servidor necesita tener protocolos de seguridad tanto en el cliente como en el servidor, en el cliente se pueden añadir controles a los formularios para chequear que es lo que el usuario coloca como datos de entrada en los formularios y validarlos, mientras que en el servidor se puede colocar restricciones en la base de datos para solo aceptar un tipo de dato o una cantidad única de caracteres en un campo, es necesario conocer a fondo que es un cliente y un servidor, en qué consisten y como se coloca seguridad en los mismos.

#### 1.8.3.3.Cliente

El cliente es un *software* de computador encargado de reunir las solicitudes que el usuario genera y las pone en cola, presenta una interfaz agradable en donde se despliegan los datos ingresados y la información del sistema de una forma ordenada y estructurada al usuario, el cliente es el responsable de guiar al usuario e interactuar con el mismo para que pueda formular las solicitudes que serán enviadas al servidor, de esta forma se tiene una estructura sólida para el manejo de la información que el usuario envía por el dispositivo de entrada, como señala [21] el cliente se lo conoce también como *front-end* y es donde se originan todas las operaciones y es la parte que tiene una interacción directa con el usuario final de la aplicación,

El *front-end* de una aplicación es la parte en donde el desarrollador coloca los formularios que la aplicación necesita y es en donde se controla la información que el usuario ingresa, es considerada una buena práctica de desarrollo validar los datos de entrada antes de que el usuario los envíe y rechazar los datos de entrada erróneos, así como también mostrar un mensaje de error con una descripción de este.

Al implementar seguridad del lado del cliente se previenen ataques cibernéticos que usan un fragmento de código malicioso, un fragmento de código malicioso consiste en líneas de código insertadas en un campo de formulario que tienen la intención de generar un error provocado y alterar la información almacenada para que la aplicación no funcione de manera correcta, los ataques de lado del cliente más comunes son:

Tabla 1.4. Ataques cibernéticos del lado del cliente.

| Nombre | Descripción |
|---|---|
| **Inyección SQL** | La inyección SQL es un ataque que permite al atacante acceder a la base de datos de una aplicación y alterar la estructura de esta, una mala validación de datos ingresados conlleva a consultas SQL maliciosas inyectadas en un campo de formulario por lo tanto el navegador envía datos maliciosos al servidor, el mismo es capaz de eliminar todos los datos almacenados en una base de datos |
| **Inyección de código fuente** | La inyección de código fuente o código HTML es un ataque que modifica la presentación de un sitio web para que muestre información distinta a la esperada, por ejemplo, el atacante puede modificar una página para mostrar una imagen ofensiva o el atacante puede mostrar un formulario de inicio de sesión y de esa forma apropiarse de las credenciales de los usuarios |

Fuente: Elaboración propia

Ambos son considerados los más practicados por parte de los atacantes, cabe resaltar que depende en gran cantidad de la habilidad y conocimiento del atacante para que el ataque tenga un efecto irreversible en la base de datos, o para que el daño causado pueda ser corregido y posteriormente prevenido por parte del desarrollador.

### 1.8.3.4.Servidor

El servidor como su nombre lo indica es la parte encargada de proveer el servicio al cliente, [22] escribe que a un servidor se lo conoce también como *back-end* y es en donde se realizan todas las operaciones lógicas de la aplicación, recibe solicitudes y envía respuestas y de igual manera que el cliente el servidor necesita de seguridad implementada en su lado para que pueda proveer la seguridad que necesita el usuario.

En cuanto a la seguridad del lado del servidor es importante conocer los tipos de amenazas que existen y que un usuario mal intencionado puede llevar a cabo, en la opinión de [23] las amenazas más comunes son:

Tabla 1.5. Ataques cibernéticos del lado del servidor.

| Nombre | Descripción | Prevención |
|---|---|---|
| **Ataques de fuerza bruta** | El intruso intenta obtener acceso adivinando la contraseña de usuario a través de los servicios que se ejecutan en el sistema. | Contar con un límite para los intentos de inicio de sesión fallidos y deshabilitar el acceso desde la dirección IP. |
| **Denegación de servicio** | Los atacantes bloquean el acceso aumentando el tráfico de un sitio web para que el servidor de la víctima no responda. | La configuración de un *firewall* para controlar el tráfico de un sitio web permite conocer cuando el mismo excede al normal. |

Fuente: [23]

Es necesario recalcar que la amenaza más frecuente son los usuarios descuidados, causas como contraseñas fáciles de adivinar, software sin parches o antivirus desactualizados son las más habituales, por ello es necesario establecer una cultura de seguridad informática para que la información de la Institución se encuentre debidamente validada y sea almacenada de forma segura en la base de datos.

Recapitulando de manera breve, la seguridad del lado del cliente es la que valida la información antes de ser ingresada mientras que la seguridad del lado del servidor valida la misma una vez que ha sido ingresada y devuelve un mensaje de error al usuario, es considerada una buena práctica de desarrollo el implementar seguridad en la aplicación para que la misma no pueda ser vulnerada por ataques cibernéticos.

De este modo se puede concluir que el objetivo principal de la arquitectura cliente/servidor es separar sus partes principales, en este caso separar el cliente del servidor significa un fácil desarrollo y un fácil mantenimiento, básicamente mediante la modificación y actualización de la aplicación alojada en el servidor.

En definitiva, una aplicación web utiliza lenguajes de programación como HTML, CSS y *JavaScript* para diseñar la interfaz y añadir contenido interactivo para el usuario, la misma es desarrollada con una arquitectura cliente/servidor en donde se implementa seguridad de ambos lados para verificar que los datos ingresados sean válidos y una vez que han sido verificados no alteren la información almacenada en la base de datos, como complemento se utiliza un servicio para controlar las versiones en donde se tiene un historial de todos los archivos y los cambios que cada uno de ellos haya tenido además de poder crear ramas de la aplicación para desarrollo e investigación de esta.

### 1.8.4.Administración de Procesos

De acuerdo con [24], la palabra administración proviene del latín *ad-ministrare* que significa servir, es una acción que se realiza bajo el mando de una persona por lo tanto se sirve a una persona, en otras palabras es quien cumple una función mientras se encuentra bajo el mando de otra persona, administrar significa planificar, dirigir y controlar las actividades que se realizan en una Organización para alcanzar de manera eficiente los objetivos de la misma mientras que la administración se puede definir como un conjunto de actividades que grupos de personas realizan cooperando entre sí para alcanzar objetivos comunes así como también el bien común de la organización.

Una descripción más exacta de lo que significa administración se ofrece en [25] e indica que la administración es el proceso de administrar a los obreros y asignar los recursos de la organización de manera eficiente y efectiva mediante el uso de principios económicos para lograr la estabilidad, el crecimiento y la rentabilidad de la organización, en la era de la información es necesario saber administrar los recursos tecnológicos de los cuales se dispone para poder alcanzar un nivel de eficiencia óptimo y esto se logra apoyándose en sistemas informáticos y en la tecnología.

Estos principios económicos a los cuales se hace referencia son principios fundamentales de la administración en general y que fomentan una estructura de funcionamiento ordenada con guías, pautas y estándares que deben seguir los obreros para que su labor sea uniforme, una vez que todos los obreros tienen un estándar de trabajo se hace más fácil evaluar no solo el desempeño de estos, sino también el de la organización como conjunto y consecuentemente la toma de decisiones.

### 1.8.4.1.Principios fundamentales de la Administración

De acuerdo con [26] existen 4 principios fundamentales en los cuales se basa una buena administración, los mismos que se exponen a continuación:

Tabla 1.6. Principios fundamentales de la administración

| | |
|---|---|
| **Principio de planeación** | Constituye en erradicar la toma de decisiones por parte del trabajador y eliminar el sentido común y en su lugar utilizar métodos de trabajo previamente planificados y comprobados. |
| **Principio de preparación** | Es básicamente un proceso de selección de trabajadores basándose en sus aptitudes y su conocimiento para luego asignarlos a trabajos en función de su capacidad y motivación. |
| **Principio de ejecución** | Distribución de tareas de forma ordenada y responsable para que los trabajadores a cargo realicen sus tareas de manera metódica. |
| **Principio de control** | Proveer instrucciones y supervisar el rendimiento de los trabajadores para asegurarse de que cumplen con los estándares de la Organización. |

Fuente: [26]

Entonces con referencia a lo mencionado por el autor el acto de administrar una Organización consiste en planificar las distintas tareas, subtareas y actividades que realizarán los trabajadores quienes previamente deben pasar por un proceso de selección basado en sus conocimientos, su preparación y sus aptitudes individuales y grupales, este esfuerzo colectivo pasa por un proceso de monitoreo por parte del Personal administrativo quienes supervisan el desempeño de los trabajadores y de ser necesario capacitan y asignan responsabilidades a los mismos.

### 1.8.4.2.Definición de Proceso

La palabra proceso según [27] viene del latín *processus* que significa avanzar o moverse hacia adelante siguiendo un orden y con un objetivo específico, un proceso es un conjunto de actividades que se relacionan entre sí y que su objetivo es ir hacia adelante en una línea de tiempo para obtener un resultado visible, dentro de la informática un proceso se conforma por datos de entrada, un proceso central y los datos de salida, conforma una pequeña parte de un sistema operativo y puede ser ejecutado miles de veces según así lo disponga el usuario que lo controla.

Ilustración 1.5. Elementos de un proceso informático




Fuente: Elaboración propia a partir de [27]

#### 1.8.4.3.Proceso informático

Ahora bien, un proceso informático según [28] empieza con una fase de planificación que es en donde se planifica lo que hará dicho proceso, de que depende para que funcione y que tipo de resultados va a mostrar, posteriormente pasa por una fase de diseño que es en donde se diseña el proceso, es decir se da valores a los datos de entrada para que puedan ser calculados y se diseñan las funciones que van a calcular los datos de entrada, luego de varias pruebas el proceso pasa por una fase de implementación que es en donde se pone el proceso en producción para que funcione en tiempo real, finalmente pasa por una fase de validación que es donde se validan los resultados.

Por lo tanto, en la informática el administrar procesos significa planificar las distintas tareas, subtareas y actividades que realizarán los trabajadores, diseñarlas para que puedan ser procesos ejecutables que reciban datos de entrada estandarizados, y que devuelvan datos de salida concretos y validados acorde a las funciones desarrolladas durante el ciclo de vida de la aplicación y a lo que se necesita en la Organización.

En resumen, para desarrollar una aplicación web que funcione de manera correcta y que administre los procesos de la organización es necesario identificar dichos procesos, interpretar cada uno de ellos en un lenguaje de programación de modo que puedan recibir datos y convertirlos en una función que la aplicación pueda entender y ejecutar cuantas veces sea necesario, es decir replicar dichos procesos en la aplicación para finalmente automatizarlos, una vez que los procesos han sido diseñados se les asignan valores de entrada a los datos con los cuales se va a trabajar, finalmente se visualiza los datos de salida al usuario por medio de la aplicación web.

### 1.8.5.Herramientas de desarrollo

Para una mayor productividad y eficiencia se utilizaron herramientas que brindan al desarrollador una interfaz intuitiva para la elaboración de un proyecto desde su análisis hasta su implementación, para conseguir los objetivos del proyecto planteado se utilizaron las siguientes herramientas de desarrollo:

- Laravel que fue el *Framework* con el que se desarrolló el proyecto.
- Atom que fue el IDE que se usó para el desarrollo del proyecto.
- HeidiSQL como Sistema Gestor de Base de Datos.
- Laragon como servidor local.
- Composer para el manejo de librerías dependientes.
- JavaScript como lenguaje para el manejo de scripts.

Es necesario resaltar el hecho de que todas las herramientas utilizadas para el desarrollo del presente proyecto de investigación son de código abierto y se encuentran abiertas al uso de entidades públicas o privadas y que por lo tanto no fue necesario adquirir licencias operativas para lograr el objetivo planteado, adicionalmente las mismas traen manuales de operación para comprender su funcionamiento, sin embargo, si se necesita conocimiento básico de cómo funcionan dichas herramientas.

#### 1.8.5.1.Laravel

Para poder comprender de mejor manera cómo funciona *Laravel* y cuáles son sus componentes y su estructura es mejor tener claro el concepto de *Framework,* esto con el propósito de evitar confusiones al respecto, de acuerdo con [29], un *Framework* es un conjunto de librerías fácilmente personalizables que tiene una estructura propia y que es utilizado para el Desarrollo Rápido de Aplicaciones, al instalar un *Framework* se obtienen muchos beneficios que vienen de paquete, tales como:

- Integración sencilla con una base de datos (solamente se colocan las credenciales de la base de datos en un archivo de entorno).

- Extensa documentación de las funciones que cumple cada componente del *Framework,* foros y páginas web con tutoriales y guías de desarrollo.

- Al ser de código abierto no se depende de una licencia para tener las características más actuales de desarrollo.

- Rapidez al momento de acelerar el proceso de desarrollo y de implementar funciones de manera rápida y sencilla.

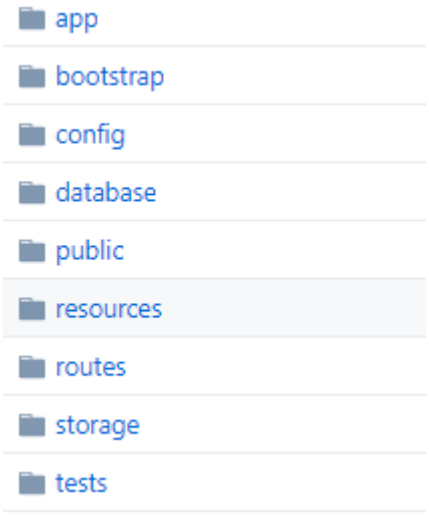
Es por esta razón que el uso de un *Framework* para realizar una aplicación web es una gran ayuda para el desarrollador ya que el mismo tiene una estructura prácticamente definida en donde solamente se tiene que añadir el resto de la aplicación (Modelos, Vistas, Controladores) además de también configurar la conexión a la base de datos.

*Laravel* es un *Framework* muy popular con una gran acogida por parte de los desarrolladores y es utilizado por los mismos para emprender proyectos simples y complejos, tiene una fácil adaptabilidad a la aplicación que se quiera desarrollar y cuenta con una gran base de desarrolladores dispuestos a compartir su conocimiento en el foro oficial del *Framework*, el mismo es altamente compatible con la estructura MVC (Modelo-Vista-Controlador) como se podrá ver más adelante en la redacción de este documento, de acuerdo con [30], *Laravel* es considerado la mejor opción para el desarrollo de aplicaciones web gracias a que es fácil de aprender y simple de usar.

El mismo viene por defecto con características importantes como por ejemplo el manejo de rutas y el desarrollo básico de operaciones CRUD para cualquier entidad que se encuentra relacionada con la aplicación, otra característica importante y digna de resaltar es la capacidad de mantener los registros activos para una consulta eficaz y eficiente por parte del usuario, dicho en otras palabras cada aplicación cuenta con una base de datos y en las mismas existen tablas que almacenan registros, los modelos son clases utilizadas para interactuar con los datos almacenados en las tablas de forma avanzada y sin la necesidad de escribir consultas con lenguaje estructurado (SQL).

*Laravel* cuenta con una estructura conformada por carpetas en donde cada carpeta almacena distintas subcarpetas y en estas existen clases las mismas que cumplen con un objetivo y un propósito especifico, por ejemplo la carpeta *app* contiene los archivos más importantes de la aplicación y en esta se encuentran los modelos, los controles y las reglas de validación, en la carpeta *resources* se encuentran los archivos que serán las vistas o páginas de la aplicación mientras que la carpeta *routes* contiene las rutas que la aplicación maneja, dicha estructura es la siguiente:

Ilustración 1.6. Estructura de un proyecto en Laravel

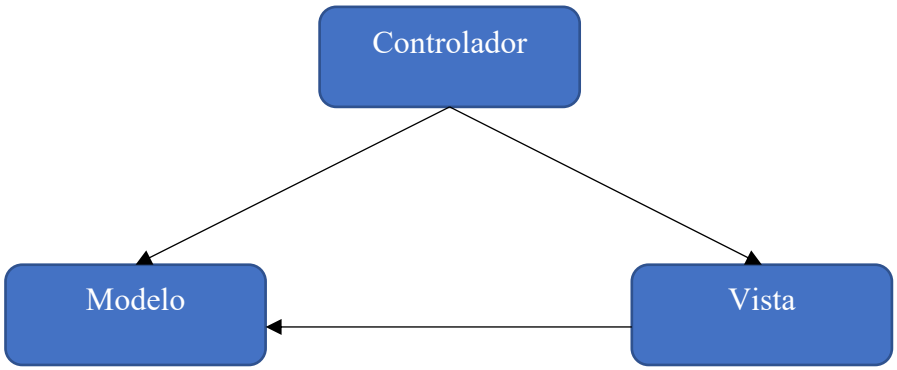



Fuente: Elaboración propia

La arquitectura MVC es una arquitectura usada para el desarrollo de aplicaciones web y su particularidad es la de diferenciar y separar la lógica de la aplicación de la interfaz de usuario para facilidad de diseño y desarrollo, de esta forma se puede diseñar la interfaz de usuario de una manera más sencilla mientras que de igual manera se puede desarrollar la lógica de la aplicación sin tener que comprometer la interfaz de esta, de acuerdo con [31], los 3 componentes de MVC cumplen una función específica para que en conjunto puedan funcionar de tal manera que la aplicación web sea fluida es decir no tenga ningún error y pueda ser ejecutada libremente por el usuario.

Dentro de *Laravel* los modelos son las entidades que se crean en la base de datos es decir las tablas, los controles se encuentran en la carpeta *app* y son los encargados de controlar como se ejecutan las respectivas funciones y adicionalmente se encargan de gestionar las operaciones CRUD de los modelos, mientras que las vistas son las páginas que contienen lenguaje PHP, HTML, CSS y JS y son previamente compiladas por el motor procesador de texto para convertirse en la interfaz de usuario.

Ilustración 1.7. Arquitectura MVC




Fuente: Elaboración propia a partir de [31]

- Modelo: El modelo es la parte que representa las tablas de datos de la aplicación y que contiene los métodos para usarlos, es una parte de la arquitectura con la cual el usuario no interactúa por motivos de seguridad.

- Vista: La vista consiste en la interfaz de usuario, es en donde el usuario genera todas las solicitudes de servicio por medio de botones o enlaces predefinidos, la misma tiene una interacción directa con el usuario.

- Controlador: El controlador es el encargado de controlar las operaciones que se ejecutan del lado del cliente, como se puede observar en la ilustración 1.7 el controlador se relaciona al mismo tiempo con la vista y con el modelo.

  La relación con la vista le permite recibir los datos ingresados por el usuario y de ser necesario visualizarlos, mientras que la relación con el modelo le permite almacenar esos datos de manera segura en la base de datos, cabe resaltar que es en los controles donde se encuentran la mayoría de las funciones almacenadas en archivos que se encuentran en subcarpetas.

#### 1.8.5.2.Atom

Atom es un entorno de desarrollo integrado, diseñado para el desarrollo ágil de aplicaciones web y móviles, las características más importantes de esta herramienta se las encuentra en [32] y son:

- Marca líneas, coincide con palabras y paréntesis y destaca el código fuente de forma sintáctica y semántica lo que permite fácilmente editar y generar código.

- Presenta una vista profunda de los archivos que contiene el proyecto al mismo tiempo que muestra una vista lateral de las carpetas y subcarpetas que existen.

- Subraya los errores durante la codificación lo que permite una fácil depuración.

- Es altamente compatible con todos los sistemas operativos independientemente del lenguaje de programación que se va a usar.

#### 1.8.5.3.HeidiSQL

Es un sistema gestor de base de datos (SGBD) *open-source* que se encuentra a disposición de los desarrolladores que buscan agilidad, no ocupa muchos recursos por parte del computador por lo que es liviano, como sugiere [33], existen varias razones para usar *HeidiSQL* como SGBD la más importante es la alta compatibilidad que tiene con los motores de base de datos más populares que existen en el mercado actual, la capacidad de alto rendimiento que tiene el mismo lo convierte en un excelente SGBD.

#### 1.8.5.4.Laragon

*Laragon* instala un servidor local de manera rápida y sencilla, desarrollado como un programa ejecutable que permite instalar y configurar el servidor *Apache* para utilizarlo como un servidor de prueba, de acuerdo con [34], *Laragon* es una distribución simple y ligera que hace que sea extremadamente fácil para los desarrolladores crear un servidor web local para fines de pruebas y despliegue de aplicaciones.

#### 1.8.5.5.Composer

*Composer* es una herramienta para manejo de dependencias en PHP, permite declarar las librerías de las cuales el proyecto depende y las instala de una forma fácil lo que permite que el desarrollador las actualice posteriormente de ser necesario, de acuerdo con [35], el beneficio principal de usar *Composer* para el proyecto planteado es un manejo de librerías liviano que instala los componentes de la aplicación.

#### 1.8.5.6.JavaScript

*JavaScript* es un lenguaje de programación orientado a objetos, según [36], *JavaScript* es un lenguaje complementario que puede ser conectado a los objetos en su entorno para poder programar un control sobre ellos ya sea del lado del cliente o del lado del servidor lo que permite ampliar de esta forma sus funciones y su implementación, el mismo es ejecutado una vez que el usuario ha realizado alguna acción por lo tanto su objetivo es interactuar con eventos programados.

# CAPÍTULO II

# METODOLOGÍA

## 2.1.Metodología de la investigación

De acuerdo con [37], la metodología de la investigación es una combinación de prácticas, procedimientos, procesos, rutinas y convenciones utilizadas por el investigador para obtener información detallada y especifica de un evento en particular, al mismo tiempo establece que existen dos enfoques de la investigación, el enfoque Cuantitativo y el enfoque Cualitativo, mismos que se detallan a continuación:

Tabla 2.1. Tipos de investigación

| Investigación Cuantitativa | Investigación Cualitativa |
|---|---|
| Características | |
| Mide fenómenos | Explora los fenómenos en profundidad |
| Utiliza la estadística | No utiliza la estadística |
| Procesos | |
| Inductivo | Deductivo |
| Secuencial | No tiene una secuencia definida |
| Analiza una sola realidad | Analiza múltiples realidades |
| Ventajas | |
| Generalización de resultados | Profundidad de los resultados |
| Predicción | Riqueza interpretativa |
| Precisión | Amplitud |

Fuente: Elaboración propia a partir de [37]

De acuerdo con el autor, son varias las ventajas que se obtienen al usar cualquier enfoque investigativo, sin embargo, se debe definir qué tipo de investigación es la que complementa el desarrollo del proyecto, en este caso en particular; la riqueza interpretativa que se obtiene con la investigación cualitativa permite al investigador interpretar los resultados por medio de preguntas de investigación y los resultados varían de acuerdo al estudio en particular, la acción indagatoria es dinámica y se mueve entre los fenómenos y su interpretación por lo que la secuencia no siempre es la misma, por esta razón, para el desarrollo del proyecto planteado se utilizó un enfoque cualitativo para la recolección de información y para la investigación de los hechos.

### 2.1.1.Tipos de investigación

Los tipos de investigación utilizados para el desarrollo del proyecto son los siguientes:

#### 2.1.1.1.Investigación bibliográfica

Con la ayuda de la investigación bibliográfica se pudo obtener importante información, la misma que se encuentra resumida en el capítulo I de este documento, tal información detallada es sobre los tipos de aplicaciones que existen en la actualidad, las distintas arquitecturas que se emplean en el desarrollo de aplicaciones web, qué es un proceso y la administración de ellos, qué es un *Framework* y sus características, la documentación de las distintas herramientas de desarrollo utilizadas, y finalmente información referente a la metodología de desarrollo empleada, todo aquello con el propósito de elaborar un marco conceptual para que el lector pueda conocer más sobre el proyecto desarrollado y sobre las fases y actividades de la metodología utilizada.

#### 2.1.1.2.Investigación de campo

En la Escuela de educación básica Jerusalén ubicada en las calles Pichincha Alta y José Mires parroquia Celiano Monge de la ciudad de Ambato específicamente en el Departamento administrativo, se llevó a cabo una investigación de campo con el propósito de conocer cómo se realizaban los procesos administrativos y poder observar cómo se manejaba la información, por medio de dicha investigación fue posible determinar cuáles eran los requerimientos de la aplicación y de los usuarios.

### 2.1.2.Técnicas de recolección

#### 2.1.2.1.Entrevista

La entrevista es un instrumento de recolección de información de la investigación cualitativa considerado como una reunión para conversar e intercambiar información entre el entrevistador y el entrevistado con el objeto de generar datos, en este caso la entrevista es de tipo estructurada y se la realizó a la Licenciada Fanny Hernández, Directora de la Institución, porque es la autoridad que más información puede proveer y se la realizó con el fin de conocer cómo se ejecutaban los procesos administrativos.

### 2.1.3.Instrumento de recolección de información

El instrumento de recolección de información para el proyecto planteado se puede encontrar como una guía de la entrevista efectuada en la parte final del documento con el nombre de anexo 1 para mejor referencia de la investigación.

### 2.1.4.Población

La población a estudiar en la investigación en este caso consistió en una sola persona que es la Directora de la Institución, misma que participó en la entrevista estructurada.

## 2.2.Metodología de desarrollo

La metodología de desarrollo tiene un gran impacto en el ciclo de vida de la aplicación, es lo que dicta qué se va a hacer, el momento que se lo hará y cómo se lo debe hacer, escoger la metodología adecuada es muy importante ya que de acuerdo con la misma se puede controlar el progreso en tiempo real del desarrollo del proyecto en general.

De acuerdo con [38], la metodología de desarrollo de una aplicación es un esfuerzo combinado entre el departamento técnico y el departamento administrativo con el objetivo de transformar las necesidades, expectativas y limitaciones del cliente en una solución y posteriormente respaldar esa solución a lo largo de toda su trayectoria.

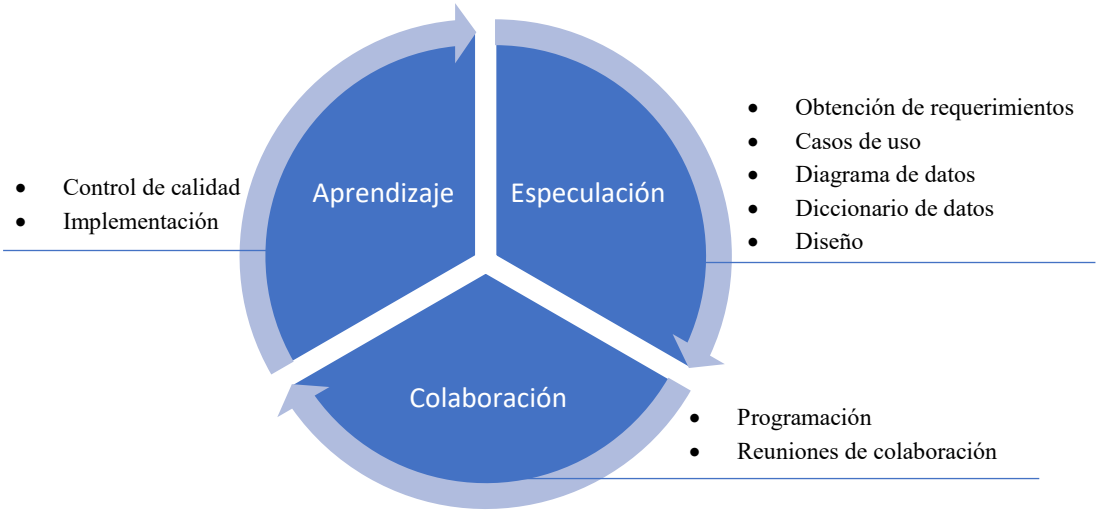
Las metodologías de desarrollo de *software* han evolucionado para poder adaptarse al mercado actual de desarrollo de aplicaciones en donde el cliente desea estar al tanto del progreso alcanzado mediante entregables, un entregable es un producto o parte de uno que se elabora para medir el progreso de un proyecto, ante la necesidad constante de desarrollar proyectos pequeños sin usar una metodología compleja y posiblemente anticuada surge la perspectiva de una metodología ágil de desarrollo que está orientada a proyectos de pequeño a mediano alcance, en este caso debido a que *Laravel* se actualiza constantemente se utilizó una metodología de desarrollo adaptativa.

### 2.2.1.Desarrollo adaptativo de software

El Desarrollo Adaptativo de *Software* (DAS) es una metodología ágil de desarrollo que favorece el prototipado rápido de aplicaciones, de acuerdo con [39] el DAS es tolerante a los cambios del cliente y reconoce que dentro del ciclo de desarrollo de *software* pueden existir cambios, por lo tanto establece que el cliente debe colaborar durante el desarrollo y durante las pruebas técnicas con el propósito de que en el control de calidad se apruebe cada cambio y se pueda empezar una nueva iteración.

El DAS es iterativo y en cada iteración se desarrolla, se prueba, y de ser necesario se corrigen errores y finalmente se despliega un módulo de la aplicación, según [40], las fases y actividades del DAS son las siguientes:

Ilustración 2.1. Fases del desarrollo adaptativo de software




Fuente: Elaboración propia a partir de [40]

#### 2.2.1.1.Especulación

En esta fase se realiza el levantamiento de toda la información necesaria y se empieza a planificar cada iteración mediante la realización de las siguientes actividades:

- **Obtención de requerimientos:** A partir de esta etapa se obtienen y se analizan los requerimientos del cliente por medio de reuniones, historias de usuario o entrevistas pactadas previamente, con las respuestas de las personas entrevistadas se procede a realizar el análisis de los requerimientos y se construyen los casos de uso en donde se define el número de componentes a desarrollar y las veces que se repetirá el ciclo de desarrollo de la aplicación.

- **Casos de uso:** Los casos de uso se elaboran con la información obtenida a partir de las entrevistas o reuniones realizadas con el Personal administrativo y cada uno de ellos representa en detalle una función o característica del sistema.

- **Diagrama de datos:** El diagrama de datos es un diagrama entidad-relación de la estructura de las relaciones que tienen las tablas que conforman la base de datos de la aplicación.

- **Diccionario de datos:** El diccionario de datos contiene información de todos los campos que contienen cada una de las tablas de la base de datos.

- **Diseño:** El diseño es un breve maquetado de la interfaz de la aplicación.

#### 2.2.1.2.Colaboración

En esta fase se desarrolla la aplicación según los requerimientos obtenidos de la fase de especulación, se desarrolla iteración por iteración en donde al final de cada una de estas se presenta al cliente un entregable, función o prototipo de la aplicación, en esta fase se realizan las siguientes actividades:

- **Programación:** Aquí el equipo de trabajo construye la aplicación, se procede inicialmente a instalar el *Framework* y las librerías que se van a utilizar para cada iteración por medio de un gestor de dependencias, posteriormente se editan las vistas de la aplicación y se crea la interfaz de usuario, se añaden funciones en los controladores pertenecientes a cada componente, al final estas se encargan de interactuar con cada modelo de datos respectivamente.

- **Reuniones de colaboración:** Esta actividad es opcional ya que durante el desarrollo se presentan obstáculos que necesitan de la intervención del cliente, de ser necesario en este caso se programa una reunión con este para que el mismo pueda colaborar en el desarrollo del componente, se lo realiza si existe alguna duda no aclarada por parte del desarrollador.

#### 2.2.1.3.Aprendizaje

Una vez que la fase de colaboración termina un componente empieza la fase de aprendizaje en donde se realizan pruebas para controlar la calidad del producto desarrollado, en esta fase se realizan las siguientes actividades:

- **Control de calidad:** Para realizar esta actividad se convoca al cliente a una reunión en donde este interactúa con el componente desarrollado y se realiza una prueba de aceptación en donde se le presenta al cliente el módulo desarrollado con cada una de las funciones implementadas y se registra en un documento la respuesta de este ya sea positiva o negativa, si existen cambios o nuevos requerimientos el ciclo de desarrollo regresa a la fase de colaboración en donde se implementan los cambios solicitados, caso contrario se aprueba el componente y se continua con la siguiente iteración.

- **Implementación:** Cuando se consigue la aceptación final del cliente del ultimo modulo desarrollado entonces se procede a implementar la aplicación en el servidor y documentar todo tipo de credenciales que surgieren en el proceso para un posterior soporte y mantenimiento de la aplicación desarrollada.

## 2.3.Diagnóstico de la gestión de procesos académicos de la Institución

Por medio de la investigación de campo realizada en la Institución Educativa se pudo conocer el estado de la gestión de procesos académicos de esta, dichos procesos fueron identificados y redactados por las autoridades de la Escuela y se encuentran al final del documento con el nombre de anexo 2 para mejor referencia de la investigación.

Es necesario resaltar el hecho de que el diagnóstico de los procesos académicos redactados en el anexo 2 se realizó únicamente a los procesos que se relacionaban con alguna función o característica de la aplicación, es decir a los procesos académicos relacionados con las matrículas y pensiones de los estudiantes de la Escuela.

### 2.3.1.Lista de procesos académicos

La lista de procesos académicos de la Institución se documenta a continuación:

Tabla 2.2. Lista de procesos académicos

| Procesos |
| --- |
| - Registro de la matrícula estudiantil.<br>- Actualización de datos de la matrícula estudiantil.<br>- Anulación de la matrícula estudiantil.<br>- Certificado de estudios.<br>- Planificación de la oferta estudiantil por parte de los docentes.<br>- Cierre del año lectivo. |

Fuente: Unidad Educativa Jerusalén

Documentación y diagnóstico del proceso de registro de la matricula estudiantil.

Tabla 2.3. Proceso registro de la matrícula

| | |
| --- | --- |
| Proceso | Registro de la matrícula estudiantil. |
| Subproceso | - Registro de datos del estudiante.<br>- Registro de datos de los representantes.<br>- Registro de datos de la matrícula. |
| Descripción | La Secretaria entrega una hoja para que los representantes llenen los datos de la matrícula, posteriormente el representante devuelve la hoja a la Secretaria quien la archiva. |
| Diagnóstico | Se pudo observar hojas con la información incompleta. |

Fuente: Unidad Educativa Jerusalén

Documentación y diagnóstico del proceso de actualización de datos de la matricula estudiantil.

Tabla 2.4. Proceso actualización de datos de la matrícula

| | |
|---|---|
| Proceso | Actualización de datos de la matrícula estudiantil. |
| Subproceso | - Actualización de datos del estudiante.<br>- Actualización de datos de los representantes. |
| Descripción | La Secretaria busca en un folder la ficha del estudiante, de ser necesario corrige los datos necesarios o simplemente termina de llenar la ficha y posteriormente la archiva. |
| Diagnóstico | Se pudo observar hojas con tachones y corrector. |

Fuente: Unidad Educativa Jerusalén

Documentación y diagnóstico del proceso de anulación de la matricula estudiantil.

Tabla 2.5. Proceso anulación de la matrícula

| | |
|---|---|
| Proceso | Anulación de la matrícula estudiantil. |
| Subproceso | - Anulación de la matrícula.<br>- Devolución de la carpeta al representante. |
| Descripción | La Secretaria busca en un folder la ficha del estudiante, la coloca en una carpeta con las notas y observaciones del estudiante y se la entrega al representante. |
| Diagnóstico | Se pudo observar fichas duplicadas de los estudiantes sin matricula que aún se encontraban en el folder de los estudiantes matriculados. |

Fuente: Unidad Educativa Jerusalén

El objetivo del diagnóstico fue conocer que los procesos no estaban alineados con las tecnologías de la información ya que la mayoría de información se encontraba almacenada en folders, uno por cada curso y paralelo.

Los recursos para la gestión de la información existían ya que la infraestructura tecnológica era suficiente para poder implementar un servicio de información que ayude al Personal administrativo a realizar su trabajo, sin embargo, el no tener un valor agregado significaba que dichos recursos no se podían aprovechar en su totalidad.

La gestión de los procesos en ese entonces se los realizaba de forma manual ya que no existía ninguna aplicación para gestionar la información, actividades simples como por ejemplo la consulta de información se dificultaba si el folder tenía demasiados archivos o la actualización de datos requería de llenar una nueva ficha del estudiante.

Adicionalmente se pudo conocer que existían problemas al momento de registrar los pagos de las pensiones ya que si no se registraba el pago realizado en un cuaderno entonces dicho pago no existía y el dinero se perdía, también el hecho de que no se llevaba un registro adecuado de los adelantos realizados significaba que no se tenía una evidencia de que se haya o no realizado dicho adelanto a algún trabajador.

En síntesis, el diagnóstico reveló que los procesos que se realizaban de forma manual se los hacia empíricamente y que estos se podían perfeccionar y trasladar dichos procesos a una aplicación para que todos los procesos administrativos sean eficientes.

# CAPÍTULO III

# RESULTADOS

## 3.1.Primera iteración

### 3.1.1.Especulación

#### 3.1.1.1.Obtención de requerimientos

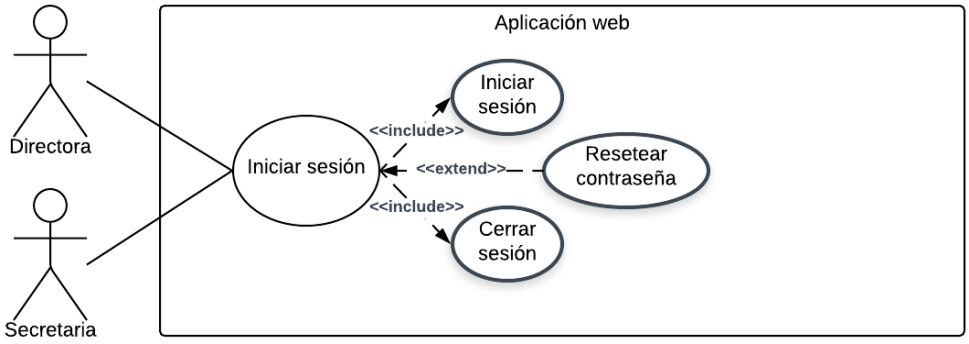
En base a la entrevista realizada a la Rectora de la Institución (referenciada como anexo 1 al final del documento) se pudo conocer cuáles eran los requerimientos del cliente, con la información recopilada fue posible construir los casos de uso y documentar el desarrollo de la aplicación utilizando la metodología mencionada.

El análisis mostró la necesidad de registrar los cobros de las matrículas y de las pensiones en un sistema informático, existió también la necesidad de registrar los adelantos de sueldo solicitados por los trabajadores para que no exista una confusión al momento de pagar el sueldo ya que según se pudo entender a través de la entrevista realizada, dichos problemas ya habían sucedido anteriormente en la Institución.

Para documentar correctamente el desarrollo de la aplicación se construyeron los casos de uso con cada una de las funciones que el sistema realiza basándose en los requerimientos obtenidos, aquello resultó en una documentación muy clara de cada módulo desarrollado para un posterior análisis y referencia.

### 3.1.1.2.Casos de uso

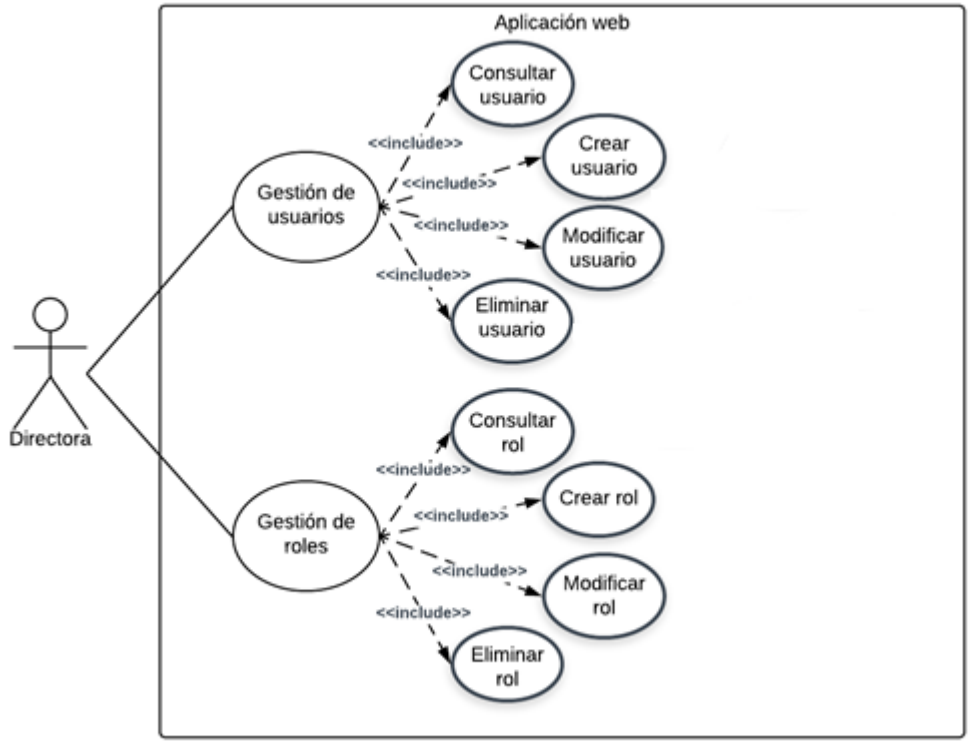
Ilustración 3.1. Diagrama de casos de uso para el inicio de sesión




Fuente: Elaboración propia

El diagrama de casos de uso del módulo de usuarios consiste en la gestión de usuarios y la gestión de roles, ambos conforman el componente de la siguiente manera:

Ilustración 3.2. Diagrama de casos de uso del módulo de usuarios




Fuente: Elaboración propia

**Caso de uso:** Iniciar sesión.

**Actores:** Usuario.

**Propósito:** Permitir al usuario iniciar sesión en la aplicación.

**Resumen:** Los usuarios registrados ingresan su dirección de correo electrónico junto con su contraseña en la pantalla de iniciar sesión para poder acceder a la aplicación.

**Curso de los eventos:**

Tabla 3.1. Caso de uso - Iniciar sesión

| Actores | Sistema |
|---|---|
| 1.- El usuario ingresa su dirección de correo electrónico y su contraseña. | 2.- El sistema valida la dirección de correo electrónico y la contraseña ingresada.<br>3.- El sistema autoriza al usuario a iniciar sesión en la aplicación.<br>4.- El sistema redirige al usuario a la pantalla principal de la aplicación. |

Fuente: Elaboración propia

**Curso alterno:**

2.1.- El sistema no encuentra la dirección de correo electrónico ingresada.

2.2.- La contraseña ingresada no es la correcta.

**Caso de uso:** Reiniciar contraseña.

**Actores:** Usuario.

**Propósito:** Permitir al usuario reiniciar su contraseña para iniciar sesión en la aplicación.

**Resumen:** Los usuarios registrados ingresan su dirección de correo electrónico para que el sistema les envíe un mensaje con un enlace para reiniciar su contraseña.

**Curso de los eventos:**

Tabla 3.2. Caso de uso - Reiniciar contraseña

| Actores | Sistema |
|---|---|
| 1.- El usuario ingresa su dirección de correo electrónico. | |
| | 2.- El sistema valida la dirección de correo electrónico ingresada. |
| | 3.- El sistema envía un mensaje con un enlace para reiniciar la contraseña. |
| 4.- El usuario ingresa su nueva contraseña. | |
| | 5.- El sistema valida la nueva contraseña. |
| | 6.- El usuario es redirigido a la pantalla principal de la aplicación. |

Fuente: Elaboración propia

**Curso alterno:**

2.1.- El sistema no encuentra la dirección de correo electrónico ingresada.

5.1.- La contraseña ingresada no es correcta.

**Caso de uso:** Cerrar sesión.

**Actores:** Usuario.

**Propósito:** Permitir al usuario cerrar su sesión en la aplicación.

**Resumen:** Los usuarios conectados a la aplicación pueden cerrar su sesión y guardar su trabajo al final de su jornada.

**Curso de los eventos:**

Tabla 3.3. Caso de uso – Cerrar sesión

| Actores | Sistema |
|---|---|
| 1.- El usuario presiona el botón de cerrar sesión. | 2.- El sistema cierra la sesión del usuario conectado.<br>3.- El sistema redirige al usuario a la pantalla de iniciar sesión. |

Fuente: Elaboración propia

**Curso alterno:**

1.1.- La sesión del usuario ha expirado.

**Caso de uso:** Consultar usuario.

**Actores:** Directora.

**Propósito:** Permitir a la Directora consultar la base de datos de la aplicación en busca de un usuario en específico.

**Resumen:** La Directora ingresa un término de búsqueda para que el sistema filtre la base de datos y devuelva un resultado de la búsqueda.

**Curso de los eventos:**

Tabla 3.4. Caso de uso - Consultar usuario

| Actores | Sistema |
|---|---|
| 1.- La Directora ingresa un término en el campo de búsqueda de la pantalla de gestión de usuarios de la aplicación. | 2.- El sistema filtra la base de datos en búsqueda de un resultado que coincida con el término de búsqueda ingresado.<br>3.- El sistema devuelve los resultados de la consulta. |

Fuente: Elaboración propia

**Curso alterno:**

2.1.- No se encontraron resultados.

**Caso de uso:** Crear usuario.

**Actores:** Directora.

**Propósito:** Permitir a la Directora crear un nuevo usuario de la aplicación.

**Resumen:** La Directora ingresa los datos del nuevo usuario para que el sistema los valide y pueda crear un nuevo usuario.

**Curso de los eventos:**

Tabla 3.5. Caso de uso - Crear usuario

| Actores | Sistema |
|---|---|
| 1.- La Directora presiona el botón para crear un nuevo usuario. | |
| | 2.- El sistema redirige a la Directora a un formulario para crear un nuevo usuario. |
| 3.- La Directora ingresa los datos del nuevo usuario. | |
| | 4.- El sistema valida los datos ingresados. |
| | 5.- El sistema crea al nuevo usuario. |

Fuente: Elaboración propia

**Curso alterno:**

4.1.- Los datos ingresados no son válidos.

4.2.- Ya existe un usuario con esa dirección de correo electrónico.

**Caso de uso:** Modificar usuario.

**Actores:** Directora.

**Propósito:** Permitir a la Directora modificar los datos de un usuario de la aplicación.

**Resumen:** La Directora selecciona a un usuario de la aplicación y modifica sus datos, el sistema los valida y los actualiza.

**Curso de los eventos:**

Tabla 3.6. Caso de uso - Modificar usuario

| Actores | Sistema |
|---|---|
| 1.- La Directora presiona el botón para modificar usuario. | |
| | 2.- El sistema redirige a la Directora a un formulario para modificar al usuario seleccionado. |
| 3.- La Directora ingresa los nuevos datos del usuario seleccionado. | |
| | 4.- El sistema valida los datos ingresados. |
| | 5.- El sistema actualiza los datos del usuario seleccionado. |

Fuente: Elaboración propia

**Curso alterno:**

4.1.- Los datos ingresados no son válidos.

4.2.- Ya existe un usuario con esa dirección de correo electrónico.

**Caso de uso:** Eliminar usuario.

**Actores:** Directora.

**Propósito:** Permitir a la Directora eliminar a un usuario de la aplicación.

**Resumen:** La Directora selecciona a un usuario de la aplicación y lo elimina.

**Curso de los eventos:**

Tabla 3.7. Caso de uso - Eliminar usuario

| Actores | Sistema |
|---|---|
| 1.- La Directora presiona el botón para eliminar usuario. | |
| | 2.- El sistema presenta a la Directora un mensaje de confirmación. |
| 3.- La Directora confirma la acción realizada. | |
| | 4.- El sistema elimina al usuario seleccionado. |
| | 5.- El sistema actualiza la tabla de datos de la pantalla de gestión de usuarios de la aplicación. |

Fuente: Elaboración propia

**Curso alterno:**

3.1.- La Directora no confirma la acción realizada.

4.1.- El sistema no elimina al usuario seleccionado.

**Caso de uso:** Consultar rol.

**Actores:** Directora.

**Propósito:** Permitir a la Directora consultar la base de datos de la aplicación en busca de un rol en específico.

**Resumen:** La Directora ingresa un término de búsqueda para que el sistema filtre la base de datos y devuelva un resultado de la búsqueda.

**Curso de los eventos:**

Tabla 3.8. Caso de uso - Consultar rol

| Actores | Sistema |
|---|---|
| 1.- La Directora ingresa un término en el campo de búsqueda de la pantalla de gestión de roles de la aplicación. | 2.- El sistema filtra la base de datos en búsqueda de un resultado que coincida con el término de búsqueda ingresado.<br>3.- El sistema devuelve los resultados de la consulta. |

Fuente: Elaboración propia

**Curso alterno:**

2.1.- No se encontraron resultados.

**Caso de uso:** Crear rol.

**Actores:** Directora.

**Propósito:** Permitir a la Directora crear un nuevo rol para los usuarios de la aplicación.

**Resumen:** La Directora ingresa los datos del nuevo rol para que el sistema los valide y pueda crear un nuevo rol.

**Curso de los eventos:**

Tabla 3.9. Caso de uso - Crear rol

| Actores | Sistema |
|---|---|
| 1.- La Directora presiona el botón para crear un nuevo rol. | |
| | 2.- El sistema redirige a la Directora a un formulario para crear un nuevo rol. |
| 3.- La Directora ingresa los datos del nuevo rol. | |
| | 4.- El sistema valida los datos ingresados. |
| | 5.- El sistema crea el nuevo rol. |

Fuente: Elaboración propia

**Curso alterno:**

4.1.- Los datos ingresados no son válidos.

4.2.- Ya existe un rol con ese nombre.

**Caso de uso:** Modificar rol.

**Actores:** Directora.

**Propósito:** Permitir a la Directora modificar los datos de un rol existente.

**Resumen:** La Directora selecciona un rol existente y modifica sus datos, el sistema los valida y los actualiza.

**Curso de los eventos:**

Tabla 3.10. Caso de uso - Modificar rol

| Actores | Sistema |
|---|---|
| 1.- La Directora presiona el botón para modificar rol. | |
| | 2.- El sistema redirige a la Directora a un formulario para modificar el rol seleccionado. |
| 3.- La Directora ingresa los nuevos datos del rol seleccionado. | |
| | 4.- El sistema valida los datos ingresados. |
| | 5.- El sistema actualiza los datos del rol seleccionado. |

Fuente: Elaboración propia

**Curso alterno:**

4.1.- Los datos ingresados no son válidos.

4.2.- Ya existe un rol con ese nombre.

**Caso de uso:** Eliminar rol.

**Actores:** Directora.

**Propósito:** Permitir a la Directora eliminar un rol.

**Resumen:** La Directora selecciona un rol y lo elimina.

**Curso de los eventos:**

Tabla 3.11. Caso de uso - Eliminar rol

| Actores | Sistema |
|---|---|
| 1.- La Directora presiona el botón para eliminar rol. | |
| | 2.- El sistema presenta a la Directora un mensaje de confirmación. |
| 3.- La Directora confirma la acción realizada. | |
| | 4.- El sistema elimina el rol seleccionado. |
| | 5.- El sistema actualiza la tabla de datos de la pantalla de gestión de roles de la aplicación. |

Fuente: Elaboración propia

**Curso alterno:**

3.1.- La Directora no confirma la acción realizada.

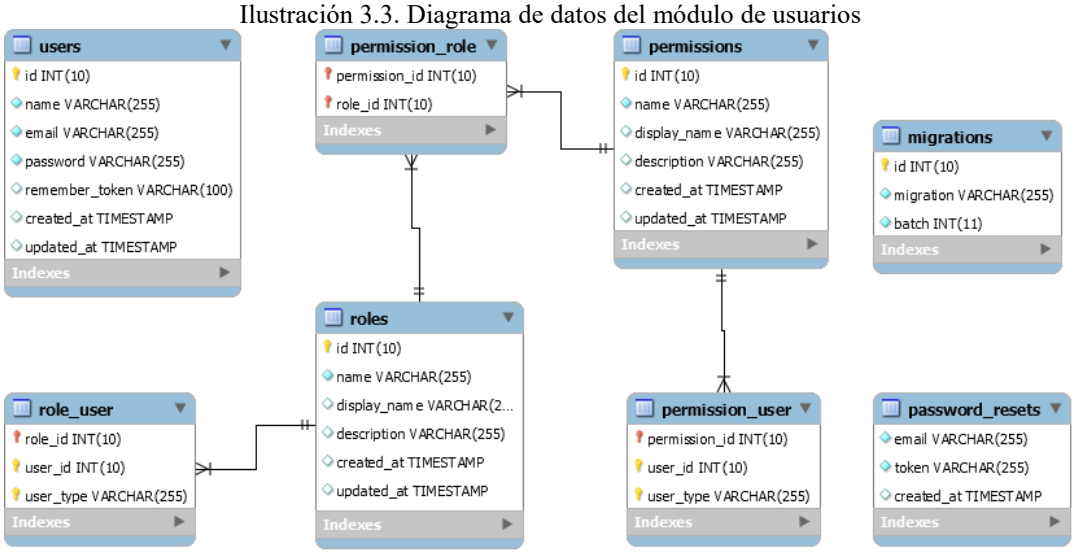
4.1.- El sistema no elimina el rol seleccionado.

### 3.1.1.3.Diagrama de datos

Para el desarrollo del módulo de usuarios se utilizaron un numero de ocho (8) tablas que almacenan los siguientes datos: la tabla *permissions* almacena los permisos que tiene la aplicación, la tabla *roles* almacena los roles que tienen los usuarios y la tabla *users* almacena los usuarios y su información, existen también tres tablas intermedias encargadas de almacenar los permisos que tiene cada rol, los permisos individuales que tiene cada usuario y finalmente los roles que puede tener cada usuario.

Adicionalmente existen dos tablas, la primera almacena las credenciales de los usuarios cuando han solicitado reiniciar su contraseña y la segunda almacena un registro de todas las migraciones de las tablas realizadas en la base de datos de la aplicación, el diagrama de datos del módulo respectivo se muestra a continuación:

Ilustración 3.3. Diagrama de datos del módulo de usuarios




Fuente: Elaboración propia

#### 3.1.1.4.Diccionario de datos

Descripción y estructura de la tabla *permissions*.

Tabla 3.12. Tabla permissions

| Nombre | Tipo | Razón |
|---|---|---|
| id | int(10) | id auto incremental y clave primaria de la tabla |
| name | varchar(255) | nombre único e identificativo del permiso |
| display_name | varchar(255) | nombre que se va a mostrar en el backend |
| description | varchar(255) | descripción corta del permiso |
| created_at | timestamp | fecha de creación del registro |
| updated_at | timestamp | fecha de actualización del registro |

Fuente: Elaboración propia

Descripción y estructura de la tabla *roles*.

Tabla 3.13. Tabla roles

| Nombre | Tipo | Razón |
|---|---|---|
| id | int(10) | id auto incremental y clave primaria de la tabla |
| name | varchar(255) | nombre único e identificativo del rol |
| display_name | varchar(255) | nombre que se va a mostrar en el backend |
| description | varchar(255) | descripción corta del rol |
| created_at | timestamp | fecha de creación del registro |
| updated_at | timestamp | fecha de actualización del registro |

Fuente: Elaboración propia

Descripción y estructura de la tabla users.

Tabla 3.14. Tabla users

| Nombre | Tipo | Razón |
|---|---|---|
| id | int(10) | id auto incremental y clave primaria de la tabla |
| name | varchar(255) | nombre completo del usuario |
| email | varchar(255) | correo electrónico del usuario |
| password | varchar(255) | contraseña única del usuario |
| remember_token | varchar(255) | token que guarda el navegador |
| created_at | timestamp | fecha de creación del registro |
| updated_at | timestamp | fecha de actualización del registro |

Fuente: Elaboración propia

Descripción y estructura de la tabla *migrations*.

Tabla 3.15. Tabla migrations

| Nombre | Tipo | Razón |
|---|---|---|
| id | int(10) | id auto incremental y clave primaria de la tabla |
| migration | varchar(255) | nombre de la migración de la tabla creada |
| batch | int(11) | lote de la migración realizada |

Fuente: Elaboración propia

Descripción y estructura de la tabla *permission_user*.

Tabla 3.16. Tabla permission_user

| Nombre | Tipo | Razón |
|---|---|---|
| permission_id | int(10) | clave principal y foránea de la tabla permissions |
| user_id | int(10) | clave principal y foránea de la tabla users |
| user_type | varchar(255) | tipo de usuario que tiene el permiso asociado |

Fuente: Elaboración propia

Descripción y estructura de la tabla *permission_role*.

Tabla 3.17. Tabla permission_role

| Nombre | Tipo | Razón |
|---|---|---|
| permission_id | int(10) | clave principal y foránea de la tabla permissions |
| role_id | int(10) | clave principal y foránea de la tabla roles |

Fuente: Elaboración propia

Descripción y estructura de la tabla *role_user*.

Tabla 3.18. Tabla role_user

| Nombre | Tipo | Razón |
|---|---|---|
| role_id | int(10) | clave principal y foránea de la tabla roles |
| user_id | int(10) | clave principal y foránea de la tabla users |
| user_type | varchar(255) | tipo de usuario que tiene el permiso asociado |

Fuente: Elaboración propia

Descripción y estructura de la tabla *password resets*.

Tabla 3.19. Tabla password resets

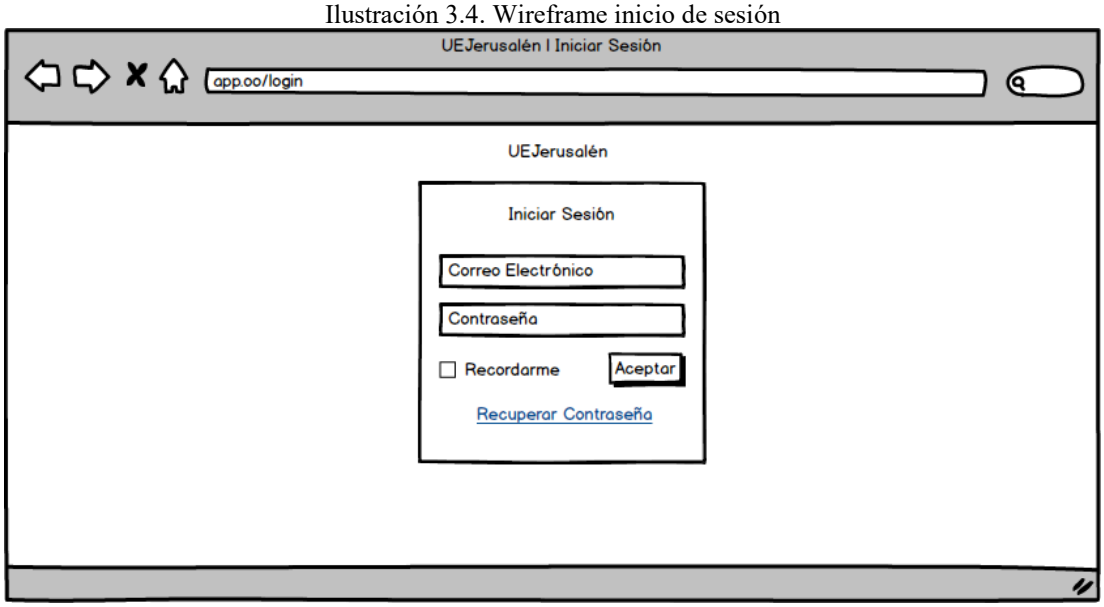
| Nombre | Tipo | Razón |
|---|---|---|
| email | varchar(255) | email del usuario para recuperar contraseña |
| token | varchar(255) | token único autogenerado por la aplicación |
| created_at | timestamp | fecha de creación del registro |

Fuente: Elaboración propia

### 3.1.1.5.Diseño

Una vez que los casos de uso del módulo de usuarios se establecieron y se construyeron las tablas de la base de datos del mismo módulo, se procedió a realizar la siguiente actividad que consistió en el diseño de la interfaz de usuario, esta contiene el panel de administración, pantallas para la gestión de usuarios y roles y el inicio de sesión de la aplicación con la opción de poder recuperar la contraseña por medio de un mensaje.

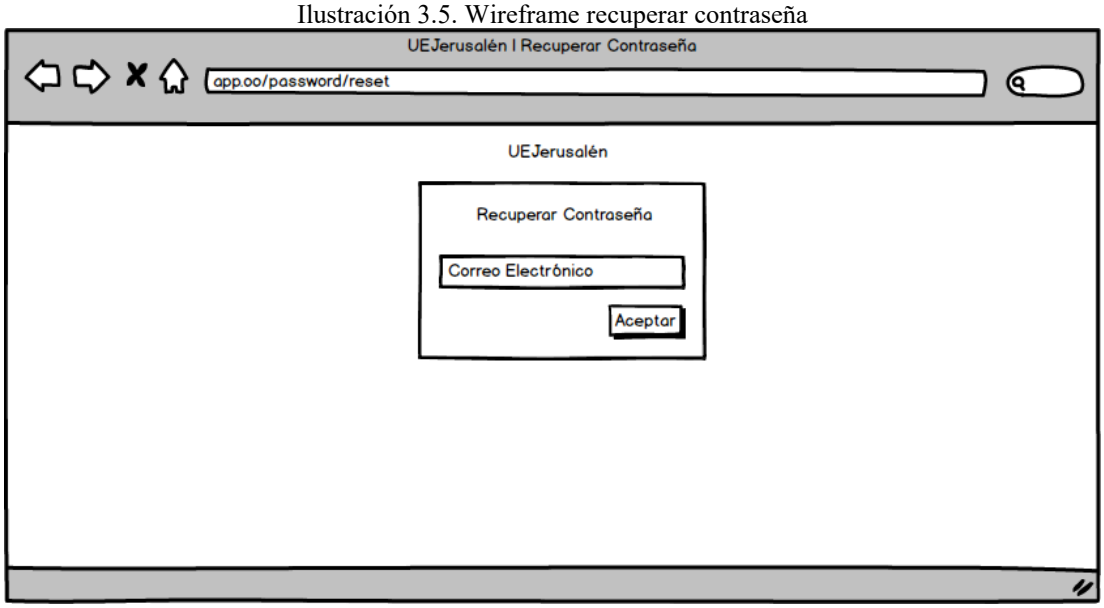
Ilustración 3.4. Wireframe inicio de sesión




Fuente: Elaboración propia

Diseño de la interfaz de usuario para poder recuperar la contraseña de inicio de sesión.

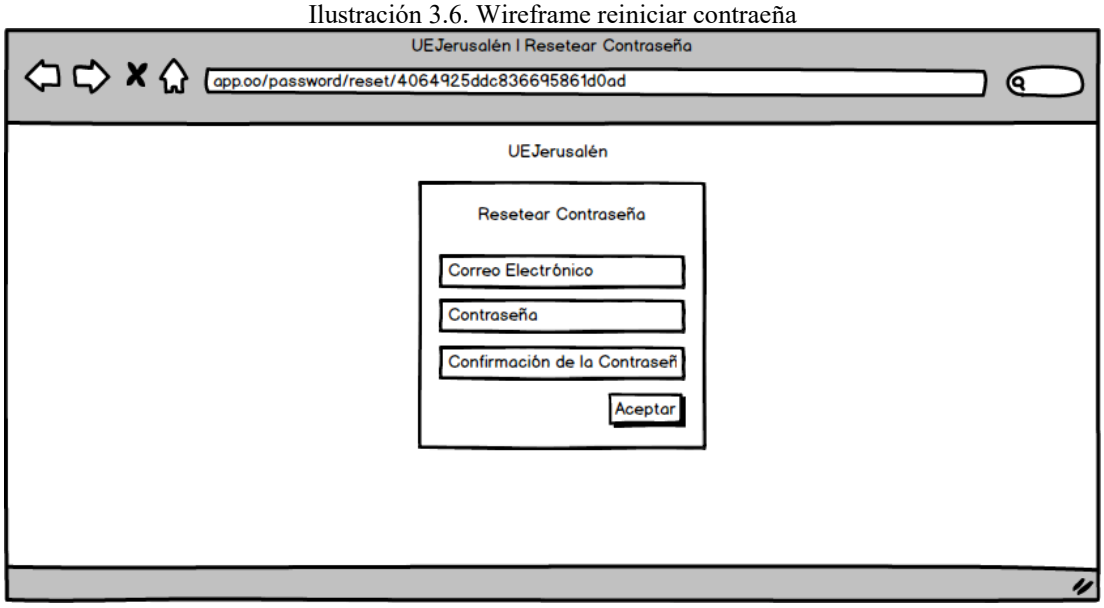
Ilustración 3.5. Wireframe recuperar contraseña




Fuente: Elaboración propia

Diseño de la interfaz de usuario para ingresar la nueva contraseña de inicio de sesión.

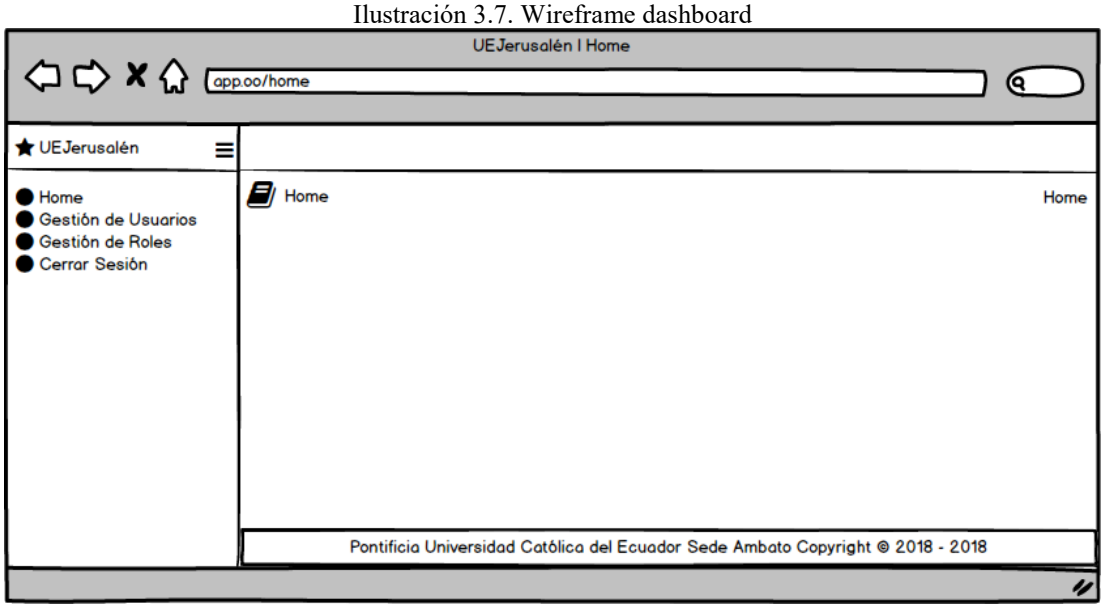
Ilustración 3.6. Wireframe reiniciar contraeña




Fuente: Elaboración propia

Diseño de la interfaz de usuario para redirigir a los usuarios que hayan iniciado sesión.

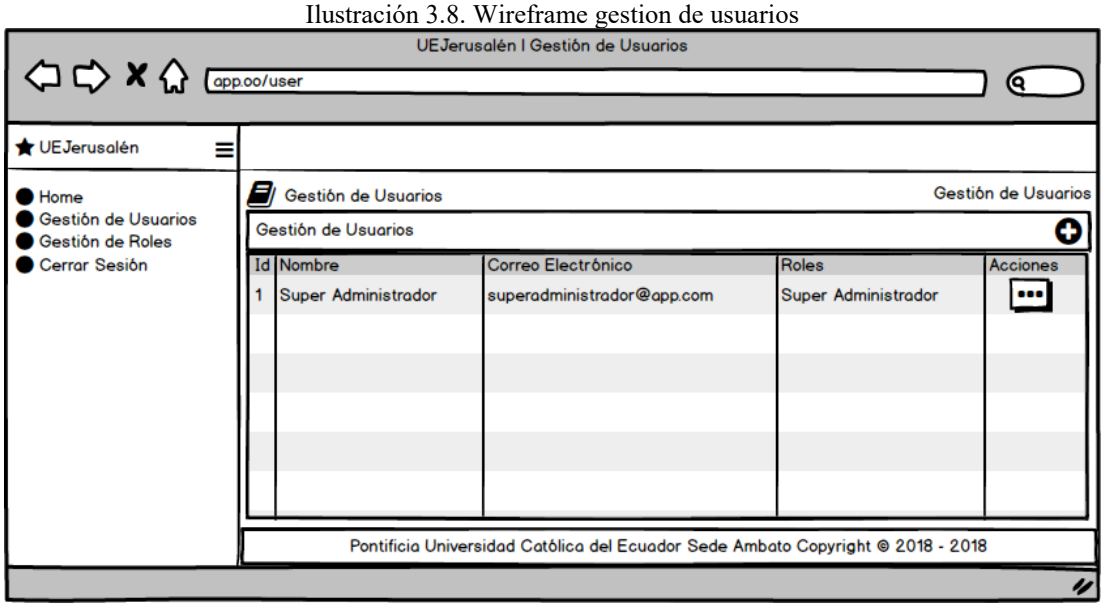
Ilustración 3.7. Wireframe dashboard




Fuente: Elaboración propia

Diseño de la interfaz de usuario para la gestión de usuarios del módulo de usuarios.

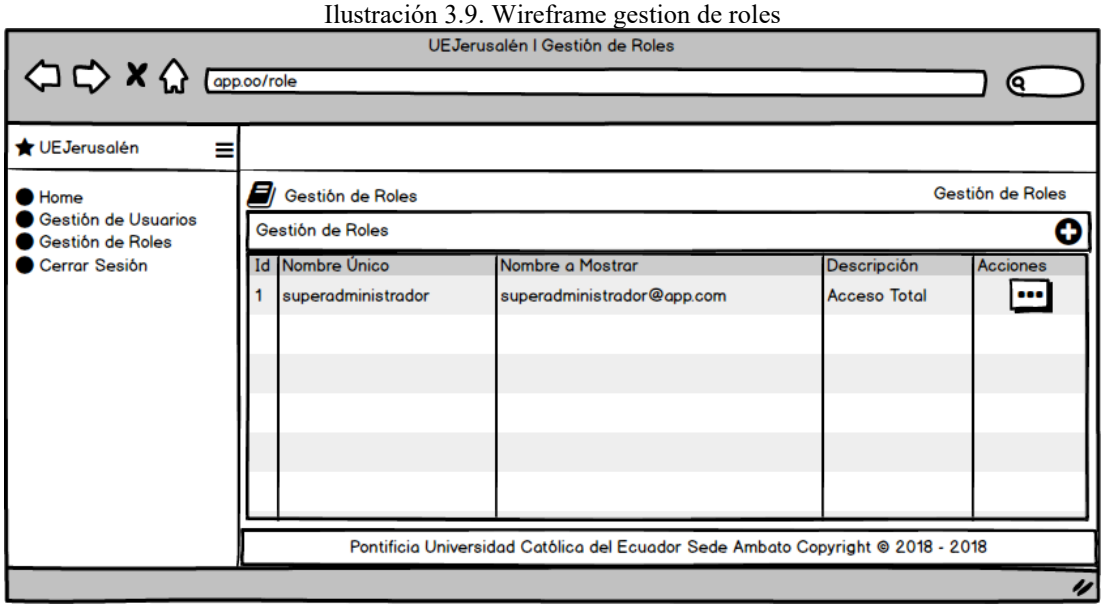
Ilustración 3.8. Wireframe gestion de usuarios




Fuente: Elaboración propia

Diseño de la interfaz de usuario para la gestión de roles del módulo de usuarios.

Ilustración 3.9. Wireframe gestion de roles




Fuente: Elaboración propia

### 3.1.2.Colaboración

#### 3.1.2.1.Programación

En esta iteración se empieza a desarrollar el módulo de usuarios de la aplicación, de acuerdo con los requerimientos del cliente se necesita controlar el acceso de los usuarios por medio de roles y permisos, por lo cual se implementaron dos modelos, el primero para almacenar los roles de la aplicación y el segundo para almacenar los permisos que puede tener cada rol, para gestionar los permisos de manera adecuada se añadieron permisos a cada función desarrollada, por ejemplo la gestión de usuarios tiene cuatro (4) permisos los mismos que permiten acceder a la gestión de usuarios, crear un usuario, modificar un usuario o a su vez eliminar un usuario.

El permiso para ver los usuarios se lo coloca en el archivo *main-sidebar.blade.php* así:

Ilustración 3.10. Permiso para ver los usuarios

```
@permission('read-users')
    <li class="{{ active_class(Active::checkUriPattern('user*')) }}">
        <a href="{{ route('user.index') }}">
            <i class="fa fa-circle"></i> <span>Gestión de Usuarios</span>
        </a>
    </li>
@endpermission
```

Fuente: Elaboración propia

El permiso para poder crear un usuario se lo coloca en el archivo *index.blade.php*, así:

Ilustración 3.11. Permiso para crear un usuario

```
<div class="box-header with-border">
    <h3 class="box-title">Gestión de Usuarios</h3>
    <div class="box-tools pull-right">
        @permission('create-users')
            <a href="{{ route('user.create') }}" data-toggle="tooltip" title="Crear Usuario" class="btn btn-box-tool">
                <i class="fa fa-2x fa-plus-circle text-primary"></i>
            </a>
        @endpermission
    </div>
</div>
```

Fuente: Elaboración propia

Mientras que el resto de los permisos se colocan en el archivo *User.php* de esta forma:

Ilustración 3.12. Permisos para modificar y eliminar un usuario

```
protected function getEditButtonAttribute()
{
    if (Auth::user()->hasPermission( permission: 'update-users')) {
        if ($this->id != 1) {
            return '<li><a href="' . route( name: 'user.edit', $this->id) . '">Editar Usuario</a></li>';
        }

        return '';
    }

    return '';
}

public function getDeleteButtonAttribute()
{
    if (Auth::user()->hasPermission( permission: 'delete-users')) {
        if ($this->id != Auth::user()->id && $this->id != 1) {
            return '<li><a href="javascript:void(0);" class="btn-delete" id="' . $this->id . '">Eliminar Usuario</a></li>';
        }

        return '';
    }

    return '';
}
```

Fuente: Elaboración propia

La dinámica es la misma para cada modelo de la aplicación, se colocan los permisos necesarios en los archivos correspondientes al modelo que se vaya a gestionar, de esta forma al crear un rol se otorgan los permisos necesarios para dicho rol.

En el archivo *UserController.php* se encuentra el método para crear un usuario, para que la aplicación pueda validar correctamente los datos ingresados se deben aplicar reglas las mismas que se encargan de controlar dicha información, en el archivo mencionado anteriormente el método para crear un usuario es el siguiente:

Ilustración 3.13. Método para crear un usuario

```php
public function store(Request $request)
{
    $this->validate($request, [
        'name' => 'required|string|max:255',
        'email' => 'required|string|email|max:255|unique:users',
        'password' => 'required|string|min:6|confirmed',
        'password_confirmation' => 'required|string|min:6',
        'roles' => 'required'
    ]);

    $user = (new User)->create([
        'name' => $request['name'],
        'email' => $request['email'],
        'password' => bcrypt($request['password']),
        'password_confirmation' => bcrypt($request['password_confirmation'])
    ]);

    $user->roles()->attach($request['roles']);

    flash( message: 'Usuario Creado.')->success();

    return redirect()->route( route: 'user.index');
}
```

Fuente: Elaboración propia

Por motivo de seguridad se debe encriptar la contraseña ingresada y también confirmarla con un segundo campo en donde el usuario ingresa la misma contraseña, el método para modificar un usuario es similar como se muestra a continuación:

Ilustración 3.14. Método para modificar un usuario

```php
public function update(Request $request, User $user)
{
    $this->validate($request, [
        'name' => 'required|string|max:255',
        'email' => 'required|string|email|max:255|unique:users,email,' . $user->id,
        'roles' => 'required'
    ]);

    $user->update([
        'name' => $request['name'],
        'email' => $request['email']
    ]);

    $user->syncRoles($request['roles']);

    flash( message: 'Usuario Actualizado.')->success();

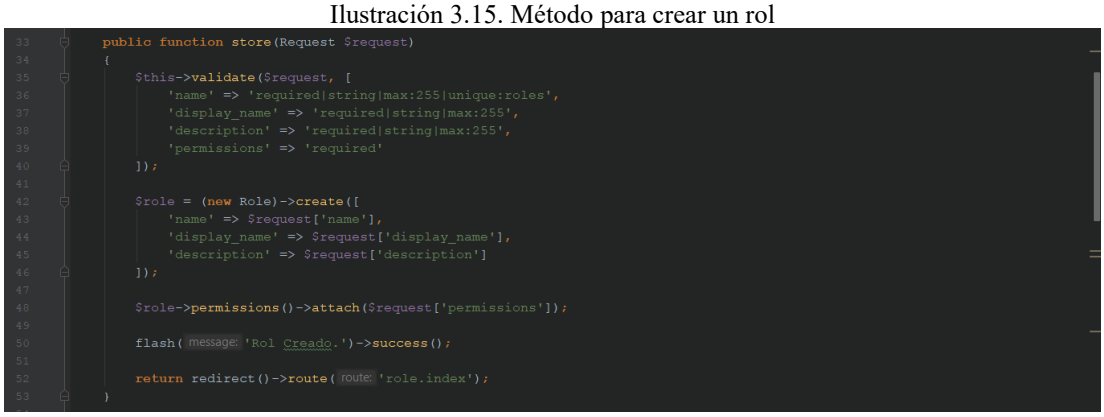
    return redirect()->route( route: 'user.index');
}
```

Fuente: Elaboración propia

La única diferencia entre ambos métodos es que al modificar un recurso se añade el modelo de este al constructor de la función para encontrar el recurso automáticamente.

En el archivo *RoleController.php* el método para crear un rol es el siguiente:

Ilustración 3.15. Método para crear un rol



```
public function store(Request $request)
{
    $this->validate($request, [
        'name' => 'required|string|max:255|unique:roles',
        'display_name' => 'required|string|max:255',
        'description' => 'required|string|max:255',
        'permissions' => 'required'
    ]);

    $role = (new Role)->create([
        'name' => $request['name'],
        'display_name' => $request['display_name'],
        'description' => $request['description']
    ]);

    $role->permissions()->attach($request['permissions']);

    flash( message: 'Rol Creado.')->success();

    return redirect()->route( route: 'role.index');
}
```

Fuente: Elaboración propia

En el mismo archivo se encuentra el método para modificar un rol, dicha función permite seleccionar un rol en especial y editarlo según sea necesario ya sea al añadir nuevos permisos o remover permisos previamente asignados al rol, modificar su nombre o su descripción, el método especificado es el siguiente:

Ilustración 3.16. Método para modificar un rol



```
public function update(Request $request, Role $role)
{
    $this->validate($request, [
        'name' => 'required|string|max:255|unique:roles,name,' . $role->id,
        'display_name' => 'required|string|max:255',
        'description' => 'required|string|max:255',
        'permissions' => 'required'
    ]);

    $role->update([
        'name' => $request['name'],
        'display_name' => $request['display_name'],
        'description' => $request['description']
    ]);

    $role->syncPermissions($request['permissions']);

    flash( message: 'Rol Actualizado.')->success();

    return redirect()->route( route: 'role.index');
}
```

Fuente: Elaboración propia

En esta iteración no fue necesario realizar una reunión de colaboración con el cliente ya que durante el desarrollo del componente no existieron dudas ni observaciones, es válido también aclarar el hecho de que ejecutar tal actividad es opcional para el desarrollo del proyecto.

### 3.1.3.Aprendizaje

#### 3.1.3.1.Control de calidad

Para realizar el control de calidad del módulo desarrollado se programó una reunión con la Directora de la Institución para probar la aplicación, el respectivo control de calidad se documentó con la siguiente prueba de aceptación la misma que contiene todos los requerimientos obtenidos del módulo y las respuestas del usuario.

Tabla 3.20. Prueba de aceptación de módulo de usuarios

| Prueba de aceptación del módulo de usuarios | | |
|---|---|---|
| Título: Revisión del módulo de usuarios | | |
| Fecha: 03-05-2018 | | |
| Usuario: Fanny Hernández | | |
| Requerimiento | Si | No |
| Puede crear, modificar y eliminar usuarios | X | |
| Puede asignar roles a los usuarios | X | |
| Puede remover y reasignar roles a los usuarios | X | |
| Puede crear, modificar y eliminar roles | X | |
| Puede asignar permisos a los roles | X | |
| Puede remover y reasignar permisos a los roles | X | |
| Puede controlar el acceso a la aplicación | X | |
| Puede realizar consultas de los usuarios y los roles | X | |
| Puede navegar de manera intuitiva por la aplicación | X | |

Fuente: Elaboración propia

La actividad de implementación se realiza en la última iteración del proyecto ya que es considerada una mala práctica de programación implementar un proyecto incompleto, además de que se malgastan recursos al realizar varias implementaciones.

## 3.2.Segunda iteración

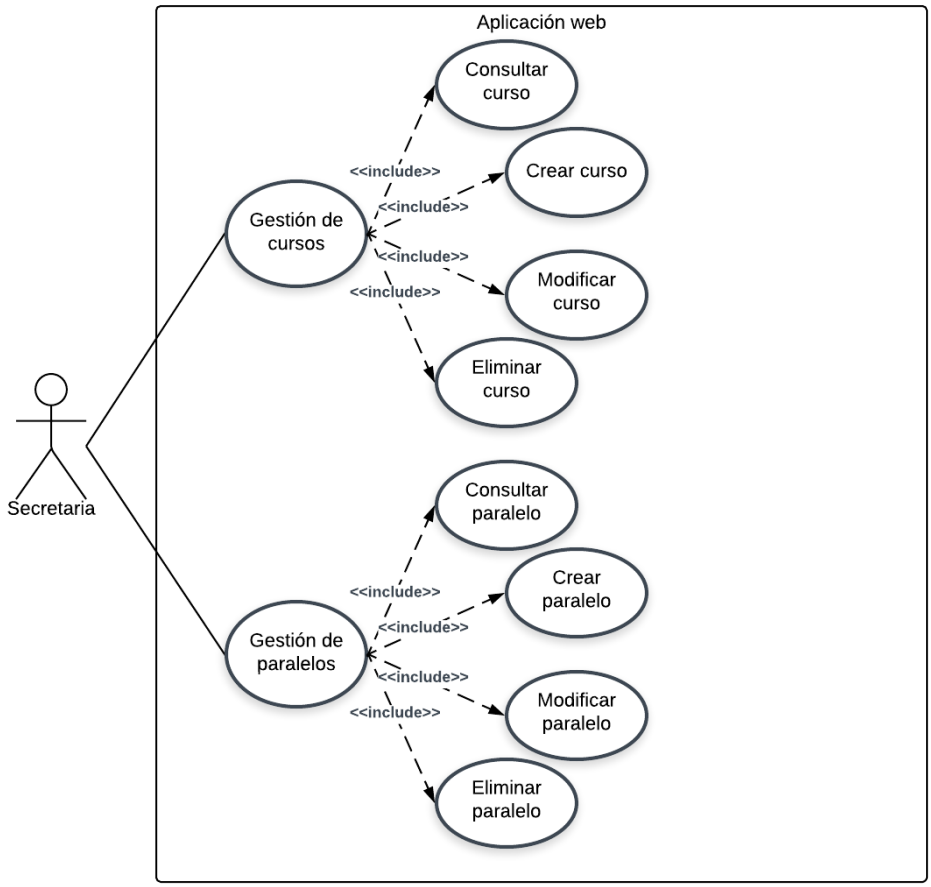
Después de la primera iteración y de la obtención inicial de requerimientos se cree que no es necesario realizar dicha actividad una segunda vez ya que estos son los mismos.

### 3.2.1.Especulación

#### 3.2.1.1.Casos de uso

A continuación, la documentación de los casos de uso del componente de matrículas.

Ilustración 3.17. Diagrama de casos de uso matrículas (cursos y paralelos)




Fuente: Elaboración propia

**Caso de uso:** Consultar curso.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria consultar la base de datos de la aplicación en busca de un curso en específico.

**Resumen:** La Secretaria ingresa un término de búsqueda para que el sistema filtre la base de datos y devuelva un resultado de la búsqueda.

**Curso de los eventos:**

Tabla 3.21. Caso de uso - Consultar curso

| Actores | Sistema |
|---|---|
| 1.- La Secretaria ingresa un término en el campo de búsqueda de la pantalla de gestión de cursos de la aplicación. | 2.- El sistema filtra la base de datos en búsqueda de un resultado que coincida con el término de búsqueda ingresado.<br>3.- El sistema devuelve los resultados de la consulta. |

Fuente: Elaboración propia

**Curso alterno:**

2.1.- No se encontraron resultados.

**Caso de uso:** Crear curso.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria crear un nuevo curso.

**Resumen:** La Secretaria ingresa los datos del nuevo curso para que el sistema los valide y pueda crear un nuevo curso.

**Curso de los eventos:**

Tabla 3.22. Caso de uso - Crear curso

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para crear un nuevo curso. | |
| | 2.- El sistema redirige a la Secretaria a un formulario para crear un nuevo curso. |
| 3.- La Secretaria ingresa los datos del nuevo curso. | |
| | 4.- El sistema valida los datos ingresados. |
| | 5.- El sistema crea al nuevo curso. |

Fuente: Elaboración propia

**Curso alterno:**

4.1.- Los datos ingresados no son válidos.

**Caso de uso:** Modificar curso.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria modificar los datos de un curso.

**Resumen:** La Secretaria selecciona un curso y modifica sus datos, el sistema los valida y los actualiza.

**Curso de los eventos:**

Tabla 3.23. Caso de uso - Modificar curso

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para modificar curso. | |
| | 2.- El sistema redirige a la Secretaria a un formulario para modificar al curso seleccionado. |
| 3.- La Secretaria ingresa los nuevos datos del curso seleccionado. | |
| | 4.- El sistema valida los datos ingresados. |
| | 5.- El sistema actualiza los datos del curso seleccionado. |

Fuente: Elaboración propia

**Curso alterno:**

4.1.- Los datos ingresados no son válidos.

**Caso de uso:** Eliminar curso.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria eliminar un curso.

**Resumen:** La Secretaria selecciona un curso y lo elimina.

**Curso de los eventos:**

Tabla 3.24. Caso de uso - Eliminar curso

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para eliminar curso. | |
| | 2.- El sistema presenta a la Secretaria un mensaje de confirmación. |
| 3.- La Secretaria confirma la acción realizada. | |
| | 4.- El sistema elimina el curso seleccionado. |
| | 5.- El sistema actualiza la tabla de datos de la pantalla de gestión de cursos de la aplicación. |

Fuente: Elaboración propia

**Curso alterno:**

3.1.- La Secretaria no confirma la acción realizada.

4.1.- El sistema no elimina el curso seleccionado.

**Caso de uso:** Consultar paralelo.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria consultar la base de datos de la aplicación en busca de un paralelo en específico.

**Resumen:** La Secretaria ingresa un término de búsqueda para que el sistema filtre la base de datos y devuelva un resultado de la búsqueda.

**Curso de los eventos:**

Tabla 3.25. Caso de uso - Consultar paralelo

| Actores | Sistema |
|---|---|
| 1.- La Secretaria ingresa un término en el campo de búsqueda de la pantalla de gestión de paralelos de la aplicación. | 2.- El sistema filtra la base de datos en búsqueda de un resultado que coincida con el término de búsqueda ingresado.<br>3.- El sistema devuelve los resultados de la consulta. |

Fuente: Elaboración propia

**Curso alterno:**

2.1.- No se encontraron resultados.

**Caso de uso:** Crear paralelo.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria crear un nuevo paralelo.

**Resumen:** La Secretaria ingresa los datos del nuevo paralelo para que el sistema los valide y pueda crear un nuevo paralelo.

**Curso de los eventos:**

Tabla 3.26. Caso de uso - Crear paralelo

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para crear un nuevo paralelo. | |
| | 2.- El sistema redirige a la Secretaria a un formulario para crear un nuevo paralelo. |
| 3.- La Secretaria ingresa los datos del nuevo paralelo. | |
| | 4.- El sistema valida los datos ingresados. |
| | 5.- El sistema crea al nuevo paralelo. |

Fuente: Elaboración propia

**Curso alterno:**

4.1.- Los datos ingresados no son válidos.

**Caso de uso:** Modificar paralelo.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria modificar los datos de un paralelo.

**Resumen:** La Secretaria selecciona un paralelo y modifica sus datos, el sistema los valida y los actualiza.

**Curso de los eventos:**

Tabla 3.27. Caso de uso - Modificar paralelo

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para modificar paralelo. | |
| | 2.- El sistema redirige a la Secretaria a un formulario para modificar al paralelo seleccionado. |
| 3.- La Secretaria ingresa los nuevos datos del paralelo seleccionado. | |
| | 4.- El sistema valida los datos ingresados. |
| | 5.- El sistema actualiza los datos del paralelo seleccionado. |

Fuente: Elaboración propia

**Curso alterno:**

4.1.- Los datos ingresados no son válidos.

**Caso de uso:** Eliminar paralelo.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria eliminar un paralelo.

**Resumen:** La Secretaria selecciona un paralelo y lo elimina.

**Curso de los eventos:**

Tabla 3.28. Caso de uso - Eliminar paralelo

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para eliminar paralelo. | |
| | 2.- El sistema presenta a la Secretaria un mensaje de confirmación. |
| 3.- La Secretaria confirma la acción realizada. | |
| | 4.- El sistema elimina el paralelo seleccionado. |
| | 5.- El sistema actualiza la tabla de datos de la pantalla de gestión de paralelos de la aplicación. |

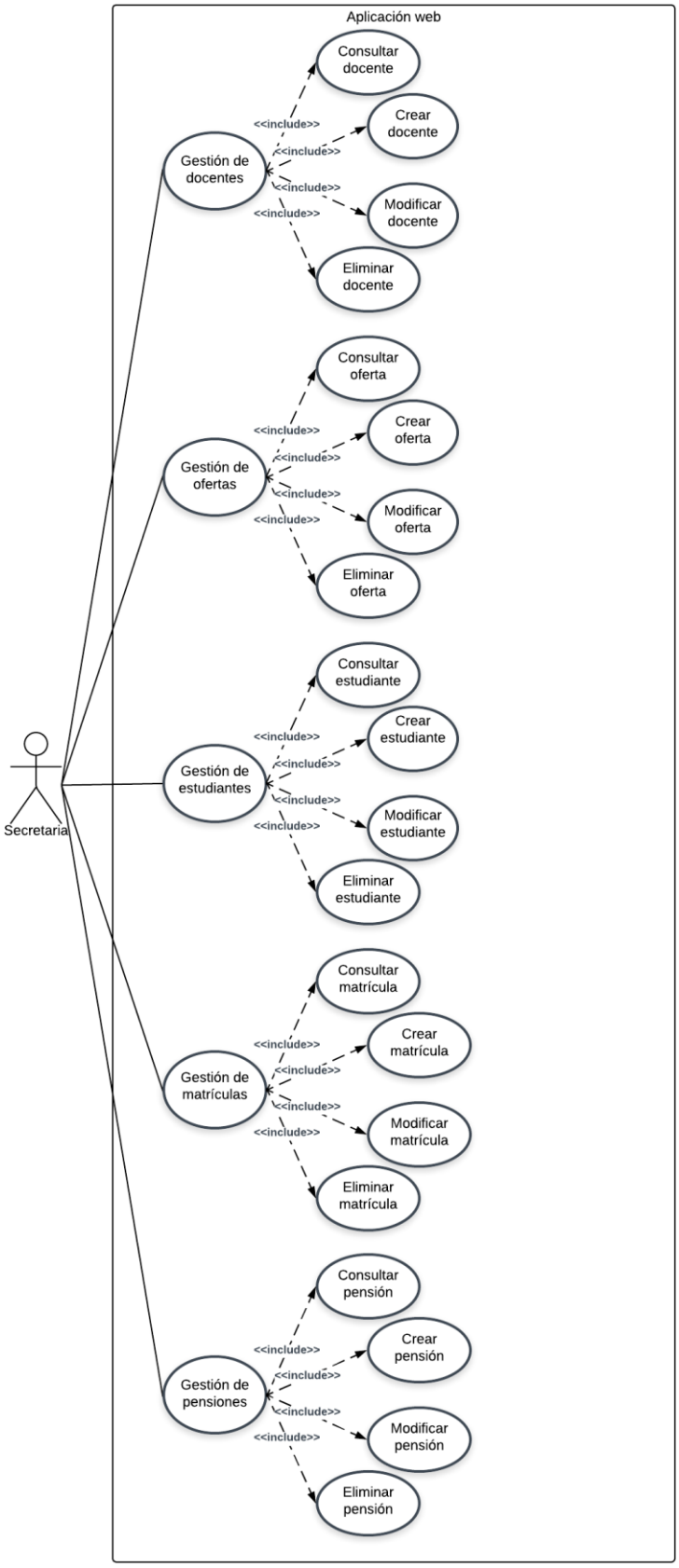
Fuente: Elaboración propia

**Curso alterno:**

3.1.- La Secretaria no confirma la acción realizada.

4.1.- El sistema no elimina el paralelo seleccionado.

Ilustración 3.18. Diagrama de casos de uso del módulo de matrículas




Fuente: Elaboración propia

**Caso de uso:** Consultar docente.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria consultar la base de datos de la aplicación en busca de un docente en específico.

**Resumen:** La Secretaria ingresa un término de búsqueda para que el sistema filtre la base de datos y devuelva un resultado de la búsqueda.

**Curso de los eventos:**

Tabla 3.29. Caso de uso - Consultar docente

| Actores | Sistema |
|---|---|
| 1.- La Secretaria ingresa un término en el campo de búsqueda de la pantalla de gestión de docentes de la aplicación. | 2.- El sistema filtra la base de datos en búsqueda de un resultado que coincida con el término de búsqueda ingresado.<br>3.- El sistema devuelve los resultados de la consulta. |

Fuente: Elaboración propia

**Curso alterno:**

2.1.- No se encontraron resultados.

**Caso de uso:** Crear docente.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria crear un nuevo docente.

**Resumen:** La Secretaria ingresa los datos del nuevo docente para que el sistema los valide y pueda crear un nuevo docente.

**Curso de los eventos:**

Tabla 3.30. Caso de uso - Crear docente

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para crear un nuevo docente. | |
| | 2.- El sistema redirige a la Secretaria a un formulario para crear un nuevo docente. |
| 3.- La Secretaria ingresa los datos del nuevo docente. | |
| | 4.- El sistema valida los datos ingresados. |
| | 5.- El sistema crea al nuevo docente. |

Fuente: Elaboración propia

**Curso alterno:**

4.1.- Los datos ingresados no son válidos.

4.2.- Ya existe un docente con ese número de cédula.

**Caso de uso:** Modificar docente.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria modificar los datos de un docente.

**Resumen:** La Secretaria selecciona un docente y modifica sus datos, el sistema los valida y los actualiza.

**Curso de los eventos:**

Tabla 3.31. Caso de uso - Modificar docente

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para modificar docente. | |
| | 2.- El sistema redirige a la Secretaria a un formulario para modificar al docente seleccionado. |
| 3.- La Secretaria ingresa los nuevos datos del docente seleccionado. | |
| | 4.- El sistema valida los datos ingresados. |
| | 5.- El sistema actualiza los datos del docente seleccionado. |

Fuente: Elaboración propia

**Curso alterno:**

4.1.- Los datos ingresados no son válidos.

4.2.- Ya existe un docente con ese número de cédula.

**Caso de uso:** Eliminar docente.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria eliminar un docente.

**Resumen:** La Secretaria selecciona un docente y lo elimina.

**Curso de los eventos:**

Tabla 3.32. Caso de uso - Eliminar docente

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para eliminar docente. | |
| | 2.- El sistema presenta a la Secretaria un mensaje de confirmación. |
| 3.- La Secretaria confirma la acción realizada. | |
| | 4.- El sistema elimina al docente seleccionado. |
| | 5.- El sistema actualiza la tabla de datos de la pantalla de gestión de docentes de la aplicación. |

Fuente: Elaboración propia

**Curso alterno:**

3.1.- La Secretaria no confirma la acción realizada.

4.1.- El sistema no elimina al docente seleccionado.

**Caso de uso:** Consultar oferta.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria consultar la base de datos de la aplicación en busca de una oferta en específico.

**Resumen:** La Secretaria ingresa un término de búsqueda para que el sistema filtre la base de datos y devuelva un resultado de la búsqueda.

**Curso de los eventos:**

Tabla 3.33. Caso de uso - Consultar oferta

| Actores | Sistema |
|---|---|
| 1.- La Secretaria ingresa un término en el campo de búsqueda de la pantalla de gestión de ofertas de la aplicación. | 2.- El sistema filtra la base de datos en búsqueda de un resultado que coincida con el término de búsqueda ingresado.<br>3.- El sistema devuelve los resultados de la consulta. |

Fuente: Elaboración propia

**Curso alterno:**

2.1.- No se encontraron resultados.

**Caso de uso:** Crear oferta.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria crear una nueva oferta.

**Resumen:** La Secretaria ingresa los datos de la nueva oferta para que el sistema los valide y pueda crear una nueva oferta.

**Curso de los eventos:**

Tabla 3.34. Caso de uso - Crear oferta

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para crear una nueva oferta. | |
| | 2.- El sistema redirige a la Secretaria a un formulario para crear una nueva oferta. |
| 3.- La Secretaria ingresa los datos de la nueva oferta. | |
| | 4.- El sistema valida los datos ingresados. |
| | 5.- El sistema crea la nueva oferta. |

Fuente: Elaboración propia

**Curso alterno:**

4.1.- Los datos ingresados no son válidos.

**Caso de uso:** Modificar oferta.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria modificar los datos de una oferta.

**Resumen:** La Secretaria selecciona una oferta y modifica sus datos, el sistema los valida y los actualiza.

**Curso de los eventos:**

Tabla 3.35. Caso de uso - Modificar oferta

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para modificar oferta. | |
| | 2.- El sistema redirige a la Secretaria a un formulario para modificar la oferta seleccionada. |
| 3.- La Secretaria ingresa los nuevos datos de la oferta seleccionada. | |
| | 4.- El sistema valida los datos ingresados. |
| | 5.- El sistema actualiza los datos de la oferta seleccionada. |

Fuente: Elaboración propia

**Curso alterno:**

4.1.- Los datos ingresados no son válidos.

**Caso de uso:** Eliminar oferta.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria eliminar una oferta.

**Resumen:** La Secretaria selecciona una oferta y la elimina.

**Curso de los eventos:**

Tabla 3.36. Caso de uso - Eliminar oferta

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para eliminar oferta. | |
| | 2.- El sistema presenta a la Secretaria un mensaje de confirmación. |
| 3.- La Secretaria confirma la acción realizada. | |
| | 4.- El sistema elimina la oferta seleccionada. |
| | 5.- El sistema actualiza la tabla de datos de la pantalla de gestión de ofertas de la aplicación. |

Fuente: Elaboración propia

**Curso alterno:**

3.1.- La Secretaria no confirma la acción realizada.

4.1.- El sistema no elimina la oferta seleccionada.

**Caso de uso:** Consultar estudiante.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria consultar la base de datos de la aplicación en busca de un estudiante en específico.

**Resumen:** La Secretaria ingresa un término de búsqueda para que el sistema filtre la base de datos y devuelva un resultado de la búsqueda.

**Curso de los eventos:**

Tabla 3.37. Caso de uso - Consultar estudiante

| Actores | Sistema |
|---|---|
| 1.- La Secretaria ingresa un término en el campo de búsqueda de la pantalla de gestión de estudiantes de la aplicación. | 2.- El sistema filtra la base de datos en búsqueda de un resultado que coincida con el término de búsqueda ingresado.<br>3.- El sistema devuelve los resultados de la consulta. |

Fuente: Elaboración propia

**Curso alterno:**

2.1.- No se encontraron resultados.

**Caso de uso:** Crear estudiante.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria crear un nuevo estudiante.

**Resumen:** La Secretaria ingresa los datos del nuevo estudiante para que el sistema los valide y pueda crear un nuevo estudiante.

**Curso de los eventos:**

Tabla 3.38. Caso de uso - Crear estudiante

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para crear un nuevo estudiante. | |
| | 2.- El sistema redirige a la Secretaria a un formulario para crear un nuevo estudiante. |
| 3.- La Secretaria ingresa los datos del nuevo estudiante. | |
| | 4.- El sistema valida los datos ingresados. |
| | 5.- El sistema crea al nuevo estudiante. |

Fuente: Elaboración propia

**Curso alterno:**

4.1.- Los datos ingresados no son válidos.

4.2.- Ya existe un estudiante con ese número de cédula.

**Caso de uso:** Modificar estudiante.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria modificar los datos de un estudiante.

**Resumen:** La Secretaria selecciona un estudiante y modifica sus datos, el sistema los valida y los actualiza.

**Curso de los eventos:**

Tabla 3.39. Caso de uso - Modificar estudiante

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para modificar estudiante. | |
| | 2.- El sistema redirige a la Secretaria a un formulario para modificar al estudiante seleccionado. |
| 3.- La Secretaria ingresa los nuevos datos del estudiante seleccionado. | |
| | 4.- El sistema valida los datos ingresados. |
| | 5.- El sistema actualiza los datos del estudiante seleccionado. |

Fuente: Elaboración propia

**Curso alterno:**

4.1.- Los datos ingresados no son válidos.

4.2.- Ya existe un estudiante con ese número de cédula.

**Caso de uso:** Eliminar estudiante.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria eliminar un estudiante.

**Resumen:** La Secretaria selecciona un estudiante y lo elimina.

**Curso de los eventos:**

Tabla 3.40. Caso de uso - Eliminar estudiante

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para eliminar estudiante. | |
| | 2.- El sistema presenta a la Secretaria un mensaje de confirmación. |
| 3.- La Secretaria confirma la acción realizada. | |
| | 4.- El sistema elimina al estudiante seleccionado. |
| | 5.- El sistema actualiza la tabla de datos de la pantalla de gestión de estudiantes de la aplicación. |

Fuente: Elaboración propia

**Curso alterno:**

3.1.- La Secretaria no confirma la acción realizada.

4.1.- El sistema no elimina al estudiante seleccionado.

**Caso de uso:** Consultar matrícula.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria consultar la base de datos de la aplicación en busca de una matrícula en específico.

**Resumen:** La Secretaria ingresa un término de búsqueda para que el sistema filtre la base de datos y devuelva un resultado de la búsqueda.

**Curso de los eventos:**

Tabla 3.41. Caso de uso - Consultar matrícula

| Actores | Sistema |
|---|---|
| 1.- La Secretaria ingresa un término en el campo de búsqueda de la pantalla de gestión de matrículas de la aplicación. | 2.- El sistema filtra la base de datos en búsqueda de un resultado que coincida con el término de búsqueda ingresado.<br>3.- El sistema devuelve los resultados de la consulta. |

Fuente: Elaboración propia

**Curso alterno:**

2.1.- No se encontraron resultados.

**Caso de uso:** Crear matrícula.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria crear una nueva matrícula.

**Resumen:** La Secretaria ingresa los datos de la nueva matrícula para que el sistema los valide y pueda crear una nueva matrícula.

**Curso de los eventos:**

Tabla 3.42. Caso de uso - Crear matrícula

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para crear una nueva matrícula. | |
| | 2.- El sistema redirige a la Secretaria a un formulario para crear una nueva matrícula. |
| 3.- La Secretaria ingresa los datos de la nueva matrícula. | |
| | 4.- El sistema valida los datos ingresados. |
| | 5.- El sistema crea la nueva matrícula. |

Fuente: Elaboración propia

**Curso alterno:**

4.1.- Los datos ingresados no son válidos.

**Caso de uso:** Modificar matrícula.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria modificar los datos de una matrícula.

**Resumen:** La Secretaria selecciona una matrícula y modifica sus datos, el sistema los valida y los actualiza.

**Curso de los eventos:**

Tabla 3.43. Caso de uso - Modificar matrícula

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para modificar matrícula. | |
| | 2.- El sistema redirige a la Secretaria a un formulario para modificar la matrícula seleccionada. |
| 3.- La Secretaria ingresa los nuevos datos de la matrícula seleccionada. | |
| | 4.- El sistema valida los datos ingresados. |
| | 5.- El sistema actualiza los datos de la matrícula seleccionada. |

Fuente: Elaboración propia

**Curso alterno:**

4.1.- Los datos ingresados no son válidos.

**Caso de uso:** Eliminar matrícula.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria eliminar una matrícula.

**Resumen:** La Secretaria selecciona una matrícula y la elimina.

**Curso de los eventos:**

Tabla 3.44. Caso de uso - Eliminar matrícula

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para eliminar matrícula. | |
| | 2.- El sistema presenta a la Secretaria un mensaje de confirmación. |
| 3.- La Secretaria confirma la acción realizada. | |
| | 4.- El sistema elimina la matrícula seleccionada. |
| | 5.- El sistema actualiza la tabla de datos de la pantalla de gestión de matrículas de la aplicación. |

Fuente: Elaboración propia

**Curso alterno:**

3.1.- La Secretaria no confirma la acción realizada.

4.1.- El sistema no elimina la matrícula seleccionada.

**Caso de uso:** Consultar pensión.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria consultar la base de datos de la aplicación en busca de una pensión en específico.

**Resumen:** La Secretaria ingresa un término de búsqueda para que el sistema filtre la base de datos y devuelva un resultado de la búsqueda.

**Curso de los eventos:**

Tabla 3.45. Caso de uso - Consultar pensión

| Actores | Sistema |
|---|---|
| 1.- La Secretaria ingresa un término en el campo de búsqueda de la pantalla de gestión de pensiones de la aplicación. | 2.- El sistema filtra la base de datos en búsqueda de un resultado que coincida con el término de búsqueda ingresado.<br>3.- El sistema devuelve los resultados de la consulta. |

Fuente: Elaboración propia

**Curso alterno:**

2.1.- No se encontraron resultados.

**Caso de uso:** Crear pensión.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria crear una nueva pensión.

**Resumen:** La Secretaria ingresa los datos de la nueva pensión para que el sistema los valide y pueda crear una nueva pensión.

**Curso de los eventos:**

Tabla 3.46. Caso de uso - Crear pensión

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para crear una nueva pensión. | |
| | 2.- El sistema redirige a la Secretaria a un formulario para crear una nueva pensión. |
| 3.- La Secretaria ingresa los datos de la nueva pensión. | |
| | 4.- El sistema valida los datos ingresados. |
| | 5.- El sistema crea la nueva pensión. |

Fuente: Elaboración propia

**Curso alterno:**

4.1.- Los datos ingresados no son válidos.

**Caso de uso:** Modificar pensión.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria modificar los datos de una pensión.

**Resumen:** La Secretaria selecciona una pensión y modifica sus datos, el sistema los valida y los actualiza.

**Curso de los eventos:**

Tabla 3.47. Caso de uso - Modificar pensión

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para modificar pensión. | |
| | 2.- El sistema redirige a la Secretaria a un formulario para modificar la pensión seleccionada. |
| 3.- La Secretaria ingresa los nuevos datos de la pensión seleccionada. | |
| | 4.- El sistema valida los datos ingresados. |
| | 5.- El sistema actualiza los datos de la pensión seleccionada. |

Fuente: Elaboración propia

**Curso alterno:**

4.1.- Los datos ingresados no son válidos.

**Caso de uso:** Eliminar pensión.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria eliminar una pensión.

**Resumen:** La Secretaria selecciona una pensión y la elimina.

**Curso de los eventos:**

Tabla 3.48. Caso de uso - Eliminar pensión

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para eliminar pensión. | |
| | 2.- El sistema presenta a la Secretaria un mensaje de confirmación. |
| 3.- La Secretaria confirma la acción realizada. | |
| | 4.- El sistema elimina la pensión seleccionada. |
| | 5.- El sistema actualiza la tabla de datos de la pantalla de gestión de pensiones de la aplicación. |

Fuente: Elaboración propia

**Curso alterno:**

3.1.- La Secretaria no confirma la acción realizada.

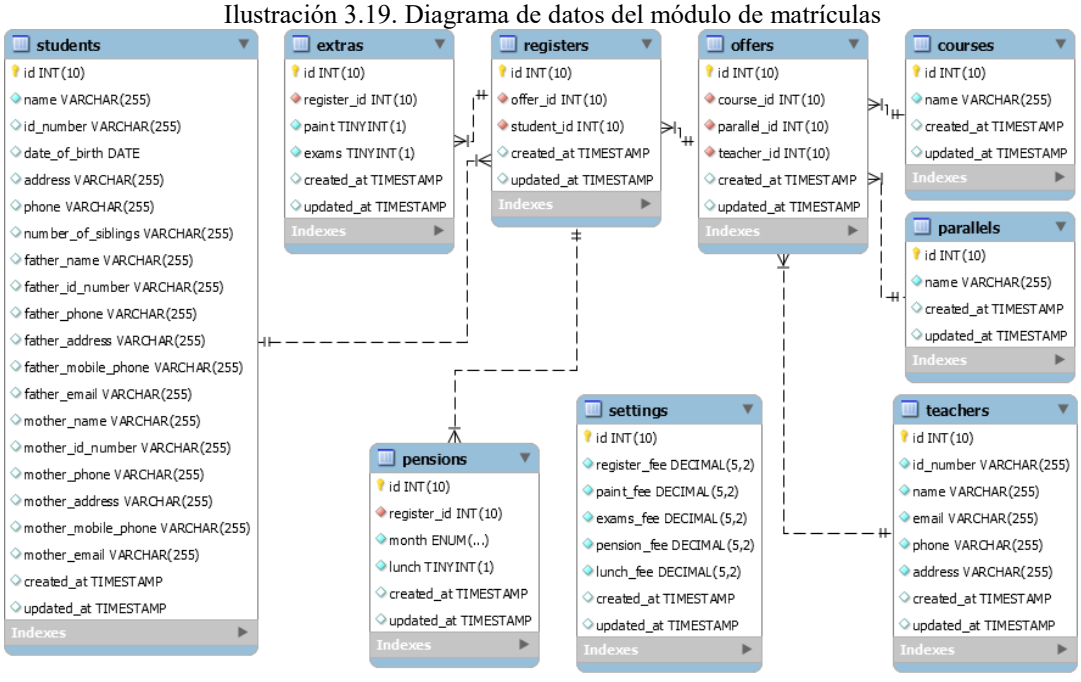
4.1.- El sistema no elimina la pensión seleccionada.

### 3.2.1.2.Diagrama de datos

Para el desarrollo del módulo de matrículas se utilizaron un numero de nueve (9) tablas que almacenan los siguientes datos: la tabla *settings* almacena las configuraciones del año lectivo, la tabla *courses* almacena los cursos del año lectivo mientras que la tabla *parallels* almacena los paralelos, la tabla *teachers* almacena los docentes de la Institución y la tabla *offers* que almacena las ofertas académicas de esta.

La tabla *students* almacena los datos completos de los estudiantes junto con los datos de sus representantes legales, la tabla *registers* almacena los datos de las matrículas de los estudiantes, está tabla está relacionada con la tabla *extras* que almacena los valores extra de las matrículas (valores de pintura y exámenes), finalmente se encuentra la tabla *pensions* que almacena las diferentes pensiones de los estudiantes.

Ilustración 3.19. Diagrama de datos del módulo de matrículas




Fuente: Elaboración propia

### 3.2.1.3.Diccionario de datos

Descripción y estructura de la tabla *settings*.

Tabla 3.49. Tabla settings

| Nombre | Tipo | Razón |
|---|---|---|
| id | int(10) | id auto incremental y clave primaria de la tabla |
| register_fee | decimal(5,2) | precio de la matrícula |
| paint_fee | decimal(5,2) | precio de la pintura |
| exams_fee | decimal(5,2) | precio de los exámenes |
| pension_fee | decimal(5,2) | precio de la pensión |
| lunch_fee | decimal(5,2) | precio del servicio de almuerzo |
| created_at | timestamp | fecha de creación del registro |
| updated_at | timestamp | fecha de actualización del registro |

Fuente: Elaboración propia

Descripción y estructura de la tabla *courses*.

Tabla 3.50. Tabla courses

| Nombre | Tipo | Razón |
|---|---|---|
| id | int(10) | id auto incremental y clave primaria de la tabla |
| name | varchar(255) | nombre del curso |
| created_at | timestamp | fecha de creación del registro |
| updated_at | timestamp | fecha de actualización del registro |

Fuente: Elaboración propia

Descripción y estructura de la tabla *offers*.

Tabla 3.51. Tabla offers

| Nombre | Tipo | Razón |
|---|---|---|
| id | int(10) | id auto incremental y clave primaria de la tabla |
| course_id | int(10) | clave principal y foránea de la tabla students |
| parallel_id | int(10) | clave principal y foránea de la tabla students |
| teacher_id | int(10) | clave principal y foránea de la tabla students |
| created_at | timestamp | fecha de creación del registro |
| updated_at | timestamp | fecha de actualización del registro |

Fuente: Elaboración propia

Descripción y estructura de la tabla *teachers*.

Tabla 3.52. Tabla teachers

| Nombre | Tipo | Razón |
|---|---|---|
| id | int(10) | id auto incremental y clave primaria de la tabla |
| id_number | varchar(255) | número de cédula del docente |
| name | varchar(255) | nombre completo del docente |
| email | varchar(255) | correo electrónico del docente |
| phone | varchar(255) | número telefónico del docente |
| address | varchar(255) | dirección del docente |
| created_at | timestamp | fecha de creación del registro |
| updated_at | timestamp | fecha de actualización del registro |

Fuente: Elaboración propia

Descripción y estructura de la tabla *parallels*.

Tabla 3.53. Tabla parallels

| Nombre | Tipo | Razón |
|---|---|---|
| id | int(10) | id auto incremental y clave primaria de la tabla |
| name | varchar(255) | nombre del paralelo |
| created_at | timestamp | fecha de creación del registro |
| updated_at | timestamp | fecha de actualización del registro |

Fuente: Elaboración propia

Descripción y estructura de la tabla *extras*.

Tabla 3.54. Tabla extras

| Nombre | Tipo | Razón |
|---|---|---|
| id | int(10) | id auto incremental y clave primaria de la tabla |
| register_id | int(10) | clave principal y foránea de la tabla registers |
| paint | tinyint(1) | valor extra correspondiente a pintura |
| exams | tinyint(1) | valor extra correspondiente a exámenes |
| created_at | timestamp | fecha de creación del registro |
| updated_at | timestamp | fecha de actualización del registro |

Fuente: Elaboración propia

Descripción y estructura de la tabla *pensions*.

Tabla 3.55. Tabla pensions

| Nombre | Tipo | Razón |
|---|---|---|
| id | int(10) | id auto incremental y clave primaria de la tabla |
| student_id | varchar(255) | clave principal y foránea de la tabla students |
| month | enum | mes del cual se registra la pensión |
| lunch | tinyint(1) | valor correspondiente al servicio de almuerzo |
| created_at | timestamp | fecha de creación del registro |
| updated_at | timestamp | fecha de actualización del registro |

Fuente: Elaboración propia

Descripción y estructura de la tabla *registers*.

Tabla 3.56. Tabla registers

| Nombre | Tipo | Razón |
|---|---|---|
| id | int(10) | id auto incremental y clave primaria de la tabla |
| course_id | int(10) | clave principal y foránea de la tabla courses |
| parallel_id | int(10) | clave principal y foránea de la tabla parallels |
| teacher_id | int(10) | clave principal y foránea de la tabla teachers |
| student_id | int(10) | clave principal y foránea de la tabla students |
| created_at | timestamp | fecha de creación del registro |
| updated_at | timestamp | fecha de actualización del registro |

Fuente: Elaboración propia

Descripción y estructura de la tabla *students*.

Tabla 3.57. Tabla students

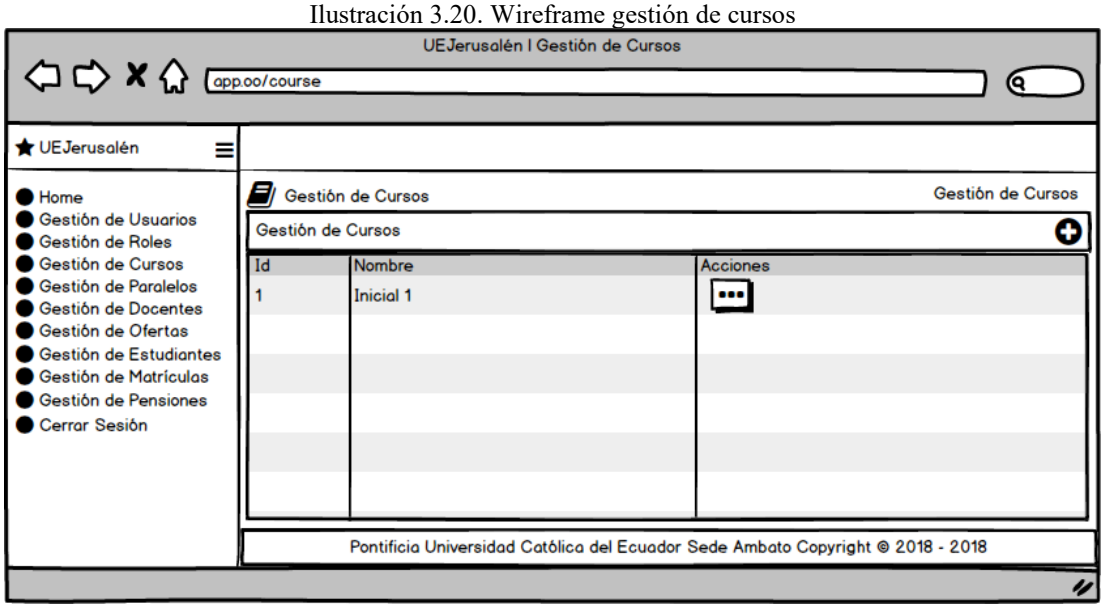
| Nombre | Tipo | Razón |
|---|---|---|
| id | int(10) | id auto incremental y clave primaria |
| name | varchar(255) | nombre completo del estudiante |
| id_number | varchar(255) | número de cédula del estudiante |
| date_of_birth | date | fecha de nacimiento del estudiante |
| address | varchar(255) | dirección del estudiante |
| phone | varchar(255) | número de teléfono del estudiante |
| number_of_siblings | varchar(255) | número de hermanos |
| father_name | varchar(255) | nombre completo del padre |
| father_id_number | varchar(255) | número de cédula del padre |
| father_phone | varchar(255) | número de teléfono del padre |
| father_address | varchar(255) | dirección del padre |
| father_mobile_phone | varchar(255) | número de teléfono celular del padre |
| father_email | varchar(255) | dirección de correo electrónico del padre |
| mother_name | varchar(255) | nombre completo de la madre |
| mother_id_number | varchar(255) | número de cédula de la madre |
| mother_phone | varchar(255) | número de teléfono de la madre |
| mother_address | varchar(255) | dirección de la madre |
| mother_mobile_phone | varchar(255) | número de teléfono celular de la madre |
| mother_email | varchar(255) | dirección de correo electrónico de la madre |

Fuente: Elaboración propia

### 3.2.1.4.Diseño

Diseño de la interfaz de usuario para la gestión de cursos del módulo de matrículas.

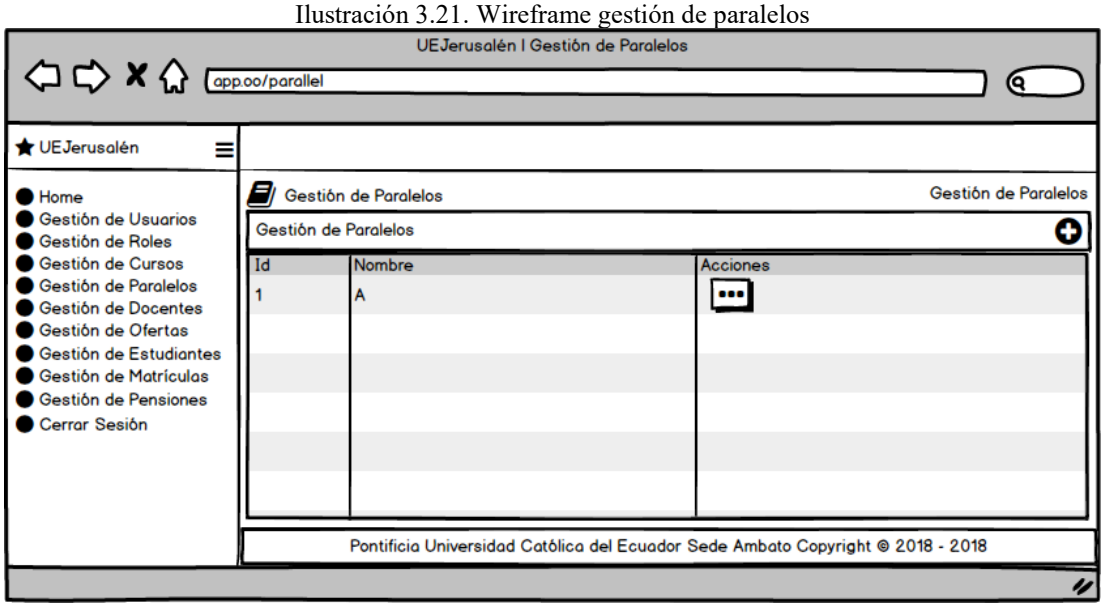
Ilustración 3.20. Wireframe gestión de cursos




Fuente: Elaboración propia

Diseño de la interfaz de usuario para la gestión de paralelos del módulo de matrículas.

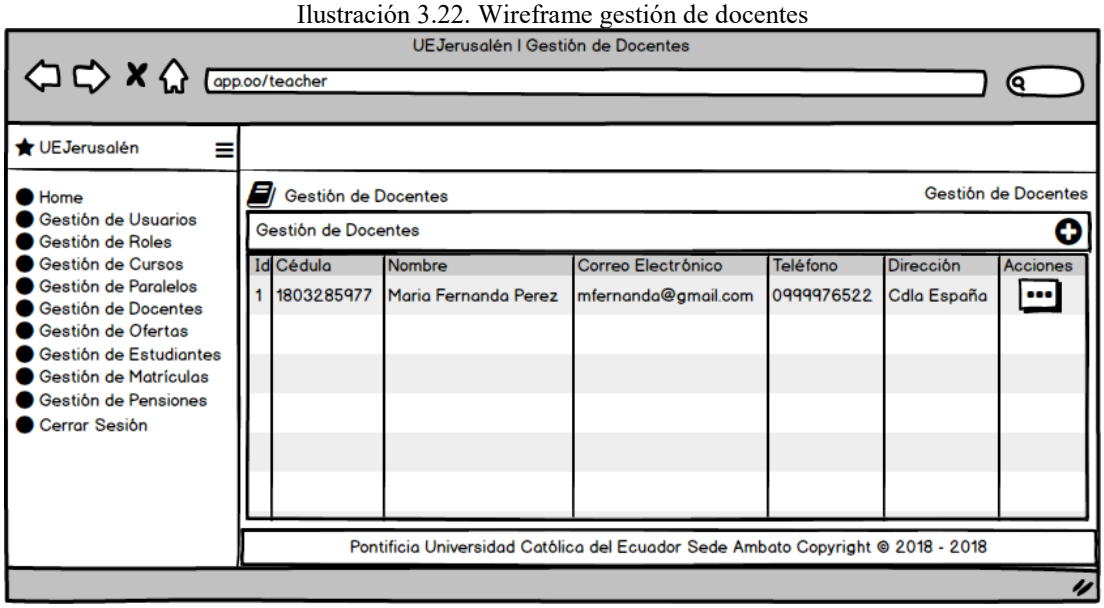
Ilustración 3.21. Wireframe gestión de paralelos




Fuente: Elaboración propia

Diseño de la interfaz de usuario para la gestión de docentes del módulo de matrículas.

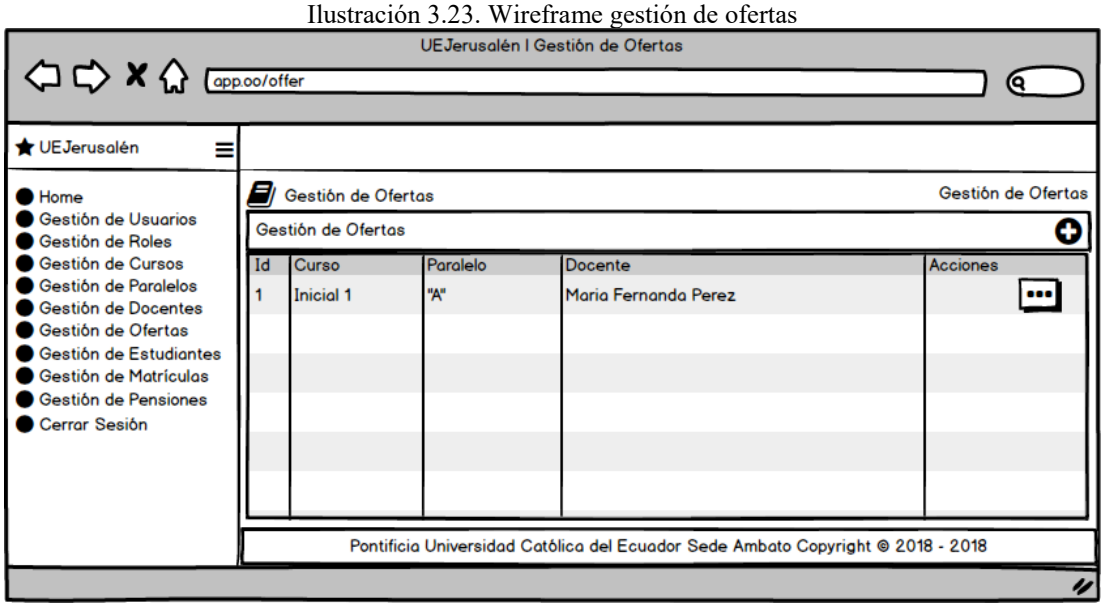
Ilustración 3.22. Wireframe gestión de docentes




Fuente: Elaboración propia

Diseño de la interfaz de usuario para la gestión de ofertas del módulo de matrículas.

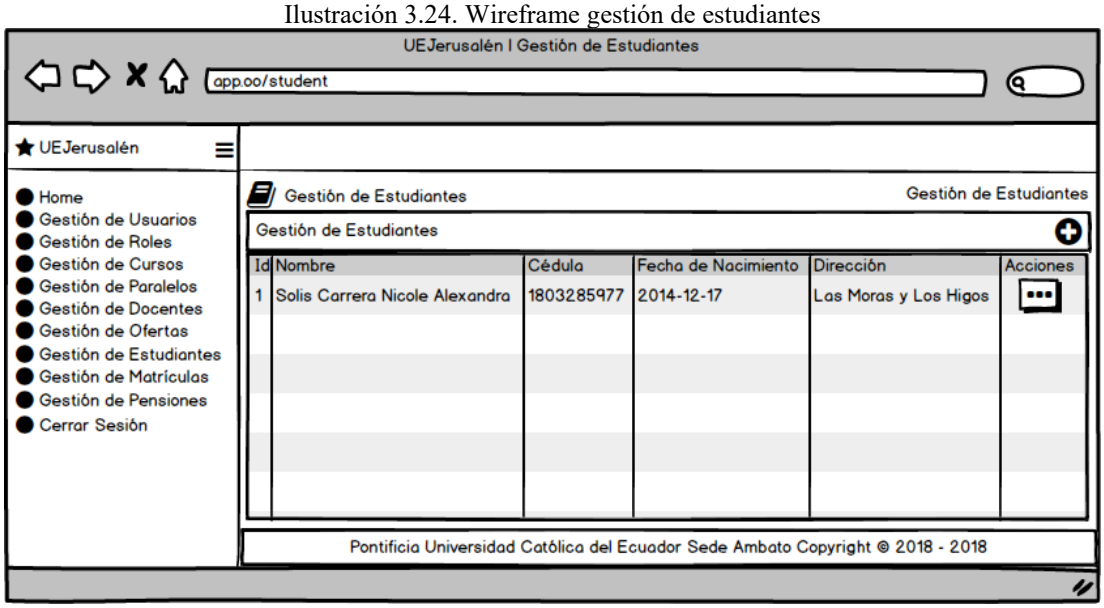
Ilustración 3.23. Wireframe gestión de ofertas




Fuente: Elaboración propia

Diseño de la interfaz de usuario para la gestión de estudiante del módulo de matrículas.

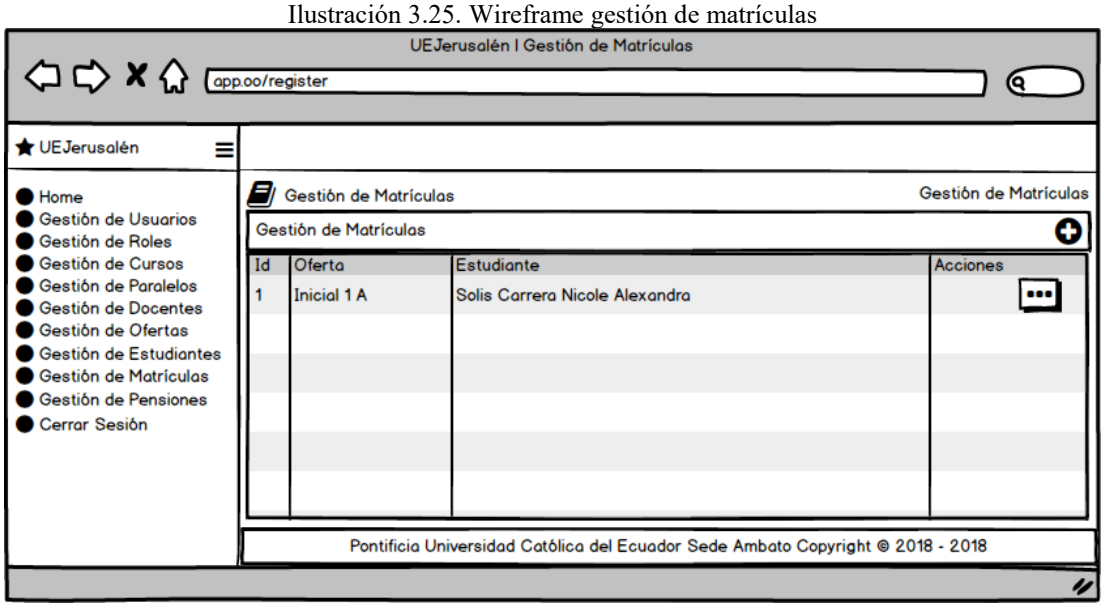
Ilustración 3.24. Wireframe gestión de estudiantes




Fuente: Elaboración propia

Diseño de la interfaz de usuario para la gestión de matrículas del módulo de matrículas.

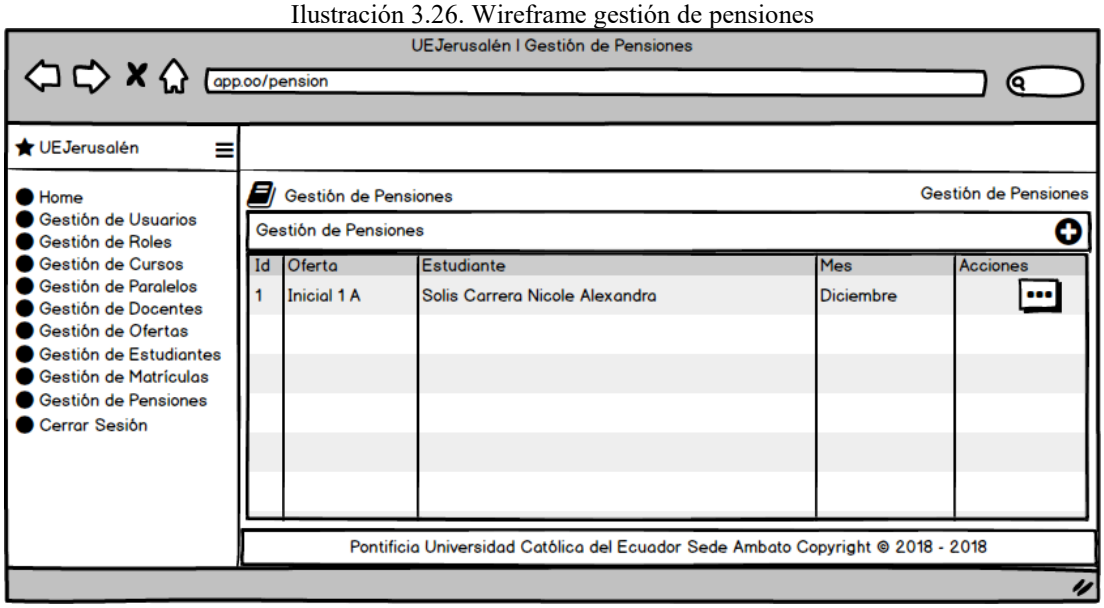
Ilustración 3.25. Wireframe gestión de matrículas




Fuente: Elaboración propia

Diseño de la interfaz de usuario para la gestión de pensiones del módulo de matrículas.

Ilustración 3.26. Wireframe gestión de pensiones




Fuente: Elaboración propia

### 3.2.2.Colaboración

#### 3.2.2.1.Programación

En esta iteración se empieza a desarrollar el módulo de matrículas y pensiones de los estudiantes de la Institución, cabe resaltar que en esta iteración no se realizaron reuniones de colaboración con el cliente, el proceso empezó por matricular a los estudiantes en un curso y un paralelo en donde un docente pueda cumplir una función como tutor, por lo tanto, existió la necesidad de crear una oferta académica que pueda almacenar la clave primaria de un registro de la tabla de cursos, de la tabla de paralelos y de la tabla de docentes, para lograr esto fue necesario construir las relaciones correctamente dentro de cada modelo con el fin de que cada modelo esté relacionado.

Las relaciones de los archivos *Course.php*, *Parallel.php* y *Teacher.php* con el modelo de las ofertas es la misma, esta relación permite a la aplicación obtener los datos del recurso con el cual está relacionada la oferta, la estructura de la relación es la siguiente:

Ilustración 3.27. Relaciónes de las ofertas

```php
public function offers()
{
    return $this->hasMany( related: 'App\Offer');
}
```

Fuente: Elaboración propia

Una vez establecidas las primeras relaciones, en el archivo *Offer.php* se procede a determinar las relaciones inversas a las anteriores de la siguiente forma:

Ilustración 3.28. Relaciones inversas de las ofertas

```php
public function course()
{
    return $this->belongsTo( related: 'App\Course');
}

public function parallel()
{
    return $this->belongsTo( related: 'App\Parallel');
}

public function teacher()
{
    return $this->belongsTo( related: 'App\Teacher');
}
```

Fuente: Elaboración propia

De esa forma se puede crear una oferta con el nombre del curso, el nombre del paralelo y el nombre del docente respectivamente, esta actividad se la realiza en el archivo *OfferController.php* con el método para crear una oferta académica a continuación:

Ilustración 3.29. Método para crear una oferta

```php
public function store(Request $request)
{
    $this->validate($request, [
        'course_id' => 'required',
        'parallel_id' => 'required',
        'teacher_id' => 'required'
    ]);

    (new Offer())->create([
        'course_id' => $request['course_id'],
        'parallel_id' => $request['parallel_id'],
        'teacher_id' => $request['teacher_id']
    ]);

    flash( message: 'Oferta Creada.')->success();

    return redirect()->route( route: 'offer.index');
}
```

Fuente: Elaboración propia

Para poder registrar a los estudiantes en la aplicación primero se determinan las relaciones que tienen los estudiantes y estas se las colocan en el archivo *Student.php,* las relaciones de los estudiantes se conforman de la siguiente manera:

Ilustración 3.30. Relaciones de los estudiantes

```php
public function register()
{
    return $this->hasOne( related: 'App\Register');
}
```

Fuente: Elaboración propia

De acuerdo con lo anterior un estudiante solamente puede tener una matrícula, es necesario resaltar el hecho de que un estudiante no puede tener dos matrículas, el método para crear un estudiante se lo encuentra en el archivo *StudentController.php*:

Ilustración 3.31. Método para crear un estudiante

```php
public function store(Request $request)
{
    $this->validate($request, [
        'name' => 'required|string|max:255',
        'id_number' => 'nullable|string|max:255|unique:students',
        'date_of_birth' => 'nullable|date',
        'address' => 'nullable|string|max:255',
        'phone' => 'nullable|string|max:255',
        'number_of_siblings' => 'nullable|string|max:255',
        'father_name' => 'nullable|string|max:255',
        'father_id_number' => 'nullable|string|max:255',
        'father_phone' => 'nullable|string|max:255',
        'father_address' => 'nullable|string|max:255',
        'father_mobile_phone' => 'nullable|string|max:255',
        'father_email' => 'nullable|string|max:255',
        'mother_name' => 'nullable|string|max:255',
        'mother_id_number' => 'nullable|string|max:255',
        'mother_phone' => 'nullable|string|max:255',
        'mother_address' => 'nullable|string|max:255',
        'mother_mobile_phone' => 'nullable|string|max:255',
        'mother_email' => 'nullable|string|max:255'
    ]);

    (new Student())->create([
        'name' => $request['name'],
        'id_number' => $request['id_number'],
        'date_of_birth' => $request['date_of_birth'],
        'address' => $request['address'],
        'phone' => $request['phone'],
        'number_of_siblings' => $request['number_of_siblings'],
        'father_name' => $request['father_name'],
        'father_id_number' => $request['father_id_number'],
        'father_phone' => $request['father_phone'],
        'father_address' => $request['father_address'],
        'father_mobile_phone' => $request['father_mobile_phone'],
        'father_email' => $request['father_email'],
        'mother_name' => $request['mother_name'],
        'mother_id_number' => $request['mother_id_number'],
        'mother_phone' => $request['mother_phone'],
        'mother_address' => $request['mother_address'],
        'mother_mobile_phone' => $request['mother_mobile_phone'],
        'mother_email' => $request['mother_email']
    ]);

    flash( message: 'Estudiante Creado.')->success();

    return redirect()->route( route: 'student.index');
}
```

Fuente: Elaboración propia

El campo perteneciente a la fecha de nacimiento utiliza un componente conocido como *datepicker,* este componente despliega un calendario que facilita al usuario elegir una fecha, para inicializar este componente en el archivo *index.blade.php* se colocan enlaces a la librería de estilos y al archivo de los *scripts* y se coloca el siguiente código:

Ilustración 3.32. Código para elegir la fecha de nacimiento del estudiante

```
@push('styles')
    {{ Html::style('https://cdnjs.cloudflare.com/ajax/libs/bootstrap-datepicker/1.8.0/css/bootstrap-datepicker.css') }}
@endpush

@push('scripts')
    {{ Html::script('https://cdnjs.cloudflare.com/ajax/libs/bootstrap-datepicker/1.8.0/js/bootstrap-datepicker.js') }}
    {{ Html::script('https://cdnjs.cloudflare.com/ajax/libs/bootstrap-datepicker/1.8.0/locales/bootstrap-datepicker.es.min.js') }}
    <script>
        $(function () {
            $('#datepicker').datepicker({
                autoclose: true,
                format: 'yyyy-mm-dd',
                language: 'es-ES',
                orientation: 'top',
                weekStart: 0,
                todayHighlight: true
            });
        });
    </script>
@endpush
```

Fuente: Elaboración propia

Para poder matricular a un estudiante es necesario antes modificar la tabla de configuraciones para que se le asigne un valor a la transacción realizada, esta tabla contiene los valores correspondientes a la matrícula en formato decimal, valores extra de la matrícula, pensión mensual y el valor por el servicio de almuerzo, en el archivo *HomeController.php* el método para modificar las configuraciones es el siguiente:

Ilustración 3.33. Método para actualizar las configuraciones

```
    public function update(Request $request, Setting $setting)
    {
        $this->validate($request, [
            'register_fee' => 'required|regex:/^\d*(\.\d{1,2})?$/',
            'paint_fee' => 'required|regex:/^\d*(\.\d{1,2})?$/',
            'exams_fee' => 'required|regex:/^\d*(\.\d{1,2})?$/',
            'pension_fee' => 'required|regex:/^\d*(\.\d{1,2})?$/',
            'lunch_fee' => 'required|regex:/^\d*(\.\d{1,2})?$/'
        ]);

        $setting->update([
            'register_fee' => $request['register_fee'],
            'paint_fee' => $request['paint_fee'],
            'exams_fee' => $request['exams_fee'],
            'pension_fee' => $request['pension_fee'],
            'lunch_fee' => $request['lunch_fee']
        ]);

        flash( message: 'Configuraciones Actualizadas.')->success();

        return redirect()->route( route: 'setting.index');
    }
```

Fuente: Elaboración propia

Cada matrícula tiene una relación con un modelo en donde se almacenan los valores extra y también tiene una relación con las pensiones de los estudiantes, de esta forma:

Ilustración 3.34. Relaciones de las matrículas

```php
public function offer()
{
    return $this->belongsTo( related: 'App\Offer');
}

public function student()
{
    return $this->belongsTo( related: 'App\Student');
}

public function extra()
{
    return $this->hasOne( related: 'App\Extra');
}

public function pensions()
{
    return $this->hasMany( related: 'App\Pension');
}
```

Fuente: Elaboración propia

Para poder consultar y exportar los reportes de las matrículas y pensiones se utiliza un componente llamado *datatables*, este componente construye una tabla de datos dinámica, por ejemplo, en el archivo *index.blade.php* ubicado en la carpeta *registers* se tiene el siguiente *script* que tiene la función de crear la tabla dinámica de datos y exportar la consulta realizada a un archivo de Excel, el código es el siguiente:

Ilustración 3.35. Componente datatables para consultar y exportar datos

```php
@push('scripts')
    {{ Html::script('https://cdn.datatables.net/1.10.18/js/jquery.dataTables.js') }}
    {{ Html::script('https://cdn.datatables.net/1.10.18/js/dataTables.bootstrap.js') }}
    {{ Html::script('https://cdn.datatables.net/buttons/1.5.4/js/dataTables.buttons.js') }}
    {{ Html::script('https://cdn.datatables.net/buttons/1.5.4/js/buttons.html5.js') }}
    {{ Html::script('https://cdnjs.cloudflare.com/ajax/libs/jszip/2.5.0/jszip.js') }}
    <script>
        $(function () {
            let registers_table = $("#registers-table");
            registers_table.DataTable({
                dom: 'Bfrtip',
                search: {
                    "smart": false
                },
                buttons: [
                    {
                        extend: 'excel',
                        text: '<i class="fa fa-file-excel-o"></i> Exportar a Excel',
                        filename: '{{ date('Y-m-d') }}',
                        title: 'Escuela Jerusalén',
                        messageTop: 'Reporte de Matriculas',
                        exportOptions: {
                            columns: [1, 2]
                        }
                    }
                ]
            });
        });
    </script>
@endpush
```

Fuente: Elaboración propia

### 3.2.3.Aprendizaje

#### 3.2.3.1.Control de calidad

Para realizar el control de calidad del módulo desarrollado se programó una reunión con la Directora y la Secretaria de la Institución para probar la aplicación, el respectivo control de calidad se documentó con la siguiente prueba de aceptación la misma que contiene todos los requerimientos obtenidos del módulo y las respuestas del usuario.

Tabla 3.58. Prueba de aceptación de módulo de matrículas

| Prueba de aceptación del módulo de matrículas | | |
|---|---|---|
| Título: Revisión del módulo de matrículas | | |
| Fecha: 24-05-2018 | | |
| Usuario: Fanny Hernández & Karla Chicaiza | | |
| Requerimiento | Si | No |
| Puede crear, modificar y eliminar cursos | X | |
| Puede crear, modificar y eliminar paralelos | X | |
| Puede crear, modificar y eliminar docentes | X | |
| Puede crear, modificar y eliminar ofertas | X | |
| Puede crear, modificar y eliminar estudiantes | X | |
| Puede matricular estudiantes en las ofertas creadas | X | |
| Puede registrar varias pensiones de un solo estudiante | X | |
| Puede enviar correo recordatorio a los representantes | X | |
| Puede realizar consultas de los estudiantes | X | |
| Puede imprimir reportes de las matrículas y pensiones | X | |
| Puede navegar de manera intuitiva por la aplicación | X | |

Fuente: Elaboración propia

Como se indicó anteriormente en la primera iteración, el proceso de implementación se lo realizó en la última iteración cuando el proceso de desarrollo ha concluido.

## 3.3.Tercera iteración

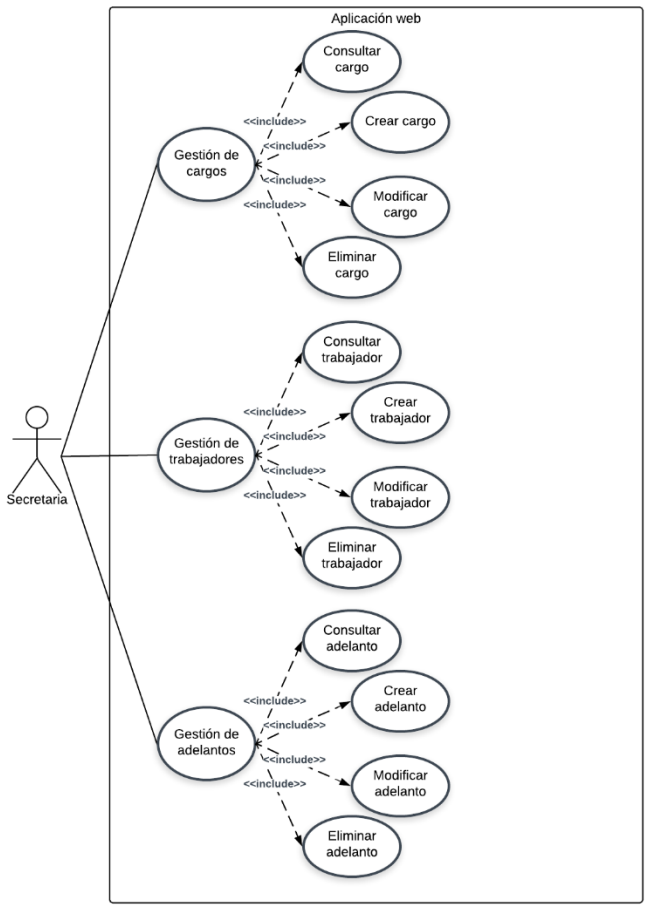
Después de la primera iteración y de la obtención inicial de requerimientos se cree que no es necesario realizar dicha actividad una segunda vez ya que estos son los mismos.

### 3.3.1.Especulación

#### 3.3.1.1.Casos de uso

A continuación, la documentación de los casos de uso del componente de adelantos.

Ilustración 3.36. Diagrama de casos de uso del módulo de adelantos




Fuente: Elaboración propia

**Caso de uso:** Consultar cargo.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria consultar la base de datos de la aplicación en busca de un cargo en específico.

**Resumen:** La Secretaria ingresa un término de búsqueda para que el sistema filtre la base de datos y devuelva un resultado de la búsqueda.

**Curso de los eventos:**

Tabla 3.59. Caso de uso - Consultar cargo

| Actores | Sistema |
|---|---|
| 1.- La Secretaria ingresa un término en el campo de búsqueda de la pantalla de gestión de cargos de la aplicación. | 2.- El sistema filtra la base de datos en búsqueda de un resultado que coincida con el término de búsqueda ingresado.<br>3.- El sistema devuelve los resultados de la consulta. |

Fuente: Elaboración propia

**Curso alterno:**

2.1.- No se encontraron resultados.

**Caso de uso:** Crear cargo.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria crear un nuevo cargo.

**Resumen:** La Secretaria ingresa los datos del nuevo cargo para que el sistema los valide y pueda crear un nuevo cargo.

**Curso de los eventos:**

Tabla 3.60. Caso de uso - Crear cargo

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para crear un nuevo cargo. | |
| | 2.- El sistema redirige a la Secretaria a un formulario para crear un nuevo cargo. |
| 3.- La Secretaria ingresa los datos del nuevo cargo. | |
| | 4.- El sistema valida los datos ingresados. |
| | 5.- El sistema crea el nuevo cargo. |

Fuente: Elaboración propia

**Curso alterno:**

4.1.- Los datos ingresados no son válidos.

**Caso de uso:** Modificar cargo.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria modificar los datos de un cargo.

**Resumen:** La Secretaria selecciona un cargo y modifica sus datos, el sistema los valida y los actualiza.

**Curso de los eventos:**

Tabla 3.61. Caso de uso - Modificar cargo

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para modificar cargo. | |
| | 2.- El sistema redirige a la Secretaria a un formulario para modificar el cargo seleccionado. |
| 3.- La Secretaria ingresa los nuevos datos del cargo seleccionado. | |
| | 4.- El sistema valida los datos ingresados. |
| | 5.- El sistema actualiza los datos del cargo seleccionado. |

Fuente: Elaboración propia

**Curso alterno:**

4.1.- Los datos ingresados no son válidos.

**Caso de uso:** Eliminar cargo.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria eliminar un cargo.

**Resumen:** La Secretaria selecciona un cargo y lo elimina.

**Curso de los eventos:**

Tabla 3.62. Caso de uso - Eliminar cargo

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para eliminar cargo. | |
| | 2.- El sistema presenta a la Secretaria un mensaje de confirmación. |
| 3.- La Secretaria confirma la acción realizada. | |
| | 4.- El sistema elimina el cargo seleccionado. |
| | 5.- El sistema actualiza la tabla de datos de la pantalla de gestión de cargos de la aplicación. |

Fuente: Elaboración propia

**Curso alterno:**

3.1.- La Secretaria no confirma la acción realizada.

4.1.- El sistema no elimina el cargo seleccionado.

**Caso de uso:** Consultar trabajador.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria consultar la base de datos de la aplicación en busca de un trabajador en específico.

**Resumen:** La Secretaria ingresa un término de búsqueda para que el sistema filtre la base de datos y devuelva un resultado de la búsqueda.

**Curso de los eventos:**

Tabla 3.63. Caso de uso - Consultar trabajador

| Actores | Sistema |
|---|---|
| 1.- La Secretaria ingresa un término en el campo de búsqueda de la pantalla de gestión de trabajadores de la aplicación. | 2.- El sistema filtra la base de datos en búsqueda de un resultado que coincida con el término de búsqueda ingresado.<br>3.- El sistema devuelve los resultados de la consulta. |

Fuente: Elaboración propia

**Curso alterno:**

2.1.- No se encontraron resultados.

**Caso de uso:** Crear trabajador.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria crear un nuevo trabajador.

**Resumen:** La Secretaria ingresa los datos del nuevo trabajador para que el sistema los valide y pueda crear un nuevo trabajador.

**Curso de los eventos:**

Tabla 3.64. Caso de uso - Crear trabajador

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para crear un nuevo trabajador. | |
| | 2.- El sistema redirige a la Secretaria a un formulario para crear un nuevo trabajador. |
| 3.- La Secretaria ingresa los datos del nuevo trabajador. | |
| | 4.- El sistema valida los datos ingresados. |
| | 5.- El sistema crea al nuevo trabajador. |

Fuente: Elaboración propia

**Curso alterno:**

4.1.- Los datos ingresados no son válidos.

**Caso de uso:** Modificar trabajador.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria modificar los datos de un trabajador.

**Resumen:** La Secretaria selecciona un trabajador y modifica sus datos, el sistema los valida y los actualiza.

**Curso de los eventos:**

Tabla 3.65. Caso de uso - Modificar trabajador

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para modificar trabajador. | |
| | 2.- El sistema redirige a la Secretaria a un formulario para modificar al trabajador seleccionado. |
| 3.- La Secretaria ingresa los nuevos datos del trabajador seleccionado. | |
| | 4.- El sistema valida los datos ingresados. |
| | 5.- El sistema actualiza los datos del trabajador seleccionado. |

Fuente: Elaboración propia

**Curso alterno:**

4.1.- Los datos ingresados no son válidos.

**Caso de uso:** Eliminar trabajador.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria eliminar un trabajador.

**Resumen:** La Secretaria selecciona un trabajador y lo elimina.

**Curso de los eventos:**

Tabla 3.66. Caso de uso - Eliminar trabajador

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para eliminar trabajador. | |
| | 2.- El sistema presenta a la Secretaria un mensaje de confirmación. |
| 3.- La Secretaria confirma la acción realizada. | |
| | 4.- El sistema elimina al trabajador seleccionado. |
| | 5.- El sistema actualiza la tabla de datos de la pantalla de gestión de trabajadores de la aplicación. |

Fuente: Elaboración propia

**Curso alterno:**

3.1.- La Secretaria no confirma la acción realizada.

4.1.- El sistema no elimina al trabajador seleccionado.

**Caso de uso:** Consultar adelanto.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria consultar la base de datos de la aplicación en busca de un adelanto en específico.

**Resumen:** La Secretaria ingresa un término de búsqueda para que el sistema filtre la base de datos y devuelva un resultado de la búsqueda.

**Curso de los eventos:**

Tabla 3.67. Caso de uso - Consultar adelanto

| Actores | Sistema |
|---|---|
| 1.- La Secretaria ingresa un término en el campo de búsqueda de la pantalla de gestión de adelantos de la aplicación. | 2.- El sistema filtra la base de datos en búsqueda de un resultado que coincida con el término de búsqueda ingresado.<br>3.- El sistema devuelve los resultados de la consulta. |

Fuente: Elaboración propia

**Curso alterno:**

2.1.- No se encontraron resultados.

**Caso de uso:** Crear adelanto.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria crear un nuevo adelanto.

**Resumen:** La Secretaria ingresa los datos del nuevo adelanto para que el sistema los valide y pueda crear un nuevo adelanto.

**Curso de los eventos:**

Tabla 3.68. Caso de uso - Crear adelanto

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para crear un nuevo adelanto. | |
| | 2.- El sistema redirige a la Secretaria a un formulario para crear un nuevo adelanto. |
| 3.- La Secretaria ingresa los datos del nuevo adelanto. | |
| | 4.- El sistema valida los datos ingresados. |
| | 5.- El sistema crea el nuevo adelanto. |

Fuente: Elaboración propia

**Curso alterno:**

4.1.- Los datos ingresados no son válidos.

**Caso de uso:** Modificar adelanto.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria modificar los datos de un adelanto.

**Resumen:** La Secretaria selecciona un adelanto y modifica sus datos, el sistema los valida y los actualiza.

**Curso de los eventos:**

Tabla 3.69. Caso de uso - Modificar adelanto

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para modificar adelanto. | |
| | 2.- El sistema redirige a la Secretaria a un formulario para modificar el adelanto seleccionado. |
| 3.- La Secretaria ingresa los nuevos datos del adelanto seleccionado. | |
| | 4.- El sistema valida los datos ingresados. |
| | 5.- El sistema actualiza los datos del adelanto seleccionado. |

Fuente: Elaboración propia

**Curso alterno:**

4.1.- Los datos ingresados no son válidos.

**Caso de uso:** Eliminar adelanto.

**Actores:** Secretaria.

**Propósito:** Permitir a la Secretaria eliminar un adelanto.

**Resumen:** La Secretaria selecciona un adelanto y lo elimina.

**Curso de los eventos:**

Tabla 3.70. Caso de uso - Eliminar adelanto

| Actores | Sistema |
|---|---|
| 1.- La Secretaria presiona el botón para eliminar adelanto. | |
| | 2.- El sistema presenta a la Secretaria un mensaje de confirmación. |
| 3.- La Secretaria confirma la acción realizada. | |
| | 4.- El sistema elimina el adelanto seleccionado. |
| | 5.- El sistema actualiza la tabla de datos de la pantalla de gestión de adelantos de la aplicación. |

Fuente: Elaboración propia

**Curso alterno:**

3.1.- La Secretaria no confirma la acción realizada.

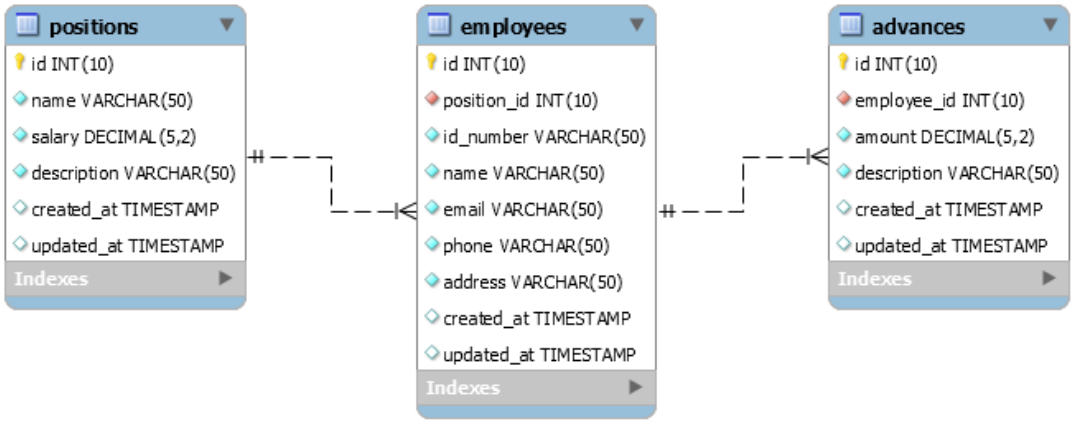
4.1.- El sistema no elimina el adelanto seleccionado.

### 3.3.1.2.Diagrama de datos

Para el desarrollo del módulo de adelantos de los trabajadores se construyeron tres (3) tablas las mismas que almacenan la siguiente información: la tabla *positions* almacena los cargos que ocupan los trabajadores de la Institución, la tabla *employees* almacena la información más relevante perteneciente a cada trabajador además de almacenar una clave foránea que contiene el identificador del cargo asignado a este

Finalmente, la tabla *advances* almacena los adelantos de cada trabajador, la relación entre la tabla de los trabajadores y la tabla de los adelantos es 1:n ya que un trabajador puede solicitar más de un adelanto económico, la tabla de los adelantos contiene información sobre el monto de cada adelanto más una breve descripción de este.

Ilustración 3.37. Diagrama de datos del módulo de adelantos




Fuente: Elaboración propia

De esta forma concluye el desarrollo de la aplicación en cuanto a la construcción de tablas de datos se refiere, para el producto final se migraron un total de veintiún (21) tablas con sus respectivas relaciones las cuales se documentaron a lo largo del desarrollo del proyecto planteado en este documento.

### 3.3.1.3.Diccionario de datos

Descripción y estructura de la tabla *advances*.

Tabla 3.71. Tabla advances

| Nombre | Tipo | Razón |
|---|---|---|
| id | int(10) | id auto incremental y clave primaria de la tabla |
| employee_id | int(10) | clave principal y foránea de la tabla advances |
| amount | varchar(255) | monto correspondiente al adelanto del trabajador |
| description | varchar(255) | descripción del adelanto registrado |
| created_at | timestamp | fecha de creación del registro |
| updated_at | timestamp | fecha de actualización del registro |

Fuente: Elaboración propia

Descripción y estructura de la tabla *employees*.

Tabla 3.72. Tabla employees

| Nombre | Tipo | Razón |
|---|---|---|
| id | int(10) | id auto incremental y clave primaria de la tabla |
| position_id | int(10) | clave principal y foránea de la tabla positions |
| id_number | varchar(255) | número de cédula del trabajador |
| name | varchar(255) | nombre completo del trabajador |
| email | varchar(255) | dirección de correo electrónico del trabajador |
| phone | varchar(255) | número de teléfono del trabajador |
| address | varchar(255) | dirección del trabajador |

Fuente: Elaboración propia

Descripción y estructura de la tabla *positions*.

Tabla 3.73. Tabla positions

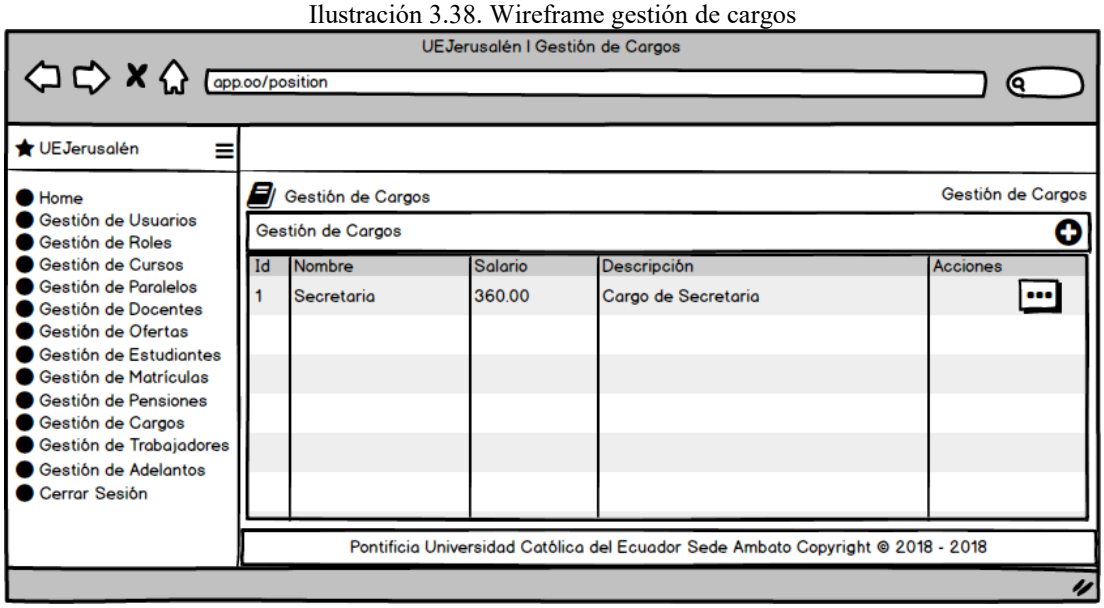
| Nombre | Tipo | Razón |
|---|---|---|
| id | int(10) | id auto incremental y clave primaria de la tabla |
| name | varchar(255) | nombre del paralelo |
| salary | varchar(255) | monto correspondiente al salario del cargo |
| description | varchar(255) | breve descripción del cargo registrado |
| created_at | timestamp | fecha de creación del registro |
| updated_at | timestamp | fecha de actualización del registro |

Fuente: Elaboración propia

### 3.3.1.4.Diseño

Diseño de la interfaz de usuario para la gestión de cargos del módulo de adelantos.

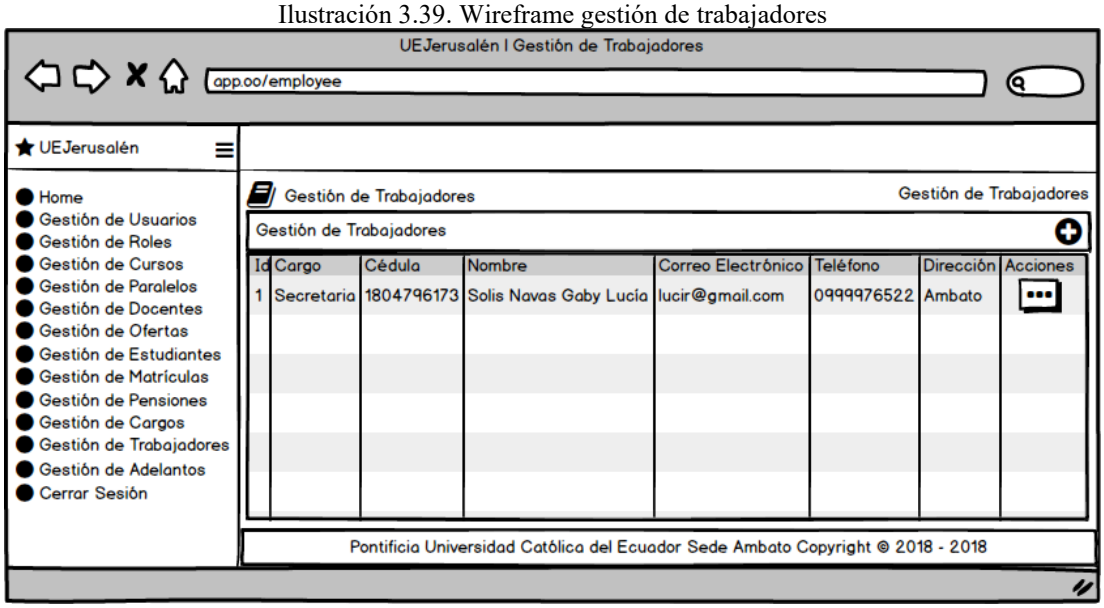
Ilustración 3.38. Wireframe gestión de cargos




Fuente: Elaboración propia

Diseño de la interfaz de usuario para la gestión de trabajadores del mismo módulo.

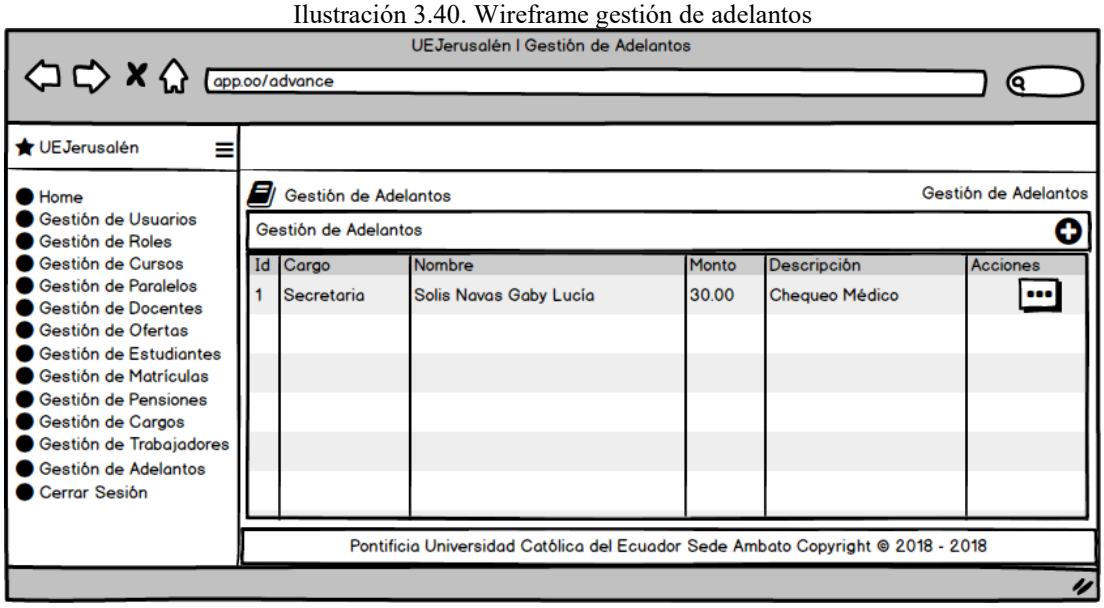
Ilustración 3.39. Wireframe gestión de trabajadores




Fuente: Elaboración propia

Diseño de la interfaz de usuario para la gestión de adelantos del mismo módulo.

Ilustración 3.40. Wireframe gestión de adelantos




Fuente: Elaboración propia

### 3.3.2.Colaboración

#### 3.3.2.1.Programación

En esta iteración se empieza a desarrollar el módulo de adelantos de los trabajadores de la Institución educativa, de acuerdo con los requerimientos del cliente se necesita que los adelantos sean registrados por la aplicación en cualquier momento y se pueda imprimir un comprobante para que el trabajador firme dicho comprobante y exista una constancia de que se realizó un adelanto económico, para lograrlo se debe crear un cargo, el archivo *PositionController.php* el método para crear un cargo es el siguiente:

Ilustración 3.41. Método para crear un cargo

```
public function store(Request $request)
{
    $this->validate($request, [
        'name' => 'required|string|max:255|unique:positions',
        'salary' => 'required|regex:/^\d*(\.\d{1,2})?$/',
        'description' => 'required|string|max:255'
    ]);

    (new Position())->create([
        'name' => $request['name'],
        'salary' => $request['salary'],
        'description' => $request['description']
    ]);

    flash( message: 'Cargo Creado.')->success();

    return redirect()->route( route: 'position.index');
}
```

Fuente: Elaboración propia

Posteriormente en el archivo *PositionTableController.php* se obtienen todos los registros de la tabla de cargos con la siguiente función y el siguiente código:

Ilustración 3.42. Método para cargar la tabla de cargos

```
public function __invoke()
{
    $query = Position::query();

    return Datatables::eloquent($query)
        ->editColumn( name: 'action', function ($position) {
            return $position->action_buttons;
        })
        ->make( mDataSupport: true);
}
```

Fuente: Elaboración propia

Una vez creado el registro y cargada la tabla se deben definir las rutas del módulo de adelantos, estas se componen por las rutas de la gestión de cargos, empleados y adelantos, cada grupo tiene dos rutas: la primera es para obtener los datos del modelo mientras que la segunda es para gestionar los recursos del modelo, estas son similares por lo que a continuación se colocan todas las rutas del módulo de adelantos:

Ilustración 3.43. Rutas para el módulo de adelantos

```
Route::get( uri: 'position/data', action: 'School\PositionTableController')->name( name: 'position.data');
Route::resource( name: 'position', controller: 'School\PositionController');

Route::get( uri: 'employee/data', action: 'School\EmployeeTableController')->name( name: 'employee.data');
Route::resource( name: 'employee', controller: 'School\EmployeeController');

Route::get( uri: 'advance/data', action: 'School\AdvanceTableController')->name( name: 'advance.data');
Route::get( uri: 'advance/{advance}/invoice', action: 'School\AdvanceInvoiceController')->name( name: 'advance.invoice');
Route::resource( name: 'advance', controller: 'School\AdvanceController');
```

Fuente: Elaboración propia

Con las rutas establecidas en el archivo *school/positions/index.blade.php* se coloca una tabla dinámica de datos que tiene la siguiente estructura y permite realizar consultas en donde el termino ingresado es filtrado inmediatamente por la función y devuelve los resultados similares al termino de búsqueda ingresado, de la siguiente forma:

Ilustración 3.44. Función para cargar la tabla de cargos

```
@push('styles')
    {{ Html::style('https://cdn.datatables.net/1.10.18/css/dataTables.bootstrap.css') }}
@endpush

@push('scripts')
    {{ Html::script('https://cdn.datatables.net/1.10.18/js/jquery.dataTables.js') }}
    {{ Html::script('https://cdn.datatables.net/1.10.18/js/dataTables.bootstrap.js') }}
    <script>
        $(function () {
            let positions_table = $('#positions_table');
            positions_table.DataTable({
                processing: true,
                serverSide: true,
                searchDelay: 500,
                ajax: {
                    url: '{{ route('position.data') }}'
                },
                columns: [
                    {data: 'id', name: 'positions.id', searchable: false},
                    {data: 'name', name: 'positions.name'},
                    {data: 'salary', name: 'positions.salary'},
                    {data: 'description', name: 'positions.description'},
                    {data: 'action', name: 'action', searchable: false, sortable: false}
                ]
            });
        });
    </script>
@endpush
```

Fuente: Elaboración propia

Para crear un trabajador se le debe asignar un cargo, la dinámica para crear un trabajador es parecida a la anterior, sin embargo, en la tabla de trabajadores se debe establecer un campo que se relacione con el cargo, en el archivo *create_employees_table.php* se coloca dicho campo de la siguiente manera:

Ilustración 3.45. Datos de la tabla de empleados

```
public function up()
{
    Schema::create('employees', function (Blueprint $table) {
        $table->increments( column: 'id');
        $table->integer( column: 'position_id')->unsigned();
        $table->string( column: 'id_number');
        $table->string( column: 'name');
        $table->string( column: 'email');
        $table->string( column: 'phone');
        $table->string( column: 'address');
        $table->timestamps();

        $table->foreign( columns: 'position_id')
            ->references( column: 'id')
            ->on( table: 'positions');
    });
}
```

Fuente: Elaboración propia

En el archivo *Employee.php* las relaciones con los modelos de cargos y adelantos son:

Ilustración 3.46. Relacions del módelo de rabajadores

```
public function position()
{
    return $this->belongsTo( related: 'App\Position');
}

public function advances()
{
    return $this->hasMany( related: 'App\Advance');
}
```

Fuente: Elaboración propia

De esta forma se puede crear un trabajador con un cargo con la siguiente función ubicada en el archivo *EmployeeController.php:*

Ilustración 3.47 . Método para crear un trabajador

```
    (new Employee())->create([
        'position_id' => $request['position_id'],
        'id_number' => $request['id_number'],
        'name' => $request['name'],
        'email' => $request['email'],
        'phone' => $request['phone'],
        'address' => $request['address']
    ]);

    flash( message: 'Trabajador Creado.')->success();

    return redirect()->route( route: 'employee.index');
```

Fuente: Elaboración propia

Finalmente, la gestión de adelantos consiste en seleccionar un trabajador y registrar un

adelanto junto con el monto de este y una breve descripción, para poder crear un adelanto se debe seleccionar un trabajador en la vista de crear adelanto y para lograrlo en el archivo *AdvanceController.php* primero se debe colocar el modelo de trabajadores y seleccionar de cada uno de ellos la columna de nombre de esta manera:

Ilustración 3.48. Función para crear un adelanto en la vista

```
public function create()
{
    if (Auth::check() && Auth::user()->hasPermission( permission: 'create-advances')) {
        $employees = Employee::query()->pluck( column: 'name', key: 'id');
        return view( view: 'school.advances.create', compact( varname: 'employees'));
    }

    return abort( code: 403);
}
```

Fuente: Elaboración propia

Con esto se cargan todos los trabajadores por su nombre en la vista de crear empleado y en el mismo archivo el método para registrar un adelanto es el siguiente:

Ilustración 3.49. Mótodo para registrar un adelanto

```
public function store(Request $request)
{
    $this->validate($request, [
        'employee_id' => 'required',
        'amount' => 'required|regex:/^\d*(\.\d{1,2})?$/',
        'description' => 'required|string|max:255'
    ]);

    (new Advance())->create([
        'employee_id' => $request['employee_id'],
        'amount' => $request['amount'],
        'description' => $request['description']
    ]);

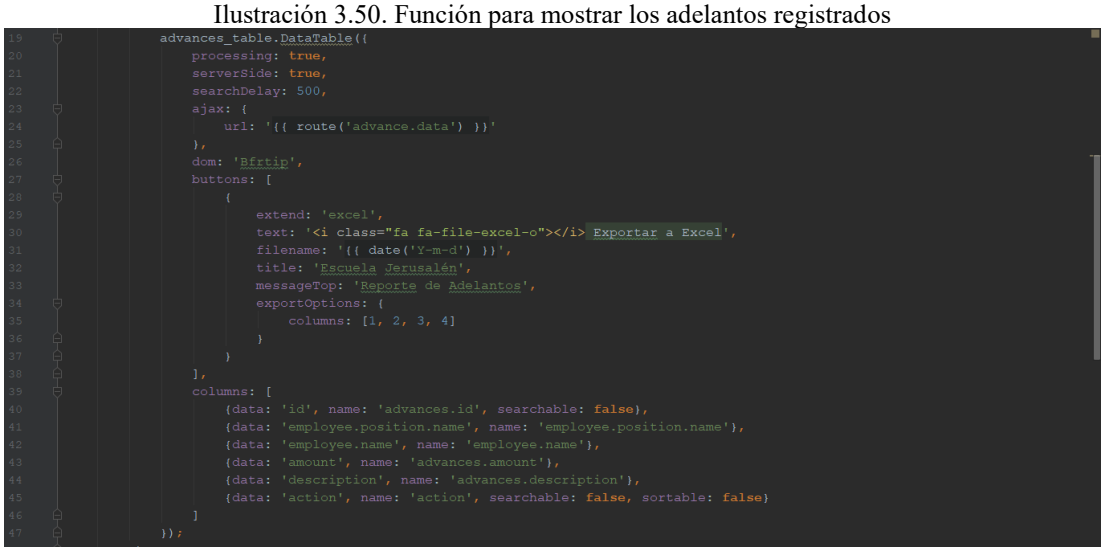
    flash( message: 'Adelanto Creado.')->success();

    return redirect()->route( route: 'advance.index');
}
```

Fuente: Elaboración propia

Es parte también de los requerimientos del cliente el poder exportar un reporte de todos los adelantos que ha realizado un determinado trabajador en especial, en el archivo *school/advances/index.blade.php* se coloca una tabla de datos que funciona con una ruta en donde se cargan las relaciones que tienen los adelantos, la siguiente función corresponde al código que permite visualizar y consultar los adelantos realizados:

Ilustración 3.50. Función para mostrar los adelantos registrados



```
advances_table.DataTable({
    processing: true,
    serverSide: true,
    searchDelay: 500,
    ajax: {
        url: '{{ route('advance.data') }}'
    },
    dom: 'Bfrtip',
    buttons: [
        {
            extend: 'excel',
            text: '<i class="fa fa-file-excel-o"></i> Exportar a Excel',
            filename: '{{ date('Y-m-d') }}',
            title: 'Escuela Jerusalén',
            messageTop: 'Reporte de Adelantos',
            exportOptions: {
                columns: [1, 2, 3, 4]
            }
        }
    ],
    columns: [
        {data: 'id', name: 'advances.id', searchable: false},
        {data: 'employee.position.name', name: 'employee.position.name'},
        {data: 'employee.name', name: 'employee.name'},
        {data: 'amount', name: 'advances.amount'},
        {data: 'description', name: 'advances.description'},
        {data: 'action', name: 'action', searchable: false, sortable: false}
    ]
});
```

Fuente: Elaboración propia

El archivo generado es exportado como una hoja de Excel que contiene los datos que se obtuvo por medio de la consulta, como se puede evidenciar existe una ruta que se encarga de obtener los datos, dicha ruta se encuentra enlazada con la siguiente función:

Ilustración 3.51. Función para la tabla de datos de los adelantos



```
public function __invoke()
{
    $query = Advance::query()->with( relations: 'employee.position');

    return Datatables::eloquent($query)
        ->editColumn( name: 'position.name', function ($advance) {
            return $advance->employee->position->name;
        })
        ->editColumn( name: 'employee.name', function ($advance) {
            return $advance->employee->name;
        })
        ->editColumn( name: 'action', function ($employee) {
            return $employee->action_buttons;
        })
        ->make( mDataSupport: true);
}
```

Fuente: Elaboración propia

Con lo anterior queda concluido el módulo de adelantos de los trabajadores, en síntesis, dicho módulo permite al usuario crear un cargo, crear un trabajador y asignarle un cargo y posteriormente registrar uno o varios adelantos para un trabajador con la opción de imprimir un comprobante para el registro creado y la opción de obtener un reporte de los adelantos registrados por cada trabajador de la Institución educativa, de esta forma se tiene un registro físico y digital de los adelantos que se han solicitado.

De acuerdo con los requerimientos del cliente se necesitaba también un portal web informativo ya que la Institución no contaba con un lugar en donde colocar información sobre los servicios que se ofrecen a los estudiantes o por ejemplo cual es la misión y la visión de la Escuela, por lo tanto, se estableció que se desplegaría una página web que pueda albergar este tipo de información para el padre de familia.

El desarrollo de la página web empieza por determinar qué tipo de información se va a mostrar en la misma, si la calidad de la información es estática o dinámica respectivamente, en este caso la información es estática lo que significa que la página web no va a interactuar con el usuario, a continuación, el siguiente paso consiste en recolectar la información de fuentes confiables para no colocar información falsa.

Una vez que se ha determinado que tipo de página web es la que se va a desarrollar y también se cuenta con la información necesaria para realizarlo se procede a crear un archivo que funcione como plantilla para el resto de los archivos, el archivo *landing.blade.php* contiene una sección que permite al motor de plantillas de *Laravel* reconocer dicha sección y permitir que el resto de los componentes se compilen.

Ilustración 3.52. Plantilla para el desarrollo de la página web

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>{{ config('app.name') }} | @yield('title')</title>
    <link rel="stylesheet" href="{{ asset('https://fonts.googleapis.com/css?family=Raleway:300,400,500,700|Open+Sans') }}">
    <link rel="stylesheet" href="{{ asset('css/styles-merged.css') }}">
    <link rel="stylesheet" href="{{ asset('css/style.css') }}">
    <link rel="stylesheet" href="{{ asset('css/custom.css') }}">
    <link rel="icon" href="{{ asset('favicon.png') }}">
</head>
<body>
<div class="probootstrap-page-wrapper">
    @include('partials.landing.header-top')
    @include('partials.landing.navbar')
    @yield('content')
    @include('partials.landing.footer')
</div>
<script src="{{ asset('js/scripts.min.js') }}"></script>
<script src="{{ asset('js/main.min.js') }}"></script>
<script src="{{ asset('js/custom.js') }}"></script>
</body>
</html>
```

Fuente: Elaboración propia

Debido a que la página web es extensa se puede separar a esta en archivos pequeños en donde cada archivo es una sección y se puede integrar a la vista maestra.

Ilustración 3.53. Sección de cabecera de la plantilla

```
<div class="probootstrap-header-top">
    <div class="container">
        <div class="row">
            <div class="col-lg-9 col-md-9 col-sm-9 probootstrap-top-quick-contact-info">
                <span><i class="icon-location2"></i>Pichincha Alta y José Mires</span>
                <span><i class="icon-phone2"></i>032 - 41-28-54</span>
                <span><i class="icon-mail"></i>swjerusalen@gmail.com</span>
            </div>
            <div class="col-lg-3 col-md-3 col-sm-3 probootstrap-top-social">
                <ul>
                    <li><a href="https://www.facebook.com/Escuela-Jerusal%C3%A9n-896674410379090/" target="_blank"><i class="icon
                </ul>
            </div>
        </div>
    </div>
</div>
```

Fuente: Elaboración propia

La sección de cabecera contiene los datos de contacto de la Institución, esta vista es muy útil ya que si la Institución dispone de cuentas en redes sociales se pueden colocar enlaces a dichas cuentas para que el usuario pueda conectarse en distintos niveles.

A continuación, la barra de navegación contiene los elementos que conforman el menú de opciones para el usuario, en este menú se aloja el enlace para acceder a la aplicación web de la Institución, dicho enlace redirige al usuario a la pantalla de iniciar sesión.

Ilustración 3.54. Barra de navegación de la plantilla

```
<nav class="navbar navbar-default probootstrap-navbar">
    <div class="container">
        <div class="navbar-header">
            <div class="btn-more js-btn-more visible-xs">
                <a href="#"><i class="icon-dots-three-vertical "></i></a>
            </div>
            <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar-collapse" aria-expa
                <span class="sr-only">Alternar Navegación</span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand" href="{{ url('/') }}"></a>
        </div>
        <div id="navbar-collapse" class="navbar-collapse collapse">
            <ul class="nav navbar-nav navbar-right">
                <li class="{{ active_class(Active::checkUri('/')) }}"><a href="{{ url('/') }}">Inicio</a></li>
                <li class="{{ active_class(Active::checkUri('about')) }}"><a href="{{ route('about') }}">Quienes Somos</a></li>
                <li class="{{ active_class(Active::checkUri('teachers')) }}"><a href="{{ route('teachers') }}">Docentes</a></li>
                <li class="{{ active_class(Active::checkUri('services')) }}"><a href="{{ route('services') }}">Servicios</a></li
                <li><a href="{{ route('login') }}">Iniciar Sesión</a></li>
            </ul>
        </div>
    </div>
</nav>
```

Fuente: Elaboración propia

El ultimo componente de la plantilla es el pie de página, este componente contiene un resumen de los enlaces disponibles junto con la información de la Institución, además se pone a conocimiento del usuario que el proyecto es propiedad de la Universidad.

Ilustración 3.55. Pié de página de la plantilla

```
<footer class="probootstrap-footer probootstrap-bg">
    <div class="container">
        <div class="row">
            <div class="col-md-6">
                <div class="probootstrap-footer-widget">
                    <h3>Enlaces</h3>
                    <ul>
                        <li><a href="{{ url('/') }}">Inicio</a> |
                            <a href="{{ route('about') }}">Quienes Somos</a> |
                            <a href="{{ route('teachers') }}">Docentes</a> |
                            <a href="{{ route('services') }}">Servicios</a>
                        </li>
                    </ul>
                </div>
            </div>
            <div class="col-md-4 col-md-push-2">
                <div class="probootstrap-footer-widget">
                    <h3>Información de Contacto</h3>
                    <ul class="probootstrap-contact-info">
                        <li><i class="icon-location2"></i><span>Pichincha Alta y José Mires</span></li>
                        <li><i class="icon-mail"></i><span>swjerusalen@gmail.com</span></li>
                        <li><i class="icon-phone2"></i><span>032 - 41-28-54</span></li>
                    </ul>
                </div>
            </div>
        </div>
        <p class="text-center">Pontificia Universidad Católica Del Ecuador Sede Ambato Copyright © 2018 - {{ date('Y') }}</p>
    </div>
</footer>
```

Fuente: Elaboración propia

En la página de bienvenida se colocan la misión y la visión a las cuales se rige el Plantel Educativo, además se coloca mediante variables provistas desde el archivo *LandingController.php* el número de estudiantes matriculados, el número de docentes capacitados y el número de ofertas académicas disponibles para los estudiantes.

Ilustración 3.56. Función para contar estudiantes, docentes y ofertas

```
    public function index()
    {
        $studentsCount = Student::query()->count();
        $teachersCount = Teacher::query()->count();
        $offersCount = Offer::query()->count();
        return view( view: 'welcome', compact( varname: 'studentsCount', 'teachersCount', 'offersCount'));
    }
```

Fuente: Elaboración propia

De esta forma se coloca la información correspondiente en cada página del sitio web, en esta iteración tampoco se realizó ninguna reunión de colaboración con el cliente ya que toda la información a publicar fue entregada en su debido tiempo

### 3.3.3.Aprendizaje

#### 3.3.3.1.Control de calidad

Para realizar el control de calidad del módulo desarrollado se programó una reunión con la Directora y la Secretaria de la Institución para probar la aplicación, el respectivo control de calidad se documentó con la siguiente prueba de aceptación la misma que contiene todos los requerimientos obtenidos del módulo y las respuestas del usuario.

Tabla 3.74. Prueba de aceptación de módulo de adelantos

| Prueba de aceptación del módulo de adelantos | | |
|---|---|---|
| Título: Revisión del módulo de adelantos | | |
| Fecha: 21-06-2018 | | |
| Usuario: Fanny Hernández & Karla Chicaiza | | |
| Requerimiento | Si | No |
| Puede crear, modificar y eliminar cargos | X | |
| Puede crear, modificar y eliminar trabajadores | X | |
| Puede asignar y reasignar cargos a los trabajadores | X | |
| Puede crear, modificar y eliminar adelantos | X | |
| Puede registrar adelantos económicos a los trabajadores | X | |
| Puede consultar los adelantos por el nombre del trabajador | X | |
| Puede exportar un reporte de los adelantos de un trabajador en una hoja de Excel | X | |
| Puede navegar de manera intuitiva por las funciones del módulo entregado | X | |
| Está satisfecho con el diseño de la página web | X | |

Fuente: Elaboración propia

### 3.3.3.2.Implementación

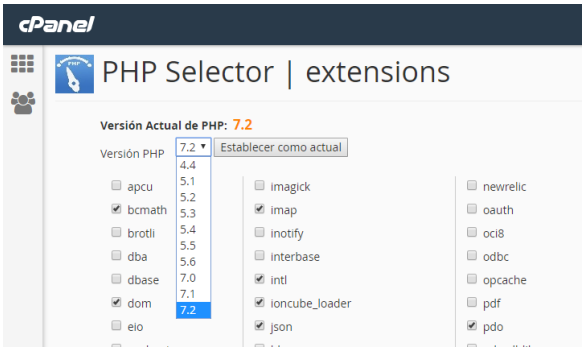
Dentro del DAS se incluye la etapa de implementación en la fase de aprendizaje cuando se ha concluido el desarrollo de la aplicación y tanto las pruebas de aceptación como los requerimientos del cliente han sido satisfactorias y se han cumplido todos respectivamente, en cuanto a la implementación y al *Framework* se refiere, *Laravel* en como herramienta requiere que el servidor cuente con las siguientes características:

- *Laravel* en su versión 5.4. requiere una versión de PHP mayor o igual a la versión 5.6.4.

- Una extensión para que el protocolo de puertos seguros (SSL por sus siglas en inglés) funcione correctamente en PHP.

Para implementar la aplicación en *Cpanel* es necesario iniciar sesión, una vez hecho esto el sitio web se redirige al panel de administración y en este se busca la opción "Seleccionar Versión PHP" y se procede a acceder al enlace para lo siguiente:

Ilustración 3.57. Seleccionar versión de PHP

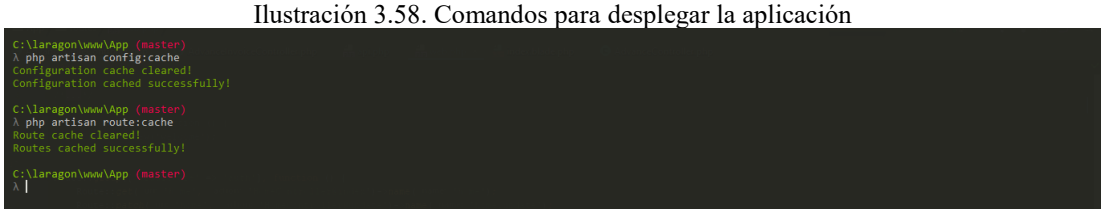



Fuente: Elaboración propia

Como se puede ver en la imagen anterior la versión actual de PHP del servidor es la más actual que es la 7.2 por lo tanto no existió la necesidad de cambiar la versión, una vez elegida la versión que se necesite se guardan los cambios y se regresa al panel.

Para que la aplicación funcione de mejor manera y para mejorar el rendimiento de esta se procede a ejecutar varios comandos propios del *Framework* que permiten guardar en memoria *cache* las rutas y las configuraciones de la siguiente manera:

Ilustración 3.58. Comandos para desplegar la aplicación

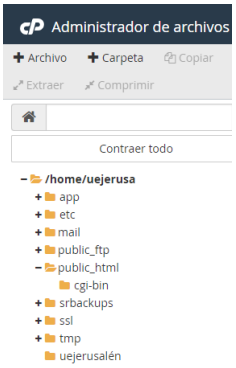



Fuente: Elaboración propia

De esta forma se compilan todas las configuraciones en un solo archivo para que la aplicación optimize su rendimiento, lo mismo sucede con las rutas de la aplicación.

A continuación, en el panel de administración se busca la opción “Administrador de archivos” y se accede al enlace, se crea una carpeta y se coloca el nombre del proyecto:

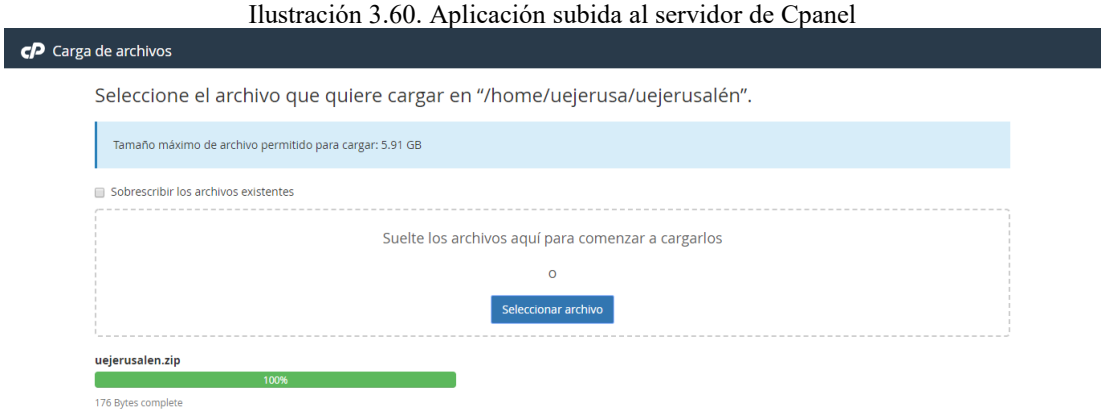
Ilustración 3.59. Administrador de archivos de Cpanel




Fuente: Elaboración propia

De regreso a la ubicación del proyecto desarrollado, se procede a comprimir toda la aplicación en un archivo comprimido, posteriormente se sube el archivo comprimido a la carpeta creada en el administrador de archivos de *Cpanel* de la siguiente manera:

Ilustración 3.60. Aplicación subida al servidor de Cpanel




Fuente: Elaboración propia

Finalmente se extraen los archivos de la aplicación en la carpeta creada en el servidor, una vez realizado lo anterior ya es posible acceder a la aplicación en el navegador web, con eso se da por terminada la implementación, la tercera iteración y la aplicación.

En síntesis, la implementación consistió en contratar un servidor y cargar los archivos de lo aplicación en el servidor, también fue necesario adquirir un nombre de dominio para redireccionar al servidor a dicho nombre de dominio, en el presente proyecto el dominio es *uejerusalen.edu.ec* y se encuentra libre para el acceso.

# CAPÍTULO IV

# ANÁLISIS Y VALIDACIÓN DE RESULTADOS

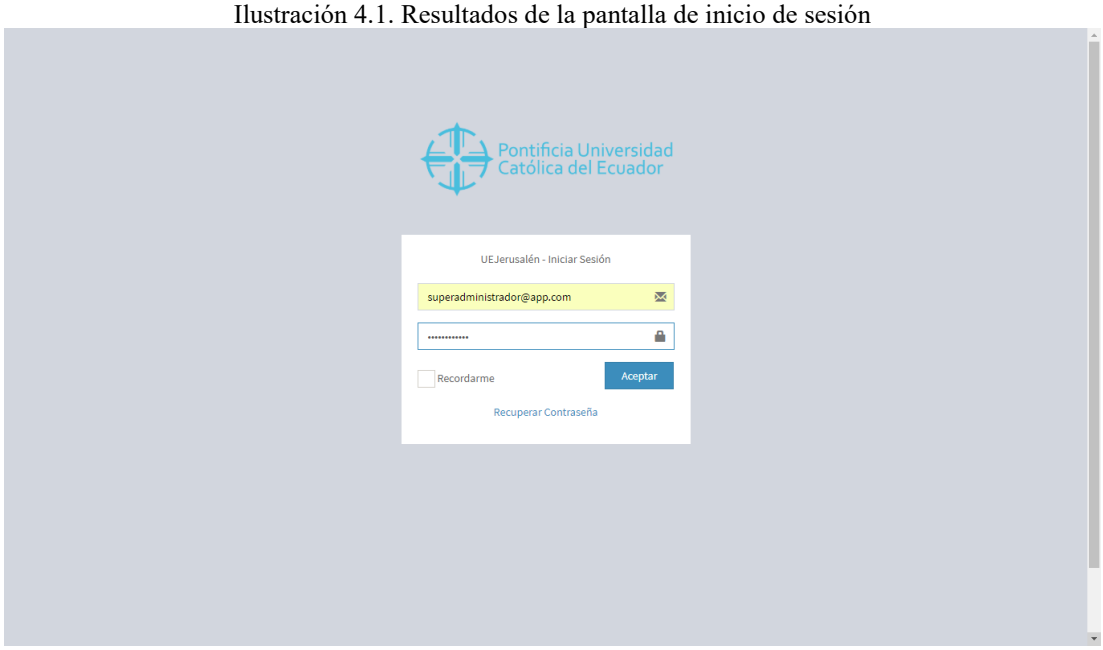
## 4.1.Análisis de resultados

En esta sección se muestran los resultados de cada módulo desarrollado y de las funciones principales que la aplicación tiene para ofrecer a los usuarios finales de esta.

- **Resultados del módulo de usuarios**

En el módulo de usuarios se puede encontrar la gestión de usuarios y la gestión de roles de la aplicación, para demostrar los resultados del proyecto desarrollado de manera correcta se va a iniciar sesión como usuario Super Administrador del sistema.

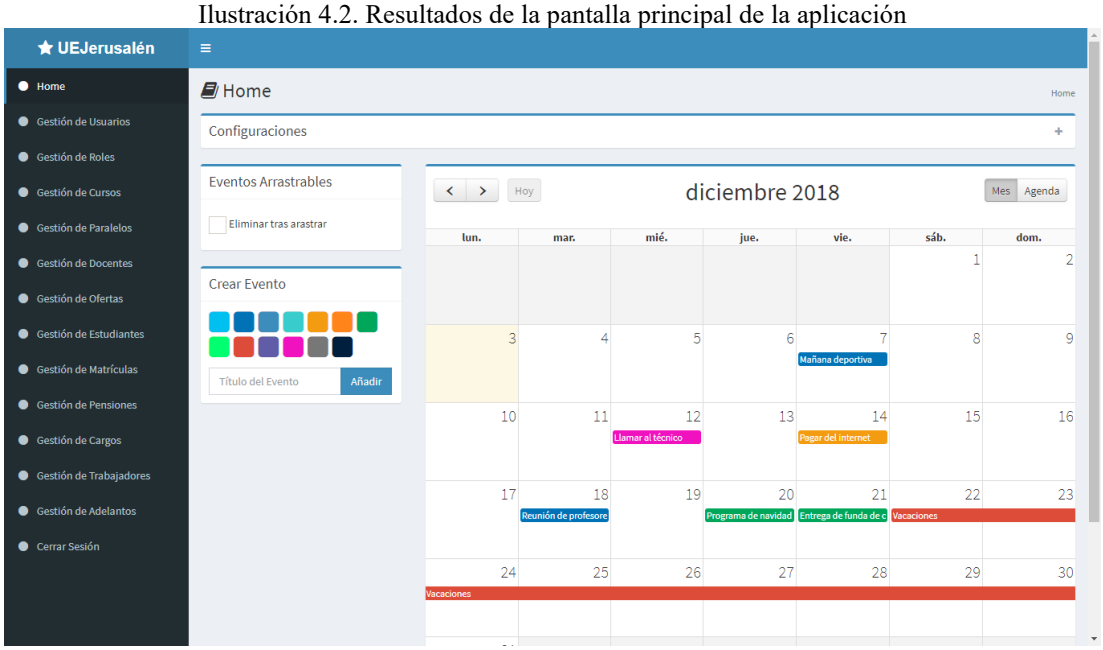
Ilustración 4.1. Resultados de la pantalla de inicio de sesión




Fuente: Elaboración propia

Cuando el usuario ha sido autenticado el sistema lo redirige a la pantalla principal.

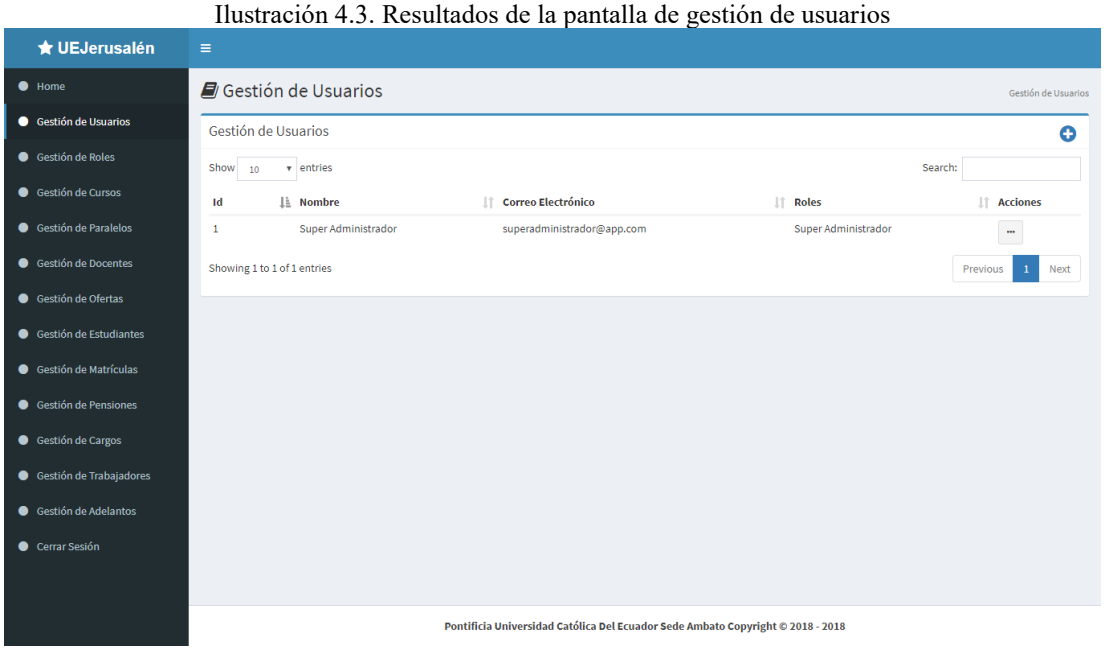
Ilustración 4.2. Resultados de la pantalla principal de la aplicación




Fuente: Elaboración propia

La pantalla de gestión de usuarios permite consultar a los usuarios que se encuentran registrados en la aplicación por medio de un término de búsqueda específico.

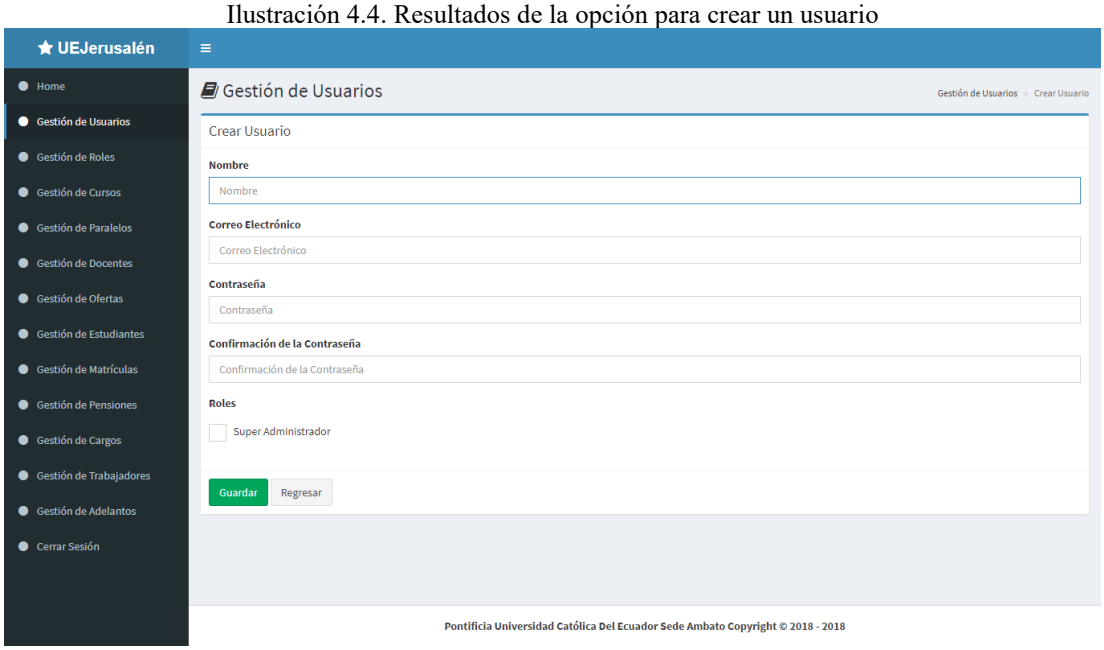
Ilustración 4.3. Resultados de la pantalla de gestión de usuarios




Fuente: Elaboración propia

La opción para crear un usuario permite asignarle un rol existente al usuario a crear.

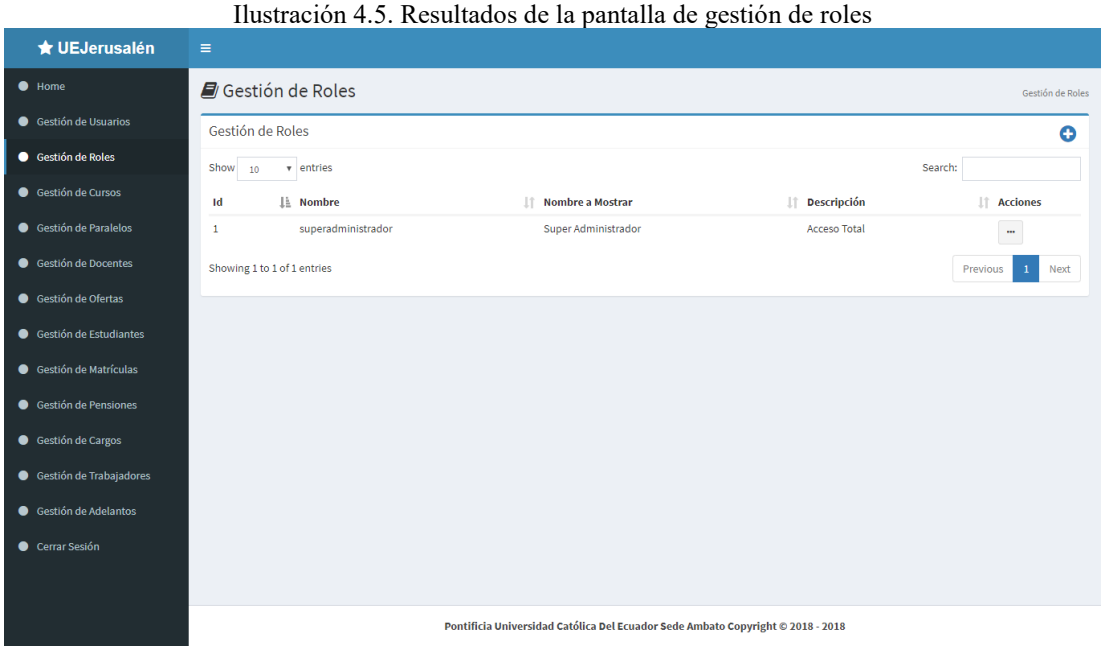
Ilustración 4.4. Resultados de la opción para crear un usuario




Fuente: Elaboración propia

La pantalla de gestión de roles permite crear un nuevo rol y asignarle permisos o a la vez modificar un rol existente y quitarle o reasignarle permisos según sea necesario.

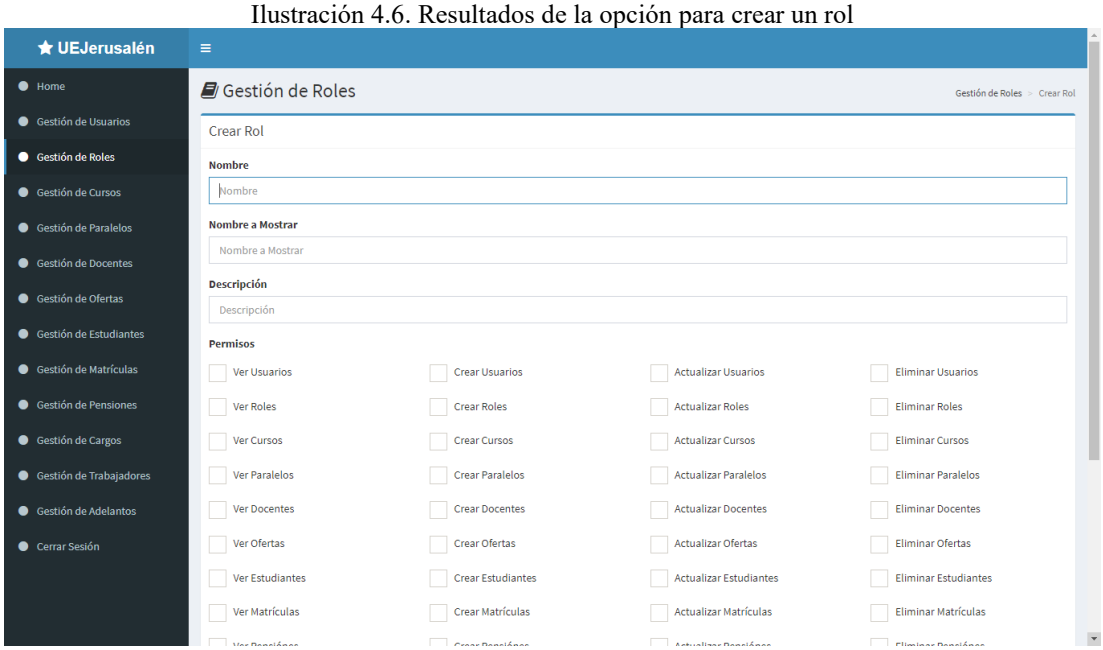
Ilustración 4.5. Resultados de la pantalla de gestión de roles




Fuente: Elaboración propia

Para crear un rol se muestran todos los permisos con los que cuenta la aplicación.

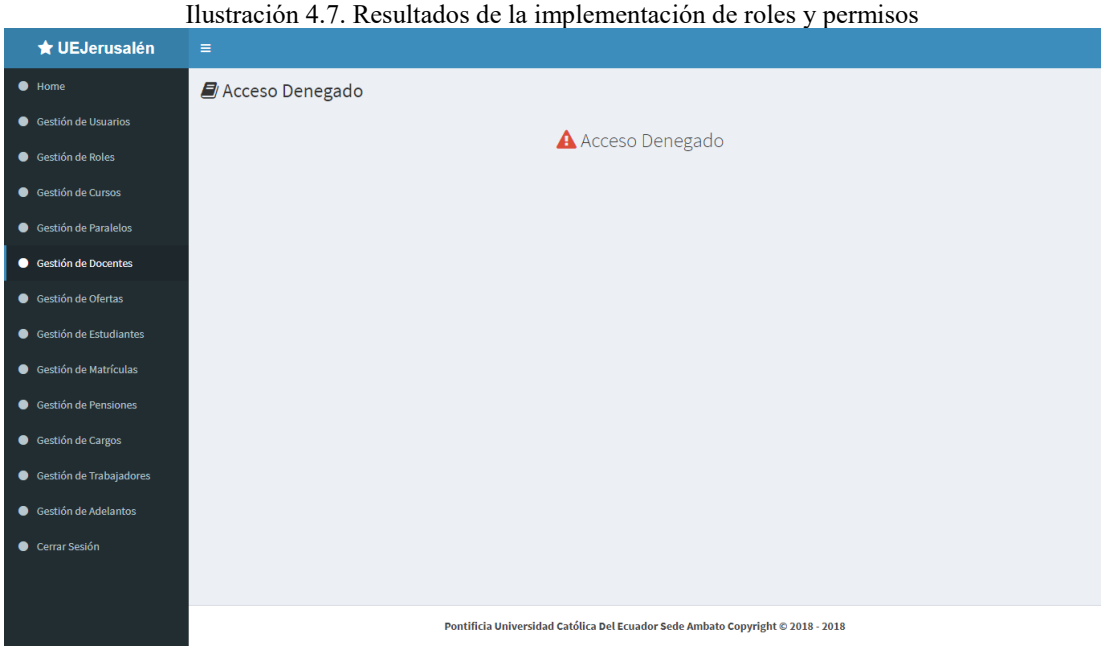
Ilustración 4.6. Resultados de la opción para crear un rol




Fuente: Elaboración propia

Cuando un usuario no cuenta con un permiso a un recurso, el enlace de este no se visualiza y en caso de forzar el ingreso a dicho recurso se muestra un mensaje de error.

Ilustración 4.7. Resultados de la implementación de roles y permisos

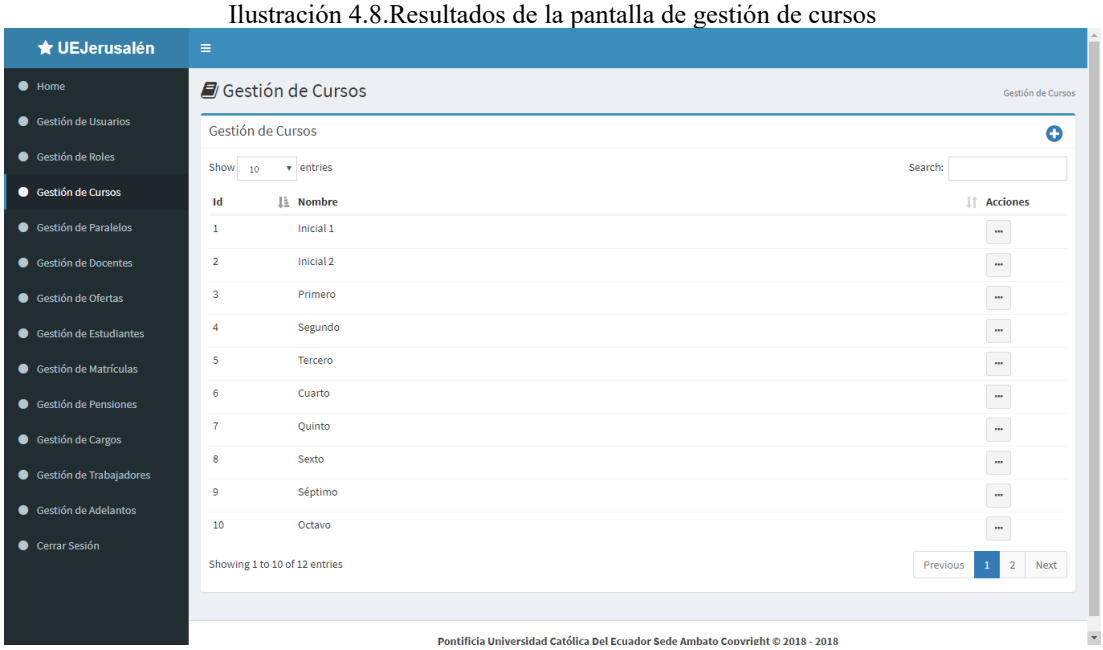



Fuente: Elaboración propia

- **Resultados del módulo de matrículas y pensiones**

La pantalla de gestión de cursos permite crear los distintos cursos de las ofertas.

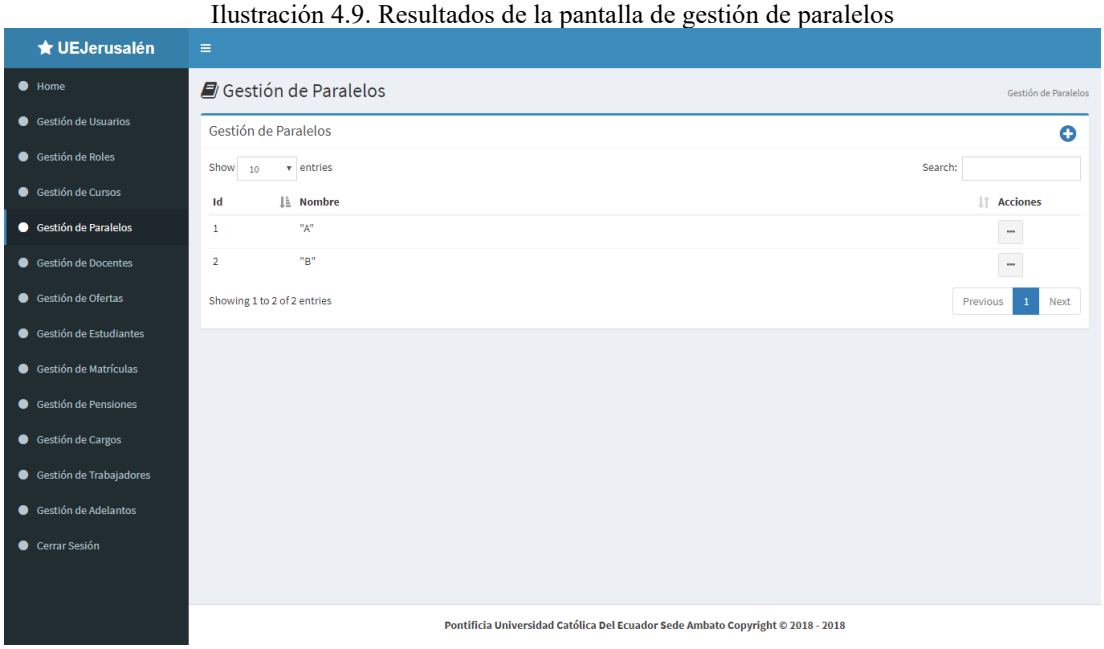
Ilustración 4.8.Resultados de la pantalla de gestión de cursos




Fuente: Elaboración propia

La pantalla de gestión de paralelos permite crear los distintos paralelos de las ofertas.

Ilustración 4.9. Resultados de la pantalla de gestión de paralelos




Fuente: Elaboración propia

La pantalla de gestión de docentes permite gestionar a los docentes de la Institución.

Ilustración 4.10. Resultados de la pantalla de gestión de docentes




Fuente: Elaboración propia

La pantalla de gestión de ofertas académicas permite consultar las ofertas creadas, crear una nueva oferta, modificar una oferta existente o eliminar una oferta creada.

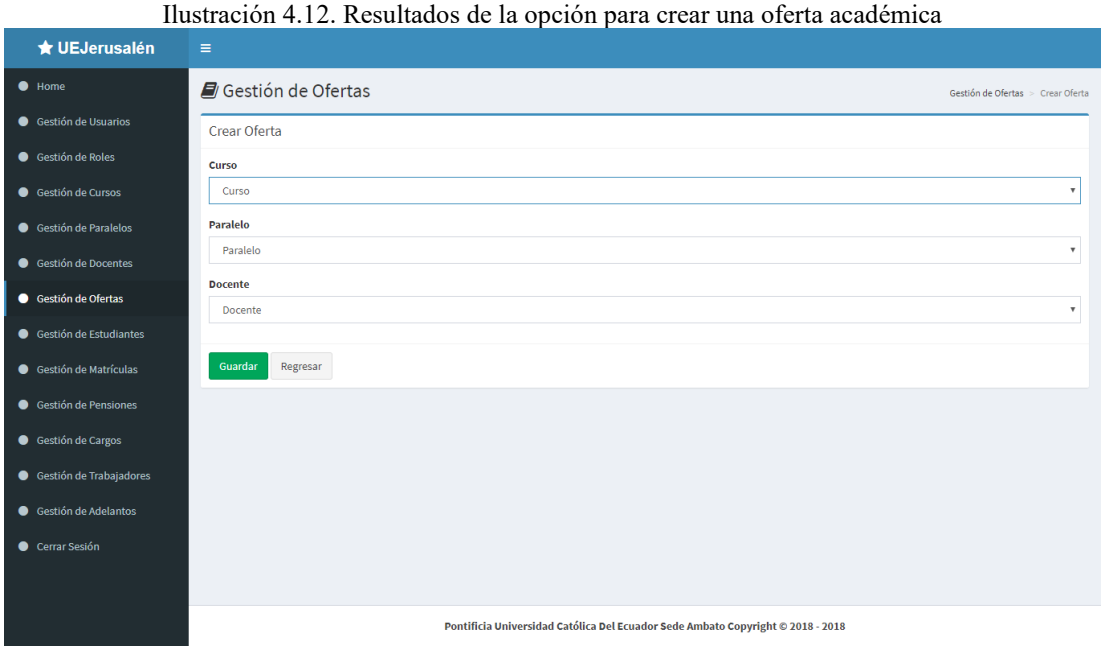
Ilustración 4.11. Resultados de la pantalla de gestión de ofertas




Fuente: Elaboración propia

La opción de crear una oferta permite seleccionar un curso, un paralelo y un docente.

Ilustración 4.12. Resultados de la opción para crear una oferta académica




Fuente: Elaboración propia

La pantalla de gestión de estudiantes permite consultar a los estudiantes registrados, crear un estudiante, modificar un estudiante o eliminar un estudiante seleccionado.

Ilustración 4.13. Resultados de la pantalla de gestión de estudiantes




Fuente: Elaboración propia

La pantalla de gestión de matrículas permite matricular a un estudiante en una oferta.

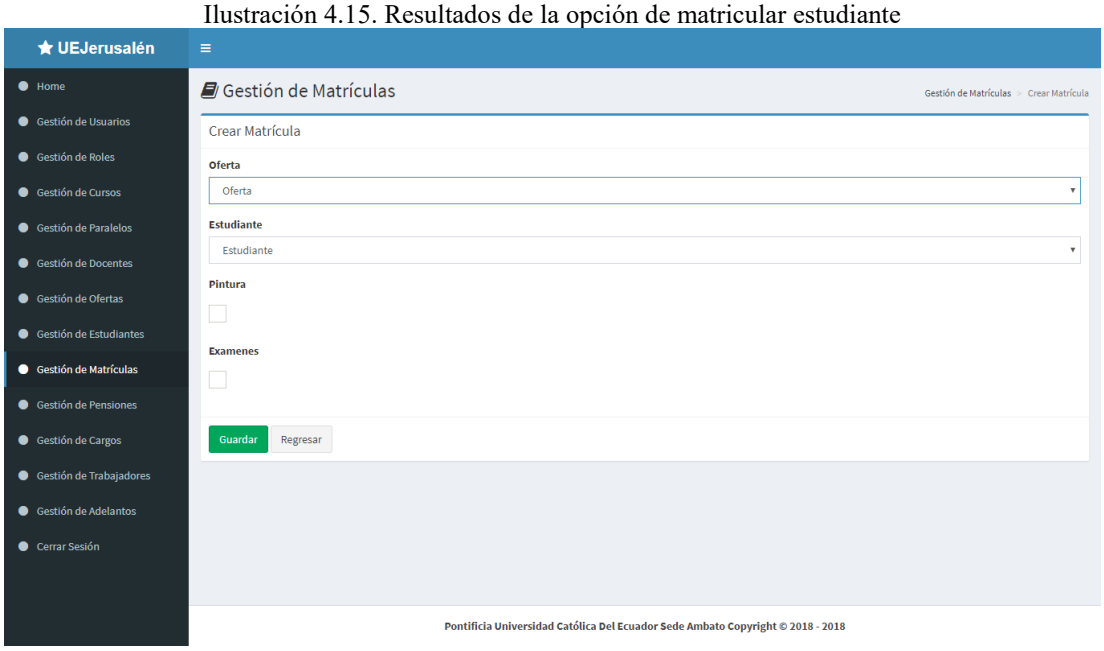
Ilustración 4.14. Resultados de la pantalla de gestión de matrículas




Fuente: Elaboración propia

La opción para crear matrícula consiste en seleccionar una oferta académica y un estudiante y a este añadirle valores correspondientes a la pintura y exámenes o copias.

Ilustración 4.15. Resultados de la opción de matricular estudiante




Fuente: Elaboración propia

La opción de consultar matrícula permite buscar un estudiante o una oferta académica.

Ilustración 4.16. Resultados de la opción de consultar matrícula




Fuente: Elaboración propia

La opción de reporte de estudiantes matriculados permite al usuario obtener un reporte de todos los estudiantes que han sido matriculados en una oferta académica especifica.

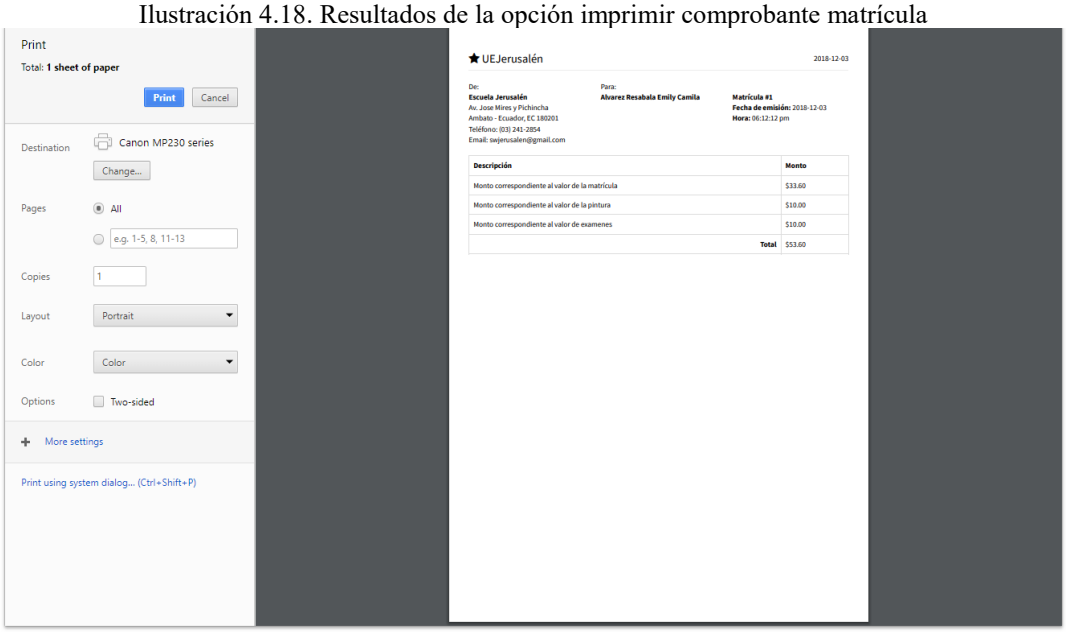
Ilustración 4.17. Resultados del reporte de estudiantes matriculados




Fuente: Elaboración propia

La opción de imprimir comprobante refleja los valores cancelados de la matrícula.

Ilustración 4.18. Resultados de la opción imprimir comprobante matrícula




Fuente: Elaboración propia

La pantalla de gestión de pensiones permite consultar las pensiones ya sea por estudiante, por oferta o por mes y crear una nueva pensión o ver una pensión ya creada.

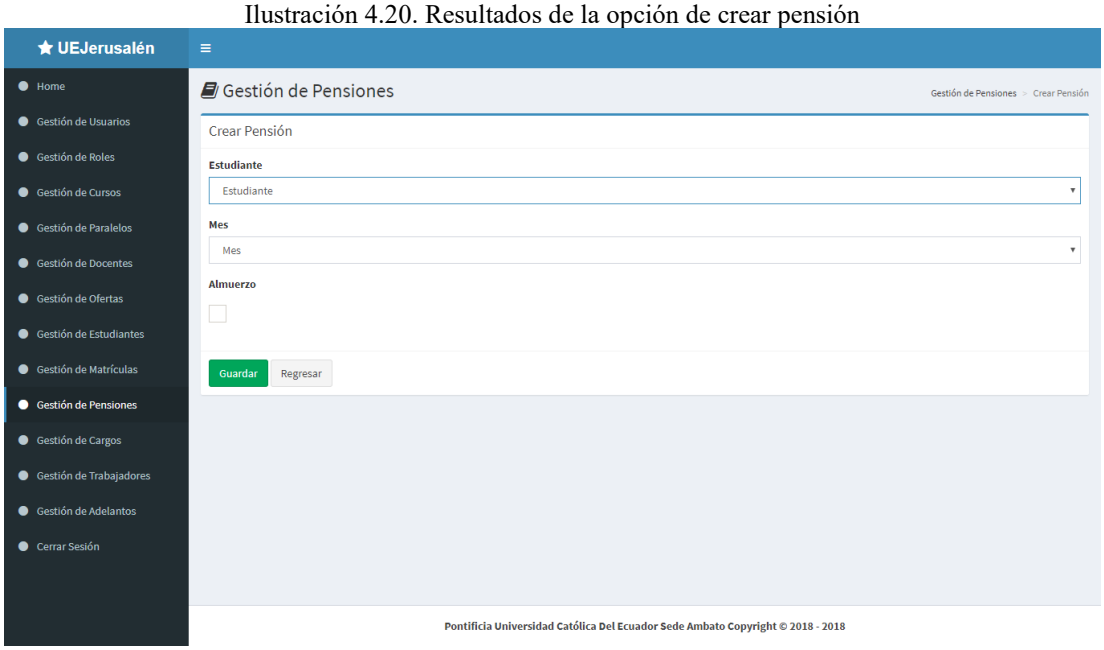
Ilustración 4.19. Resultados de la pantalla de gestión de pensiones




Fuente: Elaboración propia

La opción de crear pensión permite seleccionar un estudiante, un mes y si el estudiante dispone del servicio de almuerzo se selecciona dicha casilla y se crea la pensión.

Ilustración 4.20. Resultados de la opción de crear pensión




Fuente: Elaboración propia

La opción de reporte de pensiones se realiza por oferta, por estudiante o por mes.

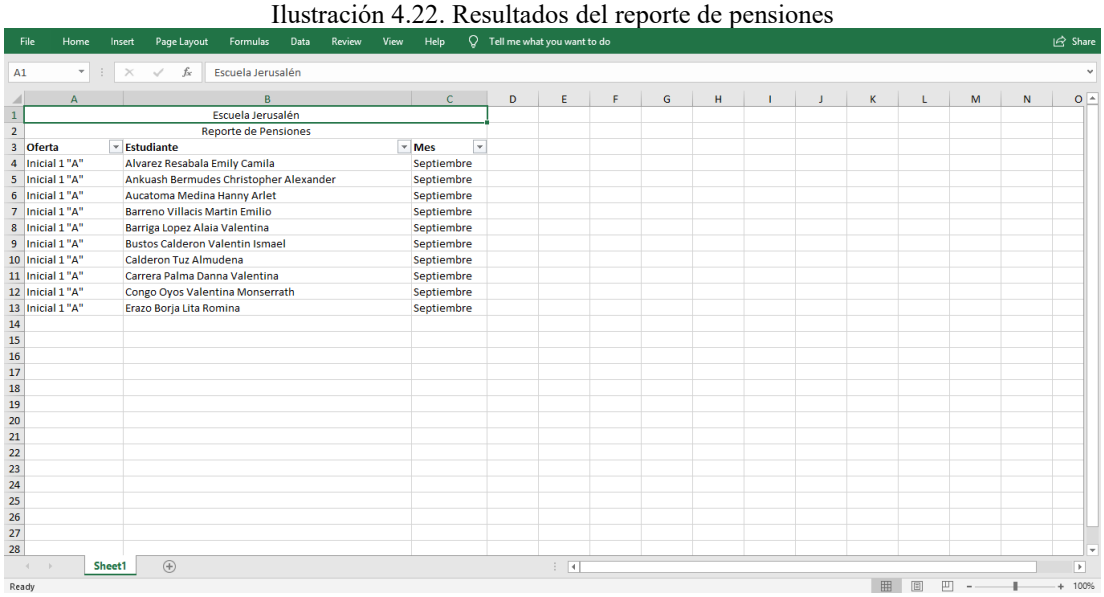
Ilustración 4.21. Resultados de la opción de consultar pensiones




Fuente: Elaboración propia

El reporte generado contiene los datos de la consulta realizada en la pantalla anterior y es exportado en una Hoja de Excel para un control de datos fácil para el usuario.

Ilustración 4.22. Resultados del reporte de pensiones

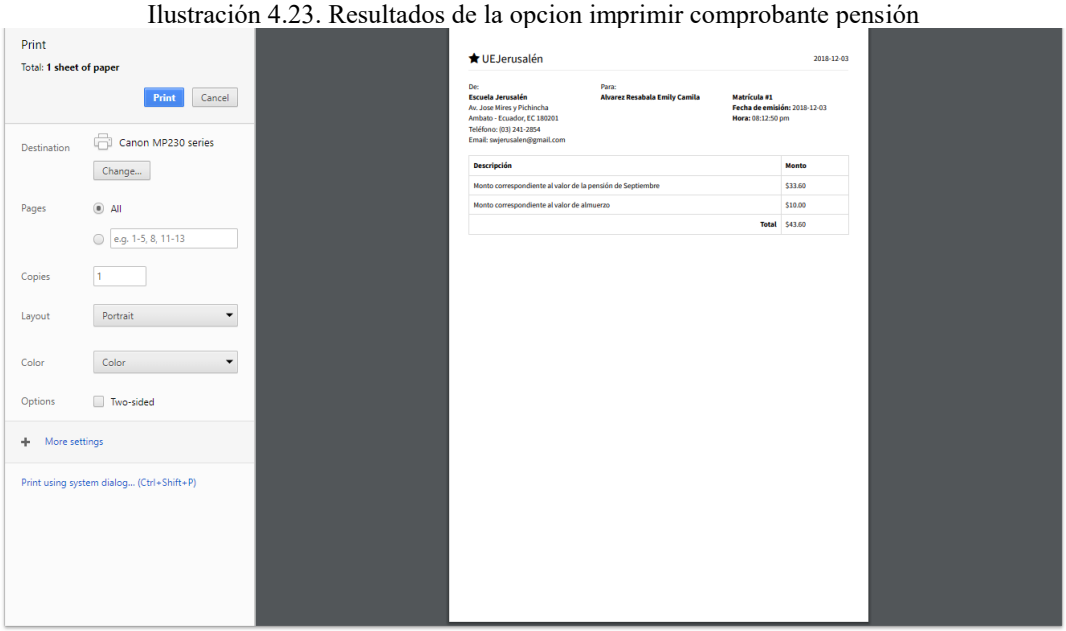


| | A | B | C |
|---|---|---|---|
| 1 | | Escuela Jerusalén | |
| 2 | | Reporte de Pensiones | |
| 3 | **Oferta** | **Estudiante** | **Mes** |
| 4 | Inicial 1 "A" | Alvarez Resabala Emily Camila | Septiembre |
| 5 | Inicial 1 "A" | Ankuash Bermudes Christopher Alexander | Septiembre |
| 6 | Inicial 1 "A" | Aucatoma Medina Hanny Arlet | Septiembre |
| 7 | Inicial 1 "A" | Barreno Villacis Martin Emilio | Septiembre |
| 8 | Inicial 1 "A" | Barriga Lopez Alaia Valentina | Septiembre |
| 9 | Inicial 1 "A" | Bustos Calderon Valentin Ismael | Septiembre |
| 10 | Inicial 1 "A" | Calderon Tuz Almudena | Septiembre |
| 11 | Inicial 1 "A" | Carrera Palma Danna Valentina | Septiembre |
| 12 | Inicial 1 "A" | Congo Oyos Valentina Monserrath | Septiembre |
| 13 | Inicial 1 "A" | Erazo Borja Lita Romina | Septiembre |

Fuente: Elaboración propia

La opción de imprimir comprobante refleja los valores cancelados de la pensión.

Ilustración 4.23. Resultados de la opcion imprimir comprobante pensión




Fuente: Elaboración propia

La opción de enviar un recordatorio del pago de la pensión mensual se lo realiza en la pantalla de gestión de estudiantes dentro del perfil de un estudiante seleccionado.

Ilustración 4.24. Resultados de la opción de ver estudiante




Fuente: Elaboración propia

La opción de enviar recordatorios permite escoger al padre o a la madre del estudiante.

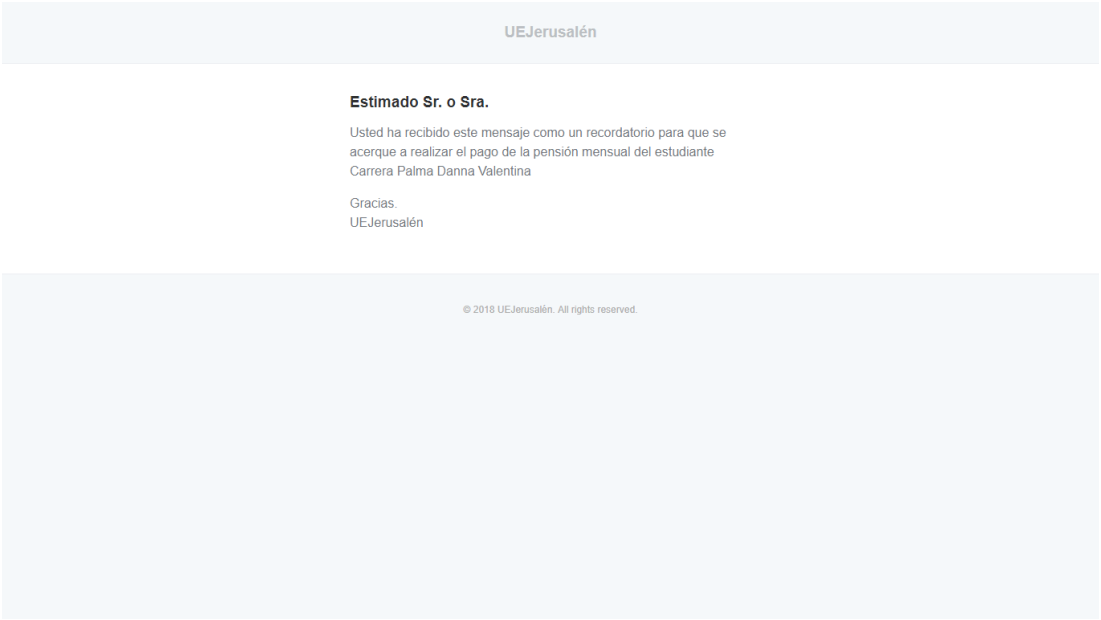
Ilustración 4.25. Resultados de la opción de enviar recordatorios




Fuente: Elaboración propia

El mensaje del correo electrónico que se envía como recordatorio es el siguiente:

Ilustración 4.26. Resultados del correo enviado como recordatorio

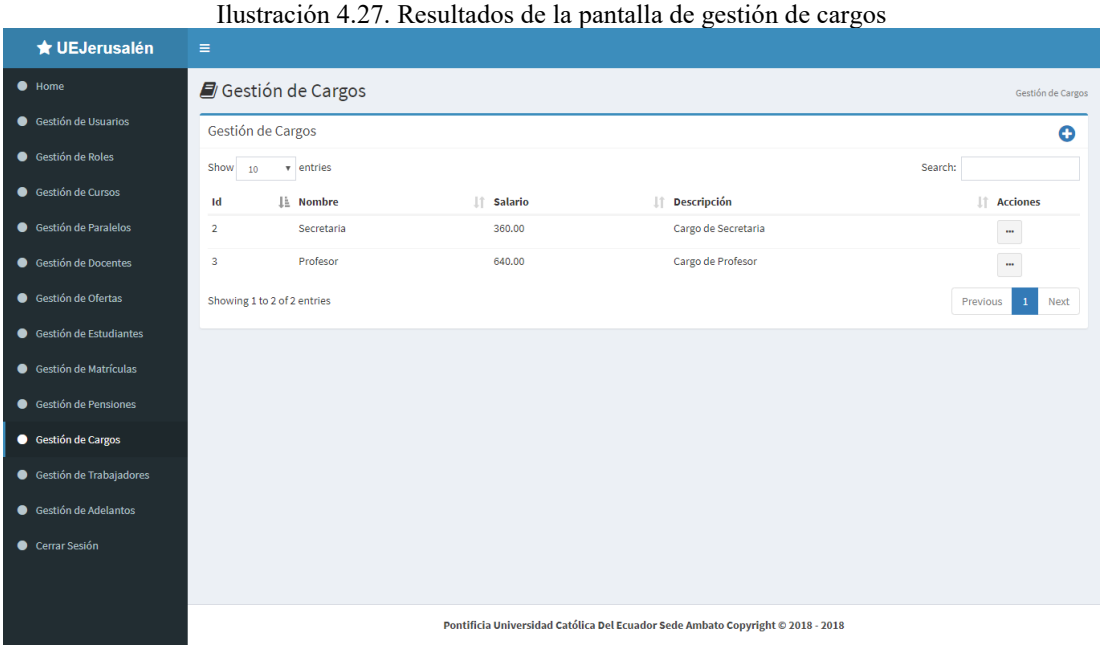



Fuente: Elaboración propia

- **Resultados del módulo de adelantos**

La pantalla de gestión de cargos permite gestionar los cargos que tiene la Institución.

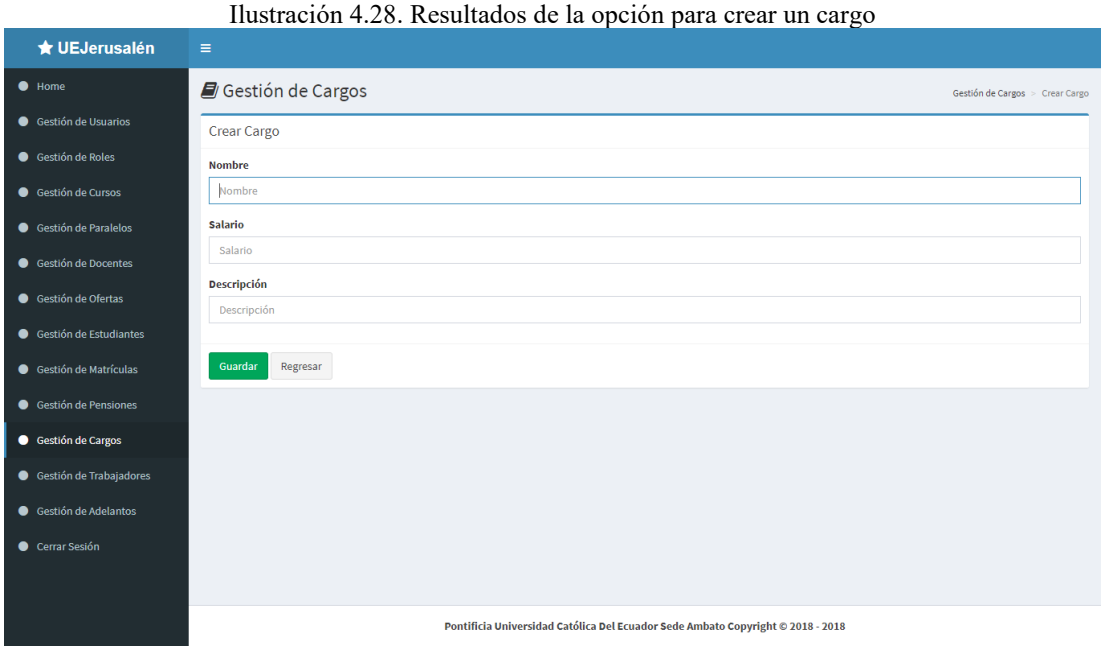
Ilustración 4.27. Resultados de la pantalla de gestión de cargos




Fuente: Elaboración propia

La opción para crear un cargo muestra los datos que se necesitan para crear un cargo.

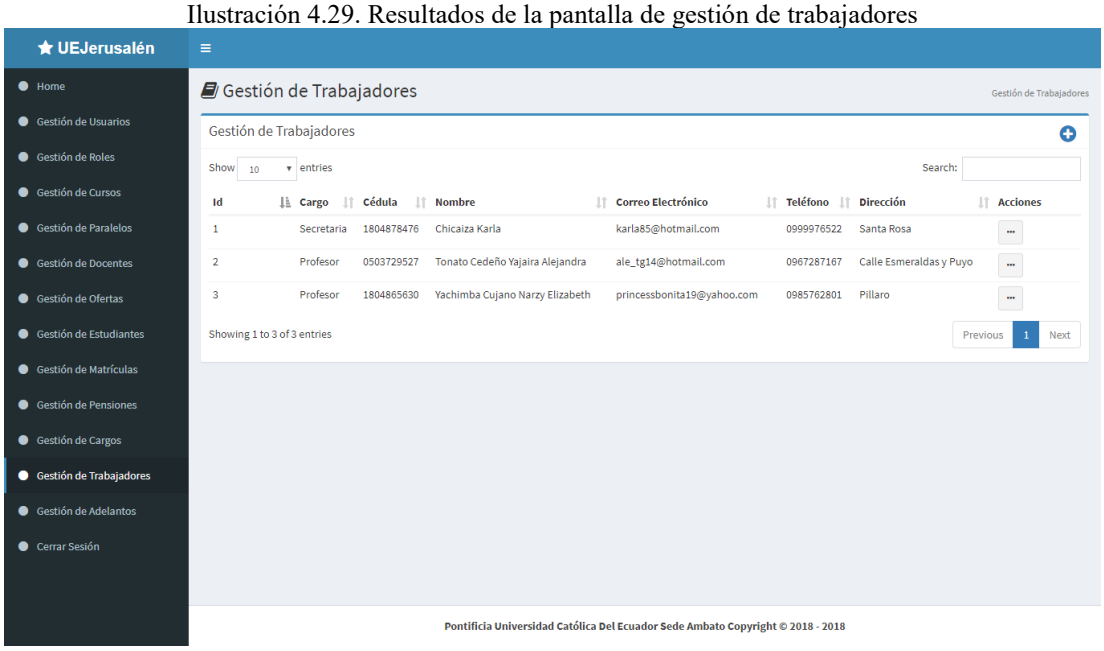
Ilustración 4.28. Resultados de la opción para crear un cargo




Fuente: Elaboración propia

La pantalla de gestión de trabajadores permite consultar a un trabajador, además de poder crear, modificar o eliminar un trabajador a este se le puede asignar un cargo.

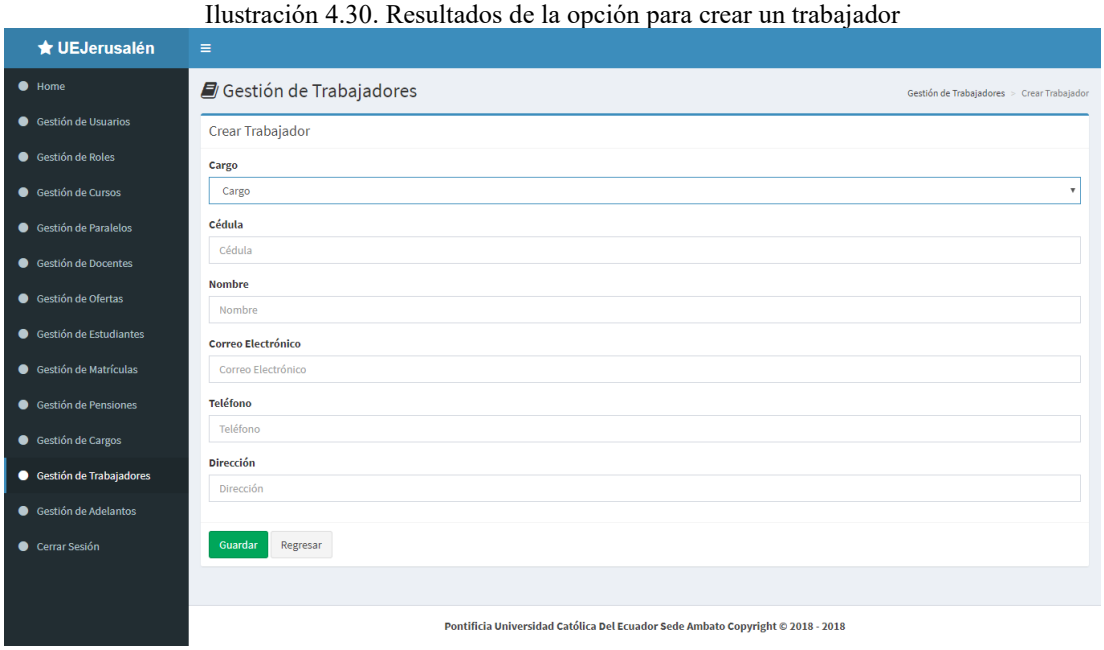
Ilustración 4.29. Resultados de la pantalla de gestión de trabajadores




Fuente: Elaboración propia

Dentro de la opción para crear un trabajador se puede registrar un nuevo trabajador.

Ilustración 4.30. Resultados de la opción para crear un trabajador




Fuente: Elaboración propia

La pantalla de gestión de adelantos contiene información sobre todos los adelantos realizados por parte de los trabajadores y permite gestionar los adelantos económicos.

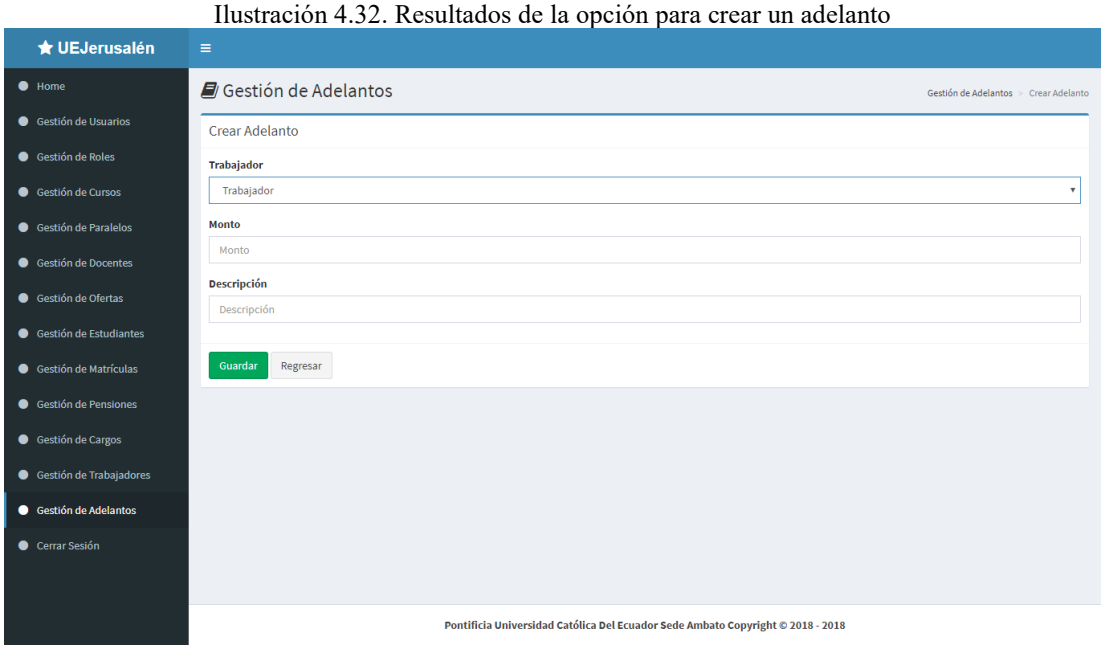
Ilustración 4.31. Resultados de la pantalla de grestión de adelantos




Fuente: Elaboración propia

La opción para crear un adelanto permite elegir a un trabajador, establecer el monto del adelanto y adjuntar una breve descripción al adelanto económico a crear.

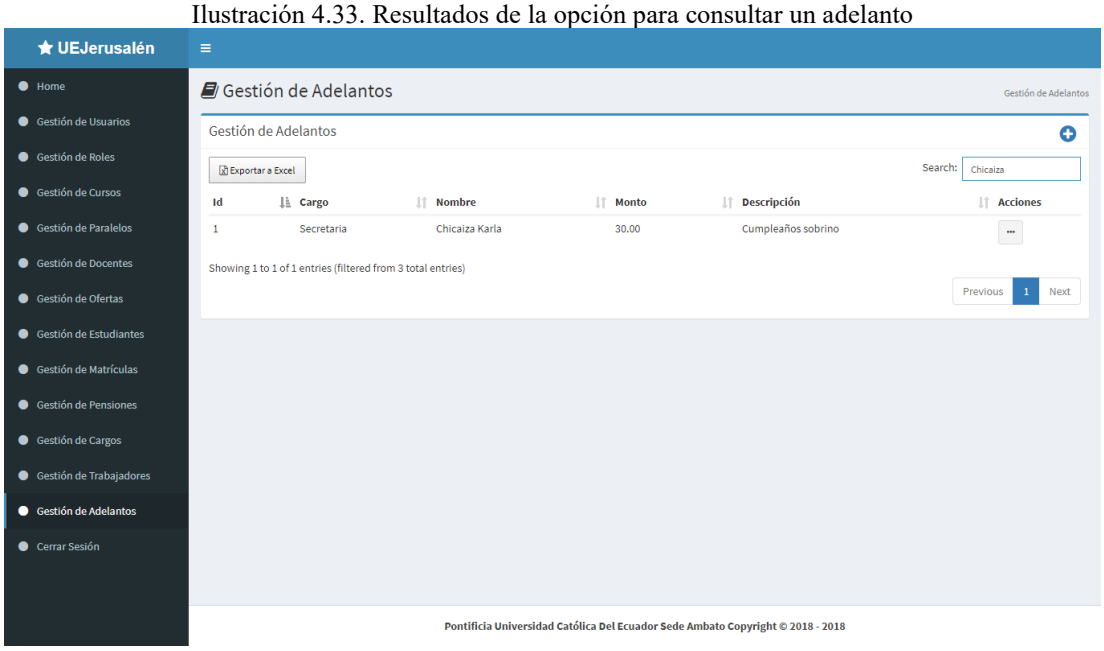
Ilustración 4.32. Resultados de la opción para crear un adelanto




Fuente: Elaboración propia

El reporte de adelantos se puede realizar por el cargo o por el nombre del trabajador.

Ilustración 4.33. Resultados de la opción para consultar un adelanto




Fuente: Elaboración propia

El resultado obtenido es un reporte en Hoja de Excel con la consulta realizada.

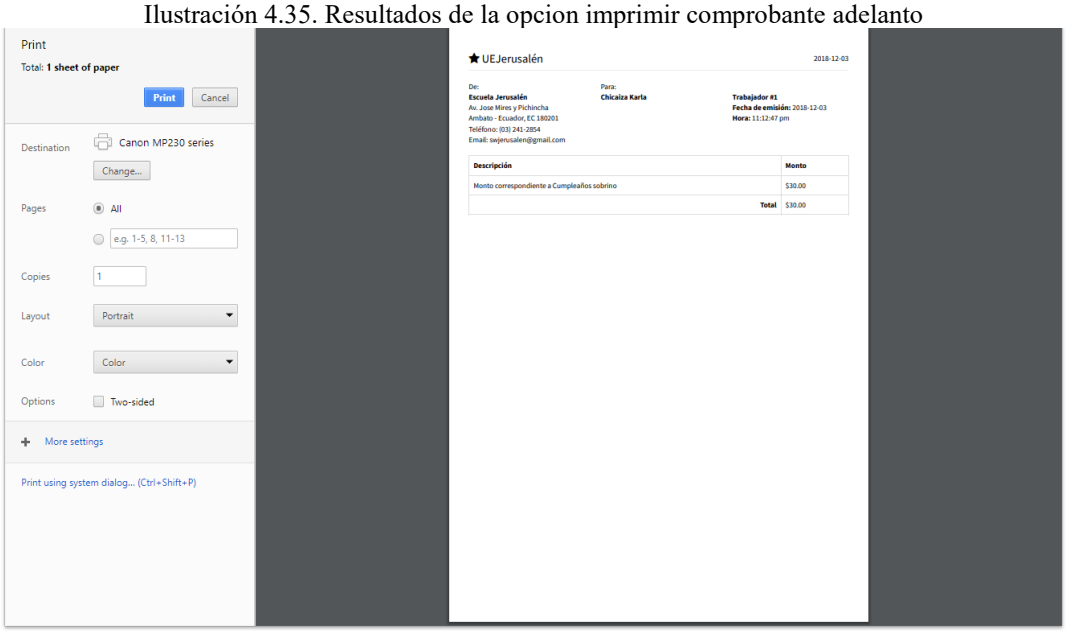
Ilustración 4.34. Resultados del reporte de adelantos




Fuente: Elaboración propia

Al registrar un adelanto se tiene la opción para imprimir un comprobante del adelanto económico realizado, para que el trabajador lo firme y quede constancia de este.

Ilustración 4.35. Resultados de la opcion imprimir comprobante adelanto




Fuente: Elaboración propia

- **Resultados del sitio web de la Unidad Educativa Jerusalén**

Cuando el visitante ingresa a uejerusalen.edu.ec puede ver la siguiente información:

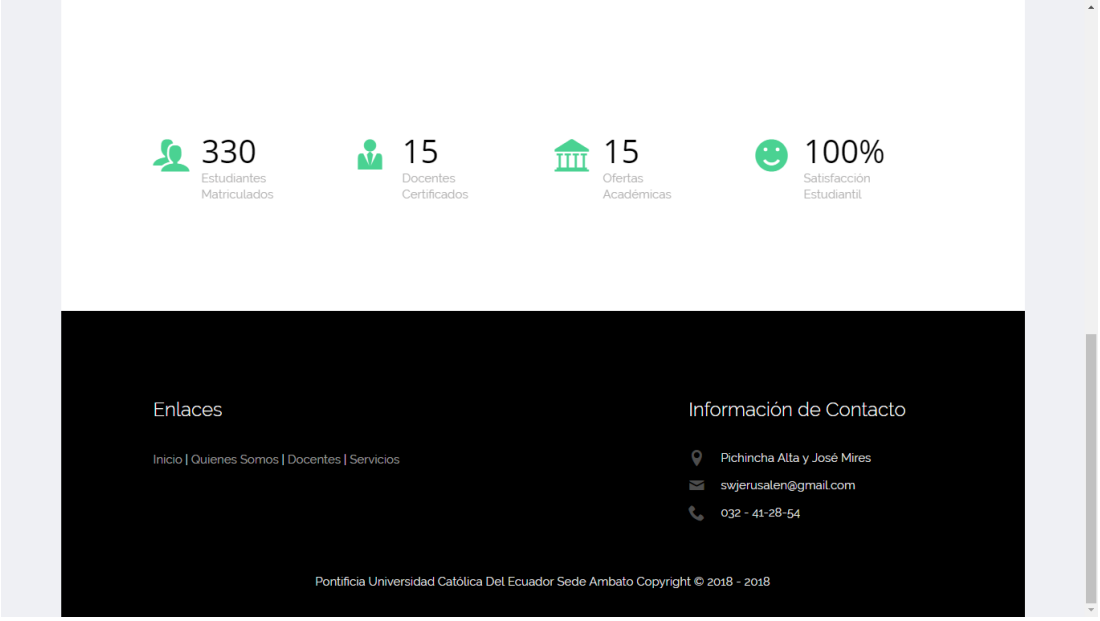
Ilustración 4.36. Resultados de la página de bienvenida del sitio web




Fuente: Elaboración propia

En la misma página el visitante puede ver un pequeño dato estadístico de la Escuela.

Ilustración 4.37. Resultados del conteo de estudiantes, docentes y ofertas




Fuente: Elaboración propia

La página “Quienes somos” alberga información sobre la historia de la Escuela.

Ilustración 4.38. Resultados de la página “Quienes Somos” del sitio web




Fuente: Elaboración propia

La página “Docentes” muestra los nombres de los Docentes y a que oferta pertenece cada uno de ellos, es decir de que oferta académica es tutor cada uno de los Docentes.

Ilustración 4.39. Resultados de la página “Docentes” del sitio web




Fuente: Elaboración propia

Finalmente, la página “Servicios” contiene información sobre los servicios que oferta la Unidad Educativa Jerusalén, con lo siguiente concluye el desarrollo del sitio web informativo y el análisis de los resultados obtenidos en este proyecto de Investigación.

Ilustración 4.40. Resultados de la página “Servicios” del sitio web




Fuente: Elaboración propia

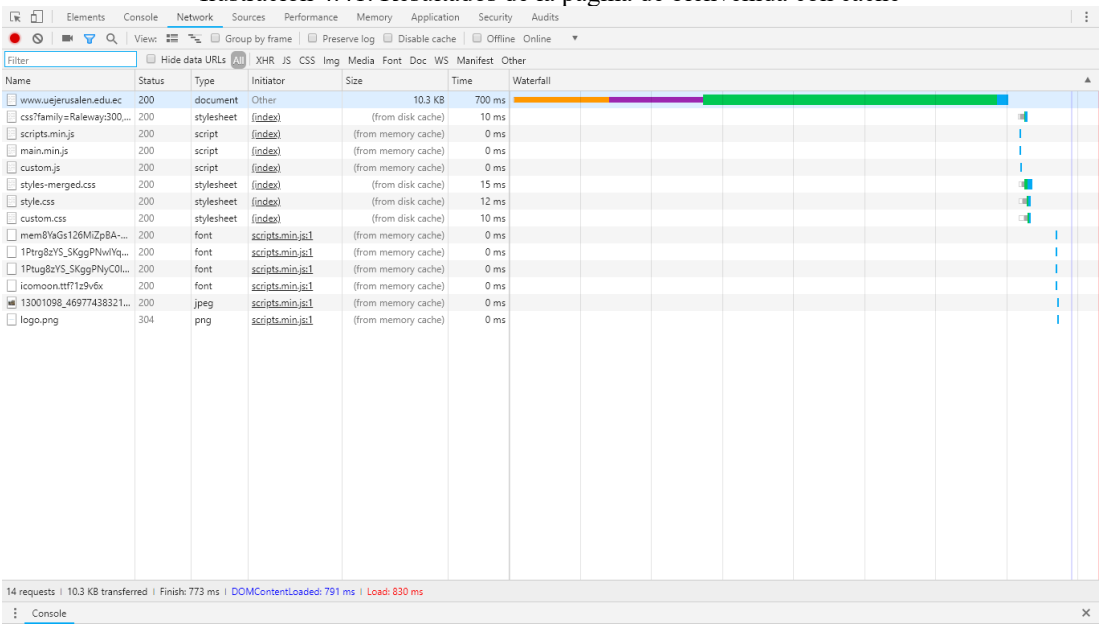
- **Resultados de las pruebas de velocidad**

Para medir la velocidad de respuesta del sitio web desarrollado y de la aplicación se utilizó una herramienta popular como lo es la consola de herramientas de desarrollador de Google Chrome, dicha consola contiene herramientas para ver elementos de la página web, la velocidad de carga, el rendimiento, datos de seguridad, etc.

Para empezar, se midió el tiempo de carga de la página de bienvenida del sitio web y de la página principal de la aplicación, posteriormente se realizó la misma actividad, pero con el *cache* deshabilitado lo que significa que ambas páginas tendrían que cargar de nuevo todos sus archivos (hojas de estilo, scripts, imágenes).

Primero los resultados de tiempo de carga de la página de bienvenida del sitio web.

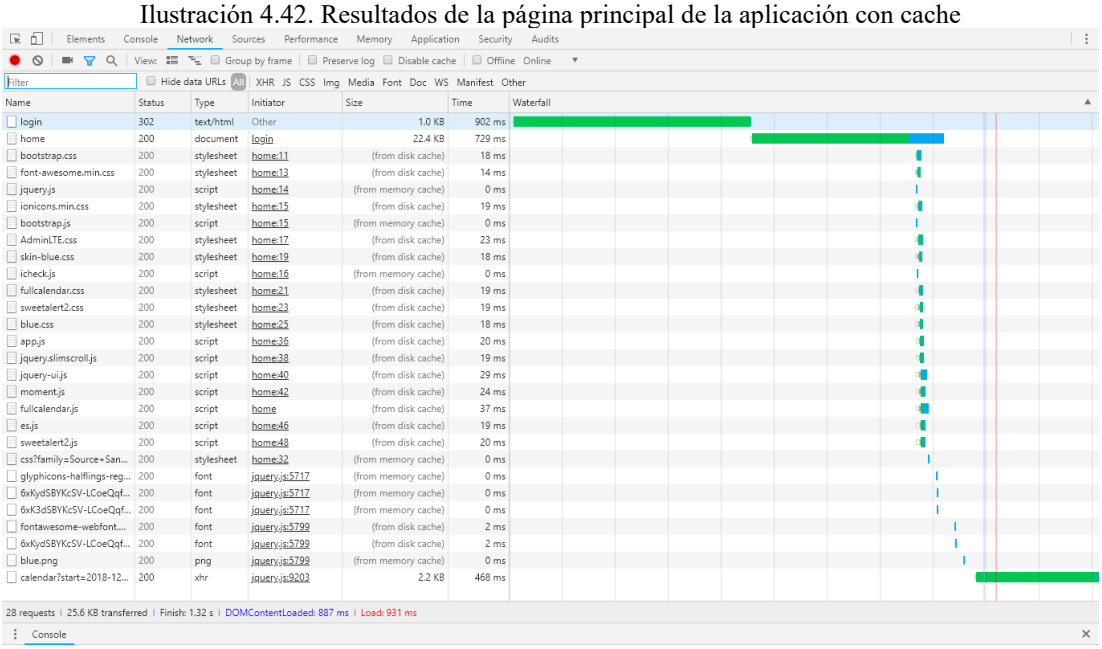
Ilustración 4.41. Resultados de la página de bienvenida con cache




Fuente: Elaboración propia

Ahora, los resultados de tiempo de carga de la página principal de la aplicación.

Ilustración 4.42. Resultados de la página principal de la aplicación con cache




Fuente: Elaboración propia

Como se puede observar en la siguiente tabla el tiempo de carga es muy bueno.

Tabla 4.1. Resultados del tiempo de carga con cache

| Tiempo de carga con cache | |
|---|---|
| Sitio Web | Finish: 773 ms \| DOMContentLoaded: 791 ms \| Load: 830 ms |
| Aplicación Web | Finish: 1.32 s \| DOMContentLoaded: 887 ms \| Load: 931 ms |

Fuente: Elaboración propia

A continuación, se documenta la misma actividad, pero en esta ocasión con el cache del navegador deshabilitado, esto con el propósito de comparar el tiempo de carga.

Tabla 4.2. Resultados del tiempo de carga sin cache

| Tiempo de carga con el cache deshabilitado | |
|---|---|
| Sitio Web | Finish: 4.52 s \| DOMContentLoaded: 3.01 s \| Load: 4.56 s |
| Aplicación Web | Finish: 2.35 s \| DOMContentLoaded: 1.90 s \| Load: 2.12 s |

Fuente: Elaboración propia

Como se puede ver el tiempo de carga es muy bueno cuando se tiene en cuenta el hecho de que la memoria cache en *Laravel* es permanente, es decir la misma se almacena dentro de una carpeta en la estructura del *Framework,* en el peor de los casos la memoria *cache* debe ser eliminada de manera forzosa del navegador web por parte del usuario lo cual no es una práctica realmente cotidiana, aún sin la memoria *cache* la velocidad de carga es muy buena para una aplicación pesada y con cientos de registros en especial en la gestión de matrículas y pensiones en donde se pueden manejar varios registros por alumno, en síntesis la velocidad de carga de la aplicación web permite que el usuario optimice su tiempo y por ende se ahorren recursos,

## 4.2.Validación de resultados

Ilustración 4.43. Certificado de validación de resultados



ESCUELA DE EDUCACIÓN BÁSICA
"JERUSALÉN"
Pichincha Alta y José Mires
18h00018@gmail.com
Telf.032412-854
Ambato-Ecuador

Ambato, 03 de Diciembre del 2018.

**CERTIFICADO**

En mi calidad de Directora de la Escuela de Educación Básica Jerusalén, por medio del presente tengo a bien certificar que el Señor: **Juan Marcelo Villagrán Fiallos**, portador de la cedula de ciudadanía N.º 1803286119, ha concluido satisfactoriamente con la implementación de la Aplicación web para la Gestión de Matrículas y Pensiones de los estudiantes, cumpliendo las expectativas de la Institución, sistema que va a ser utilizado de inmediato en el presente año lectivo.

El interesado puede hacer uso del presente documento en la manera que estime conveniente.

Atentamente,

CENTRO EDUCATIVO BÁSICO "JERUSALÉN" DIRECCIÓN AMBATO

Lcda. Fanny Hernández
**DIRECTORA**

Fuente: Elaboración propia

## 4.3.Capacitación

Resultados de la capacitación realizada a la Secretaria de la Institución Educativa.

Ilustración 4.44. Capacitación a la Secretaria.



Fuente: Elaboración propia

# CAPÍTULO V

# CONCLUSIONES Y RECOMENDACIONES

## 5.1.Conclusiones

- Por medio de la investigación bibliográfica y de la elaboración de un marco conceptual sobre aplicaciones web, se pudo adquirir una cantidad significativa de conocimiento que sirvió para determinar que un *Framework* PHP es ideal para emprender un proyecto de desarrollo sin importar el tamaño de este, se concluye además qué los beneficios obtenidos al utilizar un *Framework* PHP para la consecución del proyecto fueron de gran importancia tanto para la Unidad Educativa Jerusalén como para el desarrollador de la aplicación.

- Mediante el diagnóstico de la situación inicial a la cual estaba sujeta la Unidad Educativa Jerusalén se pudo conocer que tanto la Directora como la Secretaria llevaban un registro de las pensiones de forma individual, lo que significaba que los registros que tenía la Secretaria no eran los mismos registros que los registros que tenía la Directora, más aun cuando terceras personas receptaban el pago y no existía constancia de aquello, es por esta razón por la cual la implementación de la aplicación web en la Institución resolvió dicho problema al generar una base de datos unificada en donde se almacenan todos los registros de las transacciones realizadas.

- El uso de un servicio de repositorios es una ventaja para el desarrollo de aplicaciones ya que permite al desarrollador experimentar nuevas características y funciones, en el proyecto realizado cada módulo fue desarrollado y combinado de tal manera que los cambios se unieron a la derivación que contiene la aplicación final sin ningún problema.

- La capacitación de los usuarios finales se la realizó en dos partes, la primera consistió en una breve introducción a las funciones del sistema, mientras que la segunda se la ejecutó de manera interactiva en donde los usuarios participaron con ejemplos reales complementando así la primera parte, de acuerdo con las pruebas de aceptación realizadas al final de cada iteración en la fase de aprendizaje respectivamente, se pudo conocer que los usuarios entendían el concepto de cómo funciona la aplicación lo que indica de esta forma que la aplicación es fácil de manejar lo cual beneficia al usuario final.

## 5.2.Recomendaciones

- Se recomienda utilizar el navegador Google Chrome por defecto, en el mismo se probó la aplicación durante todo el ciclo de desarrollo.

- Mantener la información de los estudiantes, contactos y docentes en constante actualización.

- Dada la capacitación a los usuarios finales se sugiere establecer como norma de la Institución el uso obligatorio de la aplicación para registrar o realizar el cobro de aranceles.

- Para agregar nuevas funcionalidades o características presentar una solicitud de extracción con el detalle de lo que se desea agregar o implementar.

# BIBLIOGRAFÍA


[1] Ministerio del Trabajo, «Acuerdo Ministerial No. MDT-2015,» 2015. [En línea]. Available: http://www.trabajo.gob.ec/wp-content/uploads/2015/12/Acuerdo-Minsterial-262-Regulacion-de-las-Relaciones-de-Trabajo-Especiales-del-Sector-de-Transporte-Terrestre-de-Pasajeros-y-de-Carga.pdf. [Último acceso: 26 Septiembre 2018].

[2] Definicion.DE, «definicion.de/aplicacion,» [En línea]. Available: https://definicion.de/aplicacion/. [Último acceso: 27 Septiembre 2018].

[3] L. Rodriguez Araron, «Universidad Rey Juan Carlos,» [En línea]. Available: https://previa.uclm.es/profesorado/licesio/Docencia/IB/IBTema3b.pdf. [Último acceso: 27 Septiembre 2018].

[4] G. Delgado Ugarte, «Instituto de Educación Superior Tecnológico Privado del Sur,» [En línea]. Available: https://www.isur.edu.pe/es/articulo/escuela-de-tecnologias-de-la-informacion/apps-vs-software-de-escritorio. [Último acceso: 27 Septiembre 2018].

[5] C. Gartenberg, «The Verge,» [En línea]. Available: https://www.theverge.com/2018/7/10/17550430/apple-iphone-ios-app-store-10-years-look-back. [Último acceso: 27 Septiembre 2018].

[6] IBM Corporation, «Native, web or hybrid mobile-app development,» IBM Software, New York, 2012.

[7] R. Pressman, Ingeniería de Software, Un Enfoque Práctico, Madrid: McGraw-Hill, 2010.

[8] C. Mateu, Introduction to web application development, Barcelona: Universitat Oberta de Catalunya, 2010.

[9] Open Source Initiative, «Open Source Definition,» [En línea]. Available: https://opensource.org/osd.html. [Último acceso: 03 07 2017].

[10] Mozilla Developer Network, «Que es HTML?,» [En línea]. Available: https://developer.mozilla.org/es/docs/Web/HTML. [Último acceso: 02 07 2017].

[11] World Wide Web Foundation, «History of the Web,» [En línea]. Available: https://webfoundation.org/about/vision/history-of-the-web/. [Último acceso: 1 Octubre 2018].

[12] J. Eguíluz Pérez, Introducción a CSS, librosweb.es, 2008.

[13] CDNJS, «About,» [En línea]. Available: https://cdnjs.com/about. [Último acceso: 04 07 2017].

[14] J. Pavón Mestras, «Aplicaciones Web/Sistemas Web,» [En línea]. Available: https://www.fdi.ucm.es/profesor/jpavon/web/24-JavaScript.pdf. [Último acceso: 04 07 2017].

[15] H. Shakirat Oluwatosin, «Client-Server Model,» *IOSR Journal of Computer Engineering,* February 2014.

[16] T. Lampoltshammer y B. Zechmeister, «Software Release Management,» University of Salzburg.

[17] ATLASSIAN Bitbucket, «What is version control,» [En línea]. Available: https://www.atlassian.com/git/tutorials/what-is-version-control. [Último acceso:

1 Octubre 2018].

[18] ATLASSIAN Bitbucket, «Git cheat sheet,» [En línea]. Available: https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet. [Último acceso: 1 Octubre 2018].

[19] J. Pastor Perez, «Estudio y clasificación de tipos de aplicaciones web y determinacion de atributos de usabilidad mas relevantes,» UNIVERSITAT POLITÉCNICA DE VALÉNCIA, Valencia, 2013.

[20] S. Luján Mora, Programación de aplicaciones web: Historia, Principios Básicos y Clientes Web., Alicante: Imprenta Gamma, 2002.

[21] L. Chung, «Computer Science Program, The University of Texas, Dallas,» [En línea]. Available: https://www.utdallas.edu/~chung/SA/2client.pdf. [Último acceso: 20 07 2017].

[22] G. Coulouris, J. Dollimore y T. Kindberg, Sistemas Distribuidos Conceptos y Diseño, Madrid: Addison Wesley, 2001.

[23] S. Bellovin, «Columbia University,» [En línea]. Available: https://www.cs.columbia.edu/~smb/classes/f06/l09.pdf. [Último acceso: 14 08 2017].

[24] Definicion.DE, «definicion.de/administracion,» [En línea]. Available: https://definicion.de/administracion/. [Último acceso: 2 Octubre 2018].

[25] Z. Torres Hernández, Teoría General de la Administración, Grupo Editorial Patria, 2014.

[26] I. Chiavenato, Administración Teoria, Proceso y Práctica, McGraw-Hill, 2001.

[27] Definicion.DE, «definicion.de/proceso,» [En línea]. Available:

https://definicion.de/proceso/. [Último acceso: 2 Octubre 2018].

[28] J. L. Vinueza Martinez y M. A. Correa Peralta, «La calidad en los procesos informaticos de las Universidades Ecuatorianas,» *Calidad de los Sistemas Educativos,* 2014.

[29] J. Gutiérrez, «¿Qué es un framework web?,» Departamento de Lenguajes y Sistemas Informáticos Universidad de Sevilla.

[30] Q. Nguyen, «Building a web application with laravel 5,» Oulu University of Applied Sciences, 2015.

[31] E. Bahit, POO y MVC en PHP, 2011.

[32] Atom, «Atom a hackable text editor,» Github, [En línea]. Available: https://atom.io/. [Último acceso: 12 05 2018].

[33] HeidiSQL, «HeidiSQL - Home,» [En línea]. Available: https://www.heidisql.com. [Último acceso: 18 05 2018].

[34] Laragon, «Laragon - Documentation,» [En línea]. Available: https://laragon.org/docs/. [Último acceso: 22 05 2018].

[35] Composer, «Composer - Introduction,» [En línea]. Available: https://getcomposer.org/doc/00-intro.md. [Último acceso: 22 05 2018].

[36] Mozilla, «Introduction - JavaScript,» [En línea]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction#What_is_JavaScript. [Último acceso: 22 05 2018].

[37] R. Hernandez, C. Fernandez y M. Baptista, Metodología de la investigación, Mexico: McGraw-Hill, 2003.

[38] P. Bourque y R. Fairley, Guide to the Software Engineering Body of Knowledge, IEEE Computer Society, 2014.

[39] K. Orr, «Chapter 23. Adaptive Software Development,» [En línea]. Available: http://www.exa.unicen.edu.ar/catedras/agilem/cap23asd.pdf. [Último acceso: 4 Octubre 2018].

[40] J. Highsmith, Adaptive Software Development, Dorset House, 2000.

# ANEXOS

## Anexo 1 - Entrevista a la Directora de la Institución.

1. ¿Describa el proceso actual de las actividades realizadas al matricular a un estudiante?

2. ¿Actualmente qué tipo de software o sistema utiliza para administrar la información de los estudiantes?

3. ¿Actualmente cómo manejan la información de los estudiantes?

4. ¿Actualmente qué tipo de documentos o información maneja el Departamento de Secretaría?

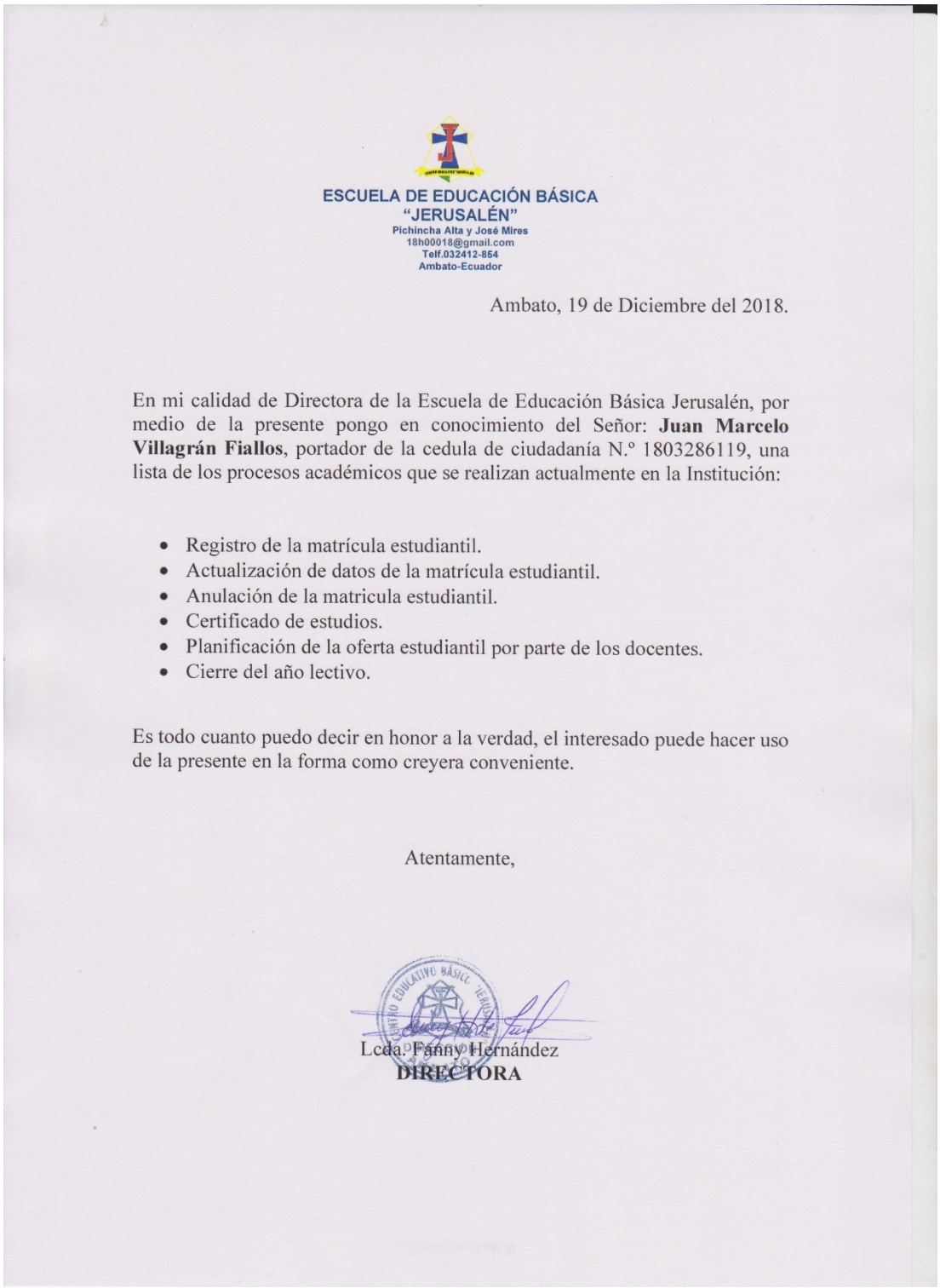
5. ¿Estaría de acuerdo en utilizar un sistema para administrar la información?

6. ¿Cuántas personas utilizarían la aplicación?

7. ¿Qué tipo de reportes le gustaría poder tener en la aplicación?

8. ¿Qué características o funciones le gustaría que la aplicación tenga?

# Anexo 2 - Lista de procesos académicos.



ESCUELA DE EDUCACIÓN BÁSICA
"JERUSALÉN"
Pichincha Alta y José Mires
18h00018@gmail.com
Telf.032412-854
Ambato-Ecuador

Ambato, 19 de Diciembre del 2018.

En mi calidad de Directora de la Escuela de Educación Básica Jerusalén, por medio de la presente pongo en conocimiento del Señor: **Juan Marcelo Villagrán Fiallos**, portador de la cedula de ciudadanía N.º 1803286119, una lista de los procesos académicos que se realizan actualmente en la Institución:

- Registro de la matrícula estudiantil.
- Actualización de datos de la matrícula estudiantil.
- Anulación de la matricula estudiantil.
- Certificado de estudios.
- Planificación de la oferta estudiantil por parte de los docentes.
- Cierre del año lectivo.

Es todo cuanto puedo decir en honor a la verdad, el interesado puede hacer uso de la presente en la forma como creyera conveniente.

Atentamente,

Lcda. Fanny Hernández
**DIRECTORA**

Ilustración 1. Lista de procesos académicos