PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

Aplicación de la metodología RUP y el patrón de diseño MVC en la construcción de un sistema de gestión académica para la Unidad Educativa Ángel De La Guarda

Wendy Jaramillo W.

Quito, 2016

Para Dios, mis padres, mis hermanos, Mami Vicky y José Eduardo León, y todo lo que representan para mí…por acompañarme y guiarme en esta etapa y experiencia de mi vida con todo su amor, apoyo y comprensión.

Para Ana María y Sebastián por ser amigos incondicionales que supieron brindarme su ayuda en este y muchos proyectos a lo largo de nuestra carrera.

Un especial agradecimiento a los Ingenieros: Fabián de La Cruz, Beatriz Campos y Eddy Sánchez; profesionales, profesores y aliados de las metas de muchos estudiantes de Ingeniería en Sistemas; quienes, junto con otros maestros, han sabido ser catedráticos de la ciencia y de la vida.

RESUMEN

Este proyecto tiene como objetivo principal brindar a la Unidad Educativa "Ángel de la Guarda" la oportunidad de optimización y crecimiento de una de sus áreas principales, el área académica; mediante el uso de herramientas tecnológicas que la conviertan en una institución más competitiva y capaz dentro del mundo moderno en el que la sociedad actual se desarrolla.

El sistema de este trabajo de disertación se enfocó mediante el uso de una de las metodologías más fiables y amigables para el cliente en todo momento, la metodología RUP (Rational Unified Process); con ella se siguió paso a paso el avance del proyecto y se identificó en cada fase los caminos correctos e incorrectos en los cuales se dirigía éste a medida que progresaba. Dividiéndose en cuatro fases principales: Inicio (visión general del proyecto), Elaboración (requerimientos detallados del cliente y forma o arquitectura en la que se construiría el sistema), Construcción (versión de prueba estable del sistema) y Transición (versión final probada y entregada del sistema), cada una está ligada a su documentación correspondiente, pudiéndose incluir: riesgos, problemas, cambios, planificaciones, diagramas, entre otros.

El sistema fue construido con un diseño de base de datos en el motor MySQL y desarrollado en su mayoría con el lenguaje de programación PHP. La arquitectura que le da forma y funcionalidad a toda esta plataforma es el patrón MVC (Modelo-Vista-Controlador) el cual se encarga de recibir lo que el usuario desea en el Controlador, enviarlo hacia el Modelo que se encargará de hacer los procesos necesarios en la base de datos para devolver una respuesta al Controlador otra vez, para finalmente mostrar al usuario una pantalla con el resultado de lo que quería. El software se encuentra funcionando en un servidor local configurado en Linux dentro de la institución.

El Sistema de Gestión Académica fue entregado a la Unidad Educativa "Ángel de la Guarda" (tanto software como hardware) sin costo alguno y las autoridades y personal de la institución lo recibieron con total entusiasmo y agradecimiento, dispuestos a usarlo y aprovecharlo al máximo, así como lo demostraron en las capacitaciones.

ABSTRACT

This project's primary objective is to give "Ángel de la Guarda" School the opportunity to optimize and enhance its academic area through the employment of technological tools that will give the institution a more competitive and skilled profile in the modern society.

The system developed in this dissertation thesis was based on one of the most user-friendly and trustworthy methodologies, the RUP (Rational Unified Process) Methodology, with it the project's progression was followed step by step, identifying the right and wrong paths in each phase as it was forging ahead. Divided into four main phases: Inception (general overview of the project), Elaboration (client's specific requirements and system architecture), Construction (system's beta version), and Transition (final version, tested and delivered); each division has its supporting documentation attached, such as: risks, issues and problems, changes and corrections, schedules, diagrams, among others.

This software was built using a database model under MySQL Object-Oriented Database Management System and mostly written on PHP Programming Language. The architecture responsible for giving functionality to the whole platform is MVC (Model-View-Controller) design pattern, that works as follows: The user sends a request to the page, this request is received by the Controller and then processed into the Model, which in turn performs the corresponding operations into the database and sends a response back to the Controller, to finally display the results for the user. The system is currently working over a local network, with Linux configuration.

This Academic Management System was given without any cost to the "Ángel de la Guarda" School (both software as hardware) and was gratefully and enthusiastically received by its directives and personnel who showed their willingness to use and take full advantage of it, as well as they did during the training period.

CONTENIDO

Capítulo 1. INTRODUCCIÓN

La Unidad Educativa Ángel de la Guarda sin duda es una de las instituciones educativas más antiguas de Quito, al ser una organización sin fines de lucro carece de aspectos que en el pasado pudieron haber sido catalogados como superfluos, pero que el día de hoy son la base sostenible del desarrollo social, económico y humano: la herramienta tecnológica. Este capítulo trata sobre los fundamentos que hacen que esta institución educativa sea importante para la sociedad y su necesidad de surgir en un mundo globalizado.

1.1 DATOS DE LA ORGANIZACIÓN

Nombre: Área académica de la Unidad Educativa Ángel De La Guarda

Actividad: Administración de los registros académicos de los estudiantes de la Unidad Educativa Ángel De La Guarda.

Ubicación: Unidad Educativa Ángel de la Guarda, La exposición E2-52 y Juan Benigno Vela, Sector La Recoleta, Quito, Ecuador.

Características:

Fundada el 19 de marzo de 1885, la Unidad Educativa Ángel de la Guarda es una institución en la Ciudad de Quito que forma de manera humanista e integral a más de 500 niñas, por sus aulas han pasado más de 110 generaciones y alberga una rica historia colonial.

Aunque ha experimento evidentes cambios en lo educativo a la par con la evolución del tiempo, teniendo laboratorios de ciencias naturales, computación, sala de audiovisuales y clases de idiomas impartidas por extranjeros voluntarios, el plantel aún no cuenta con un sistema que permita al cuerpo docente y administrativo agilizar el manejo del área académica.

Cada docente tiene con o más materias asignadas y dicta clases a uno o más cursos, de manera que lleva un registro académico de muchos estudiantes, incluyendo: calificaciones, asistencia, conducta, entre otros. A su vez, cada alumno y alumna participa en varias materias. Todo registro es llevado de distinta manera por cada profesor, ya sea manual o mediante el uso de hojas de cálculo.

1.2 JUSTIFICACIÓN

Dado que este centro de educación primaria adolece de procesos automatizados y frente a los avances científicos y tecnológicos de su entorno, es su obligación buscar una herramienta que le facilite a los miembros de la institución: autoridades, personal docente, administrativo y de servicio, liberar la sobrecarga de responsabilidades y atender sus funciones principales.

De esta forma se lograría un desarrollo transparente y seguro para la evaluación de rendimiento académico, informes de faltas, casos disciplinarios y procesos formativos institucionales de la unidad académica mencionada, mejorando la calidad de sus prestaciones y optimizando el tiempo de sus docentes.

1.3 ANTECEDENTES

En la actualidad, muchos centros educativos buscan mejorar su calidad y eficiencia mediante el uso de herramientas tecnológicas. La gestión manual y/o no estandarizada de evaluación estudiantil puede correr riesgos de errores de calificaciones, obstaculizar el tiempo eficiente de los docentes para continuar con sus tareas respectivas y a largo plazo puede haber pérdida de información vital para auditorías y procesos legales de la institución.

1.4 OBJETIVOS

1.4.1 Objetivo General

Implantar un sistema automatizado que administre y controle el área académica de la Unidad Educativa Ángel de la Guarda.

1.4.2 Objetivos Específicos

- Usar metodología RUP para el desarrollo del proyecto
- Documentar el proyecto.
- Realizar pruebas del producto final.
- Mantener involucrado al cliente en el desarrollo del producto.
- Entregar el proyecto en un período máximo de 6 meses.

Capítulo 2. FUNDAMENTOS TEÓRICOS

Cuando un libro está lleno de información sobre "algo" que se hizo, es un libro vacío. Un libro lleno es aquel capaz de explicar a sus lectores aquellos simples o complejos detalles que podrían escaparse y así confundirnos. Los fundamentos teóricos son el núcleo por dónde se empieza un proyecto, son los conceptos y definiciones de lo que tratará el libro hasta el final de sus páginas.

2.1 METODOLOGÍAS DE INVESTIGACIÓN

La metodología de investigación de este proyecto informático es tanto cualitativa como cuantitativa y se divide en cuatro partes principales. La primera fase incluye la hipótesis sobre la accesibilidad, relevancia y usabilidad de automatizar los procesos académicos de la institución. A continuación, en la segunda fase de la investigación se incluye toda la información necesaria para establecer definiciones detalladas sobre el tema investigado, marco teórico y variables que pueden afectar el análisis y la interpretación de la investigación del problema. La tercera fase se encarga de generar y colectar datos, usando métodos tanto cualitativos como cuantitativos, aplicando metodologías de desarrollo (RUP en nuestro caso), métodos de problema-solución (Patrones de diseño en nuestro caso). Finalmente, la cuarta fase determinará si la hipótesis operacional, derivada de la recolección de datos, puede ser respaldada o refutada.

De manera que se puede decir que este proyecto busca denotar una manera de elaborar una investigación mediante exploración e integración de tecnologías disponibles para entregar un sistema que se enfoque en "probar la teoría" más que en construirla, dando paso a un progreso fluido desde el desarrollo a la evaluación.

El desarrollo de este sistema se compromete con tres pasos principales: desarrollo de conceptos, construcción del sistema y evaluación del sistema.

La metodología cualitativa se aplica mediante constantes reuniones con el usuario final (Personal administrativo de la Unidad Educativa Ángel de la Guarda), para ver el progreso cualitativo (calidad, confianza, satisfacción) del sistema. Mientras que la metodología cuantitativa se verá plasmada en los indicadores e informes de la documentación generada a partir del RUP.

2.2 SELECCIÓN DE HERRAMIENTAS DE DESARROLLO

2.2.1 Metodología RUP

2.2.1.1 ¿Qué es?

El Proceso Unificado Racional (Rational Unified Process – **RUP**) es un Proceso de **Ingeniería de Software**[1] que provee un acercamiento disciplinado para asignar tareas y responsabilidades en una organización

[1] Ingeniería de Software: Disciplina de ingeniería que comprende todos los aspectos de la producción de software. (Ian Sommerville, 2005)

desarrollada. Su principal objetivo es asegurar la producción de software de alta calidad, el cual llega a las necesidades del cliente final, por medio de un horario y presupuesto predecibles.

El Proceso Unificado Racional es un proceso de producto, desarrollado y financiado por **Rational Software**, grupo que se encarga de trabajar de cerca con clientes y socios en busca del aseguramiento de que todo proceso sea actualizado y mejorado constantemente para evolucionar y probar mejores prácticas.

RUP mejora la productividad grupal mediante la asignación de fácil acceso a la línea base del conocimiento a cada miembro, con guías, plantillas y mentores de herramientas para todas las actividades críticas de desarrollo. De esta forma, no importa en qué área se esté trabajando; ya sea requerimientos, diseño, pruebas, administración del proyecto o administración de configuración; todos los miembros del grupo comparten un lenguaje, procesos y vista de cómo desarrollar el software en común.

Cabe recalcar que este proceso crea y mantiene modelos, es decir, en vez de enfocarse en la producción de grandes cantidades de documentación, enfatiza el desarrollo y mantención de modelos – representaciones altamente semánticas de sistemas de software en desarrollo.

Esta metodología es básicamente una guía para cómo usar efectivamente el Lenguaje Unificado de Modelación (Unified Modeling Language – UML). **UML** es un lenguaje estándar que permite expresar claramente requerimientos, arquitectura y diseños. Fue originalmente creado por Rational Software y ahora es mantenido por la organización de estándares Object Management Group (OMG).

El Proceso Unificado Racional está soportado por herramientas que generalmente automatizan gran parte del proceso. Son usadas para crear y mantener los varios objetos – modelos en particular – del proceso de ingeniería de software: modelado visual, programación, pruebas, entre otros. Son invaluables para el soporte contable con la administración de cambios, así como con la administración de configuración en cada iteración.

Se puede decir que este proceso es configurable, es decir, "encaja" o va bien tanto con grupos pequeños de desarrollo (incluso 1 persona) cómo grupos grades (organizaciones y empresas). Se puede variar de acuerdo a cómo las situaciones lo requieran. Contiene un "Kit de Desarrollo" que provee soporte para configurar el proceso para adaptarse a las necesidades de quién o quienes lo usen.

2.2.1.2 Características Principales

RUP captura muchas de las mejores prácticas en el desarrollo de software moderno, de tal manera que se adapta a una amplia gama de proyectos y organizaciones. En otras palabras, describe cómo desplegar eficientemente al mercado alcances del desarrollo de software para grupos de desarrollo de software. Todo esto se denomina "buenas prácticas", no necesariamente porque se puedan calificar, sino porque son observadas para ser usadas en la industria por organizaciones exitosas, de manera común. Las buenas o mejores prácticas son:

1. Desarrollo Iterativo de Software
2. Manejo de Requerimientos
3. Uso de arquitecturas basadas en componentes
4. Modelo de Software Visual
5. Verificación de Calidad de Software
6. Control de Cambios en el Software

2.2.1.3 Fases e Iteraciones

Las fases e iteraciones de la metodología RUP se detallan en la *Figura 2.2-1*

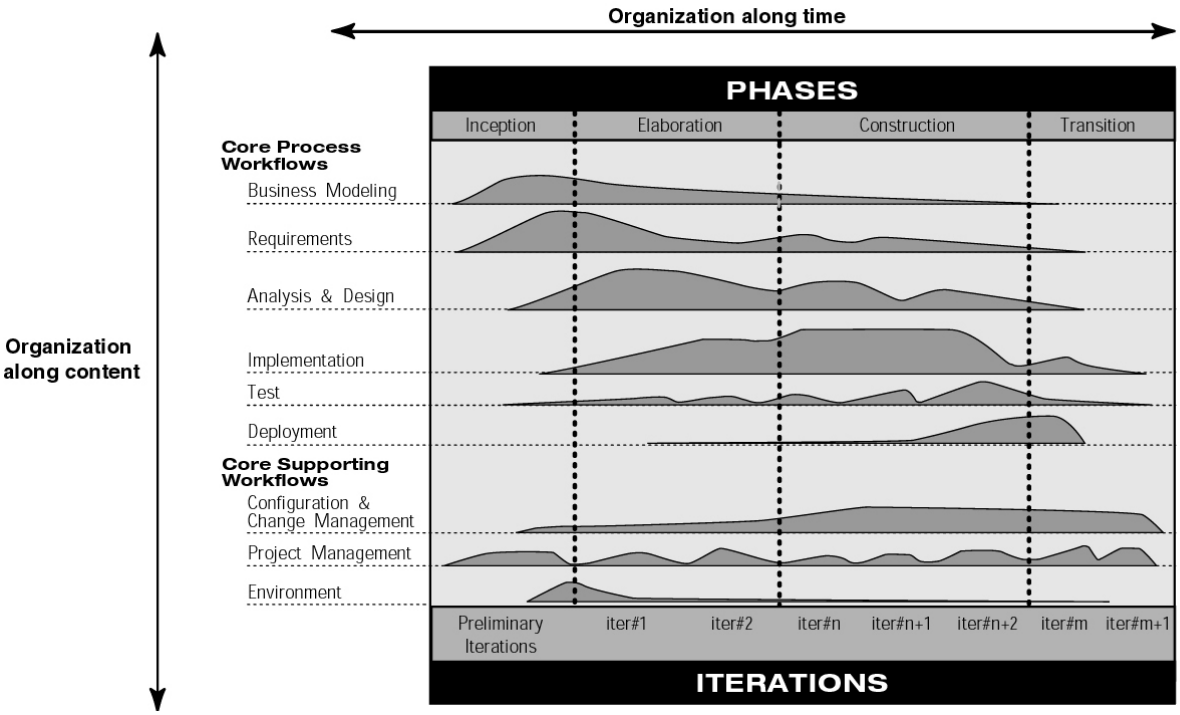

Figura 2.2-1 (Rational - The Software Development Company, 2001)

2.2.1.3.1 Fase de Incepción

En esta fase se establece el caso del negocio para el sistema y se delimita el alcance del proyecto. Para lograrlo, es necesario identificar todas las entidades eternas con las que el sistema va a interactuar (actores) y se define la naturaleza de esta interacción a alto nivel. Lo que implica identificar todos los casos de uso y describir algunos (los más significativos). El caso de negocio incluye criterio de éxito, evaluación de riesgos y estimación de recursos necesarios, y un plan de fase que muestre las fechas de las etapas principales.

Al final de la fase de incepción, los entregables son generalmente:

- Documento de visión: Visión general de requerimientos, características clave y restricciones principales.
- Modelo de casos de uso inicial
- Glosario de proyecto inicial
- Caso de negocio Inicial: Incluye contexto del negocio, criterio de éxito (proyección de ingresos, reconocimiento de mercado, entre otros) y pronóstico financiero
- Evaluación de riesgos inicial
- Plan del proyecto que muestre fases e iteraciones
- Modelo de negocio, en caso de ser necesario
- Uno o varios prototipos

La finalización de la fase de incepción se da el primer hito mayor del proyecto: Los objetivos hitos del ciclo de vida. Los criterios de evaluación para la fase de incepción son:

- Concurrencia de las partes interesadas en la definición de alcance y estimación de costos/programas (horarios).
- Evidencia de entendimiento de los requerimientos mediante la comprobación de los casos de uso primarios.
- Credibilidad de los estimados de costos/programas, prioridades, riesgos y proceso de desarrollo.
- Detalle y extensión de cualquier prototipo arquitectónico que se ha desarrollado.
- Gastos actuales Vs. Gastos planeados.

2.2.1.3.2 Fase de Elaboración

El propósito de la fase de elaboración es analizar el dominio del problema, establecer un fundamento arquitectónico, desarrollar el plan del proyecto y eliminar los elementos de mayor riesgo para el proyecto. Para lograr estos objetivos, se debe tener una vista del cuadro completo del sistema. Las decisiones sobre la arquitectura se deben hacer entendiendo el sistema completo: su alcance, requerimientos funcionales y no funcionales como requerimientos de rendimiento.

Es fácil argumentar que la fase de elaboración es la más crítica de las cuatro fases, Al final de esta fase, se considera que la "ingeniería difícil" se ha completado y que el proyecto ha pasado por el día más importante de consideración del proyecto: la decisión de continuar o no hacia las fases de construcción y transición. Mientras el proceso siempre tiene que hacer cambios, las actividades de la fase de elaboración aseguran que la arquitectura, requerimientos y planes estén suficientemente estables y que los riesgos estén suficientemente mitigados, de manera que se pueda determinar predeciblemente el costo y el programa para conclusión del desarrollo.

En la fase de elaboración un prototipo de arquitectura ejecutable se construye en una o más iteraciones, dependiendo del alcance, tamaño, riesgo y novedad del proyecto.

El resultado de la fase de elaboración es:

- Un modelo de caso de uso (completo por lo menos el 80%), en donde todos los casos de uso y actores han sido identificados y más casos de uso han sido elaborados.
- Requerimientos suplementarios capturando lo requerimientos no funcionales y cualquier requerimiento que no esté asociado con un caso de uso específico.
- Descripción de una Arquitectura de Software.
- Prototipo arquitectónico ejecutable.
- Lista de riesgos y casos de negocio revisados.
- Un plan de desarrollo para el proyecto global, incluyendo el plan del proyecto desglosado, mostrando iteraciones y criterios de evaluación para cada iteración.
- Un caso de desarrollo actualizado especificando el proceso que se usará.
- Un manual de usuario preliminar (opcional).

Al final de la fase de elaboración se da el segundo mayor hito del ciclo de vida. En este punto se examina los objetivos y alcance detallados del sistema, la selección de la arquitectura y la resolución de los mayores riesgos.

Los criterios de evaluación para esta fase deben responder las siguientes preguntas:

- ¿La visión del producto es estable?
- ¿La arquitectura es estable?
- ¿La demostración ejecutable muestra que los elementos de mayor riesgo han sido direccionados y realmente resueltos?
- ¿El plan para la construcción de la fase fue lo suficientemente detallado y preciso? ¿Está respaldado con una base de estimaciones creíble?
- ¿Todas las partes interesadas están de acuerdo en que la visión actual puede ser alcanzada si el plan actual se ejecuta para desarrollar el sistema completo, en el contexto de la arquitectura actual?
- ¿Los gastos actuales vs. los gastos planeados son aceptables?

2.2.1.3.3 Fase de Construcción

En la fase de construcción, todos los componentes que faltan y las características de la aplicación se desarrollan e integran en el producto, y todas las características se prueban. La fase de construcción es de cierto modo un proceso de manufactura que pone énfasis en manejar los recursos y controlar las operaciones para optimizar costos, programaciones y calidad. En este sentido, la mentalidad de administración sigue una transición desde el desarrollo de propiedad intelectual durante las fases de incepción y elaboración, hasta el desarrollo de un producto desplegable durante las fases de construcción y transición.

Muchos proyectos son lo suficientemente grandes para que los incrementos de la construcción en paralelo puedan ser generados. Estas actividades en paralelo pueden acelerar significativamente la capacidad de despliegues; pero también pueden incrementar la complejidad de administrar los recursos y sincronización del flujo de trabajo. Una arquitectura robusta es más fácil de construir. Esta es una razón por la cual el desarrollo balanceado de la arquitectura y del plan es enfatizado durante la fase de elaboración. La salida de la fase de construcción es un producto listo para ponerlo en las manos de los usuarios finales. Como mínimo consiste de:

- Producto de software integrado en la plataforma adecuada.
- Manuales de usuario.
- Descripción de la versión actual.

Al final de la fase de construcción se da el tercer hito mayor, en este punto se decide si el software, los sitios y los usuarios están listos para operar, sin exponer al proyecto a altos riesgos. Este hito comúnmente se denomina la **versión** "**beta**"[2].

El criterio de evaluación para la fase de construcción debe responder las siguientes preguntas:

- ¿La versión del producto es suficientemente estable y madura para ser desplegada a la comunidad de usuarios?
- ¿Están todas las partes interesadas listas para la transición en la comunidad de usuarios?
- ¿Los costos actuales vs. los costos planeados siguen siendo aceptables?

[2] Versión Beta: Software técnicamente acabado, lo cual significa que no se le añadirán de momento más funciones, y presumiblemente será lo suficientemente estable para trabajar con normalidad. (Ordenadores y Portátiles, s.f.)

2.2.1.3.4 Fase de Transición

El propósito de la fase de transición es justamente la transición del producto de software en la comunidad de usuarios. Una vez que el producto se ha entregado al usuario final, surgen inconvenientes y requieren nuevas versiones, corregir algunos problemas o terminar las características que fueron pospuestas.

La fase de transición se da cuando una línea base está suficientemente avanzada para ser desplegada en el dominio del usuario final. Esto requiere casi siempre que algunos subconjuntos utilizables del sistema se hayan completado en un nivel aceptable de calidad y que la documentación de usuario esté disponible de manera que la transición de resultados positivos para todas las partes.

Esto incluye:

- Pruebas "beta" para validar el nuevo sistema en contraste con las expectativas del usuario.
- Operación paralela con un sistema heredado al cual está reemplazando.
- Conversión de bases de datos operacionales.
- Capacitación a usuarios y personal de mantenimiento.
- Emprender el producto al mercado, distribución y grupos de venta.

La fase de transición se enfoca en las actividades requeridas para poder entregar el software a los usuarios. Típicamente, esta fase incluye algunas iteraciones, incluyendo versiones beta, versiones generales disponibles, **bug-fix** [3]y versiones de mejora. En este punto del ciclo de vida la retroalimentación de los usuarios debe ser tomada primariamente para ajustar, configurar, instalar y ver las dificultades de usabilidad del producto.

Los objetivos primarios de la fase de transición incluyen:

- Lograr que el usuario pueda usar el producto por sí mismo.
- Lograr que la concurrencia desplegada de las partes interesadas esté completa y consistente con el criterio de evaluación de la visión.
- Lograr la línea base del producto final tan rápida y económicamente efectiva como sea posible.

Al final de la fase de transición se da el cuarto hito mayor del proyecto. En este punto se decide si los objetivos fueron logrados y si se debería empezar otro ciclo de desarrollo.

Los criterios de evaluación para esta fase responden las siguientes preguntas:

- ¿El usuario está satisfecho?
- ¿Los costos actuales vs. los costos planeados siguen siendo aceptables?

2.2.1.4 Beneficios de un acercamiento iterativo

- Los riesgos son mitigados antes.
- Los cambios se pueden manejar de mejor manera.
- Mayor nivel de re-uso.
- El grupo del proyecto aprende a lo largo del camino
- Mejor calidad globalmente

[3] Bug-Fix: También denominado "depuración", consiste en localizar y corregir errores existentes en un programa informático. (Zenón J. Hernández Figueroa, 2009)

2.2.1.5 Historia Breve

La historia del RUP se resume en la *Figura 2.2-2*.

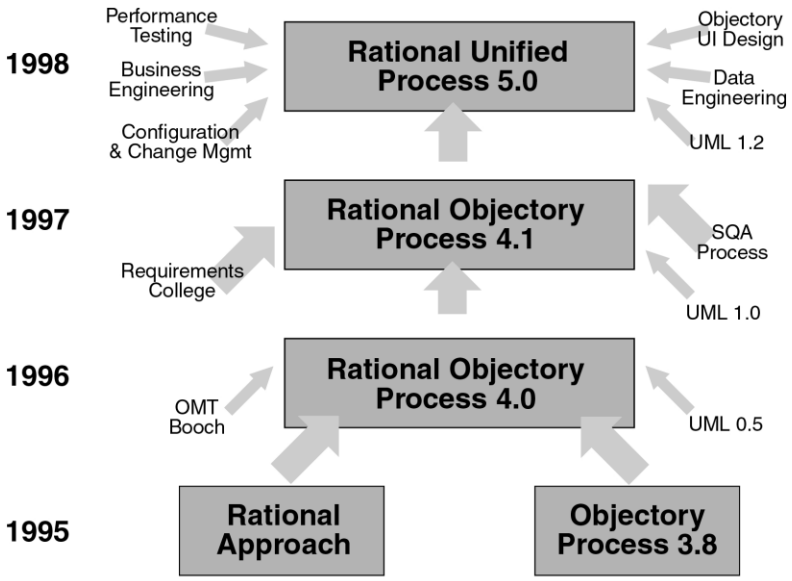

Figura 2.2-2 Genealogía del RUP (Rational - The Software Development Company, 2001)

2.2.2 Herramientas de Desarrollo

2.2.2.1 PHP

2.2.2.1.1 ¿Qué es?

PHP es un lenguaje de programación del lado del servidor que se usa para crear sitios web dinámicos y aplicaciones web interactivas. El acrónimo PHP originalmente venía dado por "Personal Home Page" pero a medida que sus funcionalidades fueron creciendo cambió a "PHP: Hyertext PreProcessor". Este acrónimo recursivo viene dado el hecho de que se necesita código PHP como entrada para producir HTML como salida. Esto significa que los usuarios no necesitan instalar ningún software para poder visualizar páginas web PHP generadas. Todo lo necesario es que el servidor web tenga instalado PHP para que pueda interpretar los scripts.

2.2.2.1.2 Características Principales

- En contraste con sitios HTML, los sitios PHP son generados dinámicamente.
- En vez de que un sitio sea hecho de numerosos archivos HTML, un sitio PHP consiste sólo de archivos de plantilla que describen sólo la estructura del sitio usando código PHP.
- El contenido de un sitio PHP es obtenido desde una base de datos.
- El formato de estilo para un sitio PHP se obtiene de un archivo **CSS**.
- PHP provee flexibilidad a un sitio web para que este sea fácil de diseñar, mantener y actualizar con nuevo contenido.

2.2.2.1.3 Historia

Antes de que PHP fuera inventado, había mucha actividad en el desarrollo web. Antes, el código de los scripts del lado del servidor era usualmente escrito en **C** o **Perl**, los cuales son lenguajes de programación generales que fueron adaptados para ser usados en Internet.

La realización original de PHP fue creada por Rasmus Lerdorf en junio de 1995, para hacer varias tareas de programación web fáciles y menos repetitivas. El nombre se entendía originalmente por "Personal Home Page" o "Página de Inicio Personal", pero ahora tiene un acrónimo recursivo: "PHP: Hypertext Preprocessor". El objetivo de ese cambio fue minimizar la cantidad de código requerida para lograr resultados, lo cual llevó a PHP a estar incorporado dentro de código HTML.

El segundo cambio de PHP, conocido como PHP/FI 2.0, fue el primero en lograr amplia difusión de popularidad, a pesar de tener inconsistencias de análisis, atrajo varias conversiones.

La realización de PHP 3 fue mayormente conducida por Zeev Suraski y Andi Gurmans, quienes reescribieron PHP desde cero y suprimieron los problemas analíticos. PHP 3 también hizo mucho más fácil a otras personas extender el lenguaje y escribir sus propios módulos, añadiendo funcionalidad a nivel central. A mediados del año 2000, PHP 3 fue el reemplazo de alrededor de 2.5 millones de **dominios**[4].

El sucesor, PHP 4, tenía una serie de modificaciones importantes, incluyendo el cambio a **Zend Engine**, una compañía fundada por Zev Suraski y Andi Gumans para promover PHP en el ambiente corporativo, con un motor que traía consigo numerosos beneficios. El motor Zend Engine introdujo conteo de referencias para asegurarse de que no hayan pérdidas en la memoria, así como abstracción a nivel del servidor para que PHP se ejecutara en **Apache** [5]1.3.x, Apache 2, **Microsoft's IIS**, **Zeus**, **AOLServer**, y más; y también cambió el modo en el que el código PHP era ejecutado de manera que se leyera una vez, se transformara a un formato interno y luego se ejecutara. Este nuevo paradigma de ejecución permitió el uso de cachés externos de código, también conocidos como "aceleradores PHP" que daban mayor aumento del rendimiento.

El cambio de PHP 4 a PHP 5 sigue siendo uno de los mayores. Viene dado con mejoras en orientación a objetos, la adición de manejo de errores con try/catch y excepciones; hay dos extensiones nuevas principales: **SimpleXML**, un rápido y fácil de aprender modo de interactuar con documentos **XML**[6], y **SQLite**, una nueva **API**[7] de archivos planos de base de datos que facilita el peso de desplegar soluciones de bases de datos simples.

[4] Dominio: Red de identificación asociada a un grupo de dispositivos o equipos conectados en Internet. (Gallardo, 2015)
[5] Apache: El Servidor HTTP Apache es un servidor web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1. (Romero, 2015)
[6] XML: Es un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible. (Miguel, 2015)
[7] API: Interfaz de programación de aplicaciones. (Wikipedia, s.f.)

2.2.2.2 Licenciamiento de Software

2.2.2.2.1 ¿Qué es?

El concepto de licencia de software se desarrolló por la industria de software desde su temprano control. Una licencia de software es el permiso de una licencia por parte del publicador del software para usar una o más copias del software, pero la propiedad le sigue perteneciendo al publicador.

2.2.2.2.2 Tipos de licencias propietario

Existen varios tipos de licencias de acuerdo al uso que se le vaya a dar al software y al usuario al cual está dirigido.

- Licencias individuales (Single-User): Esta licencia permite que el software sea usado en sólo una computadora, a la cual no pueden acceder otros usuarios por medio de una red. Los otros usuarios no tienen permiso de usar el software mientras estén conectados a la computadora que tiene el mismo. Los tipos de licencias individuales son:
    - Licencia perpetua: Permite al cliente instalar y usar el software por tiempo indefinido sin ninguna limitación.
    - Licencia por suscripción: Permite al usuario usar el software por un período de tiempo específico. Al final del período, el usuario tiene varias opciones renovar la suscripción, comprar una licencia perpetua con descuento o desinstalar el software de la computadora.
    - Licencia de prueba: Permite a los clientes usar el software por un período de tiempo (como 30 días por ejemplo) cómo una estrategia de marketing.
    - Licencia Demo: Permite al usuario final ver una demostración del software con funcionalidad parcial o por cierto período de tiempo.
    - Licencia basada en características: Permite al usuario final usar sólo ciertas características del software, en vez de todo el paquete de funcionalidades.
    - Licencia de alquiler con tiempo límite: Permite al usuario hacer un uso prepago del software.
- Licencias de Red o Multiusuarios
    - Licencia por Servidor (Red): Este tipo de licencia requiere que en el servidor de archivos exista una copia del software.
    - Licencia por Asiento (Máquina): Esta licencia requiere que la compra sea para cada computadora o equipo que necesite el software. Es típicamente usada en conjunto con una licencia por servidor.
- Add-on’s a licencias nuevas o existentes
    - Actualización: Licencia adquirida cuando un usuario ya tiene una licencia de software y quiere una versión más nueva.
    - Uso de estudiante: Permite a estudiantes usar el software durante el tiempo que sean estudiantes de determinada institución. Tienen que desinstalar el software una vez que dejen o terminen de estudiar.
    - Uso secundario: Permite al usuario final que tenga ya una licencia, usar esa licencia en una segunda computadora.
    - Derechos de trabajo en casa: Permite usar a un empleado el software del trabajo en casa. Si el trabajo finaliza, el beneficio de uso del software también.

2.2.2.2.3 Piratería

Los piratas pueden ser denominados como criptoanalistas quienes descifran los parches de cierto CD de software y la key o clave de la licencia disponible con ese CD. Cada CD de software tiene una clave de licencia única consigo. Los piratas toman algunos CD's, encuentran el parche correspondiente y hacen copias del conjunto para venderlo ilegalmente en el mercado.

Existen varios tipos de piratería como:

- Cracks y seriales: Obtención legal de una versión de prueba de software y seguidamente ingresando códigos de una licencia copiada o aplicando un parche genérico que sobrepasa la protección de una copia.
- Softlifting: Ocurre cuando una persona u organización compra una licencia individual de un programa y lo instala en varias computadoras.
- Acceso de Clientes Sin Restricción: Ocurre cuando una copia de un programa es copiada en los servidores de una organización y los clientes de la red son autorizados para acceder libremente al software.
- Carga del Disco Duro: Ocurre cuando un individuo o compañía venden computadoras con copias de software ilegales pre-instaladas.
- OEM Piratería/Disociación: El software OEM (Original Equipment Manufacturer) es aquel que es legalmente vendido sólo con hardware específico. Cuando estos programas son copiados o vendidos separadamente del hardware, se da una violación de distribución del contrato entre el vendedor y el publicador de software.
- Uso no autorizado de Software Académico: Muchas compañías de software venden versiones académicas de sus productos a escuelas públicas universidades y otras instituciones educativas. Usar este software académico en violación de la licencia de software es una forma de piratería.
- Falsificación: es la duplicación y venta de copias no autorizadas de software de tal manera que se hace pasar la copia ilegal como legal.
- CD-R Piracy: Copia y distribución ilegal de software usado tecnología de grabación CD-R.
- Piratería de descargas: Se da cuando se sube software a un sitio de Internet para cualquier persona. Cualquier persona que carga o descarga software, está haciendo una copia ilegal del mismo.
- Renta: Rentar software por uso temporal (En Estados Unidos y Canadá).

2.2.2.2.4 Software Libre y Open Source

El software Open Source es un software cuyo **código fuente**[8] está disponible abiertamente para cualquier persona que desee inspeccionarlo y estudiarlo. La mayoría de software open source también es software libre.

La definición de software libre estipula los criterios que se tienen que cumplir para que un programa sea considerado libre. De vez en cuando modificamos esta definición para clarificarla o para resolver problemas sobre cuestiones delicadas. (Free Software Foundation, Inc., 2016)

Ya que los términos de software open source y software libre son típicamente usados para describir los mismos programas, son comúnmente confundidos como si fueran lo mismo. Y es muy usual que la

[8] Código Fuente: versión de software escrita por un humano.

diferencia entre ellos sea más filosófica que práctica. Sin embargo, eventos recientes han enfatizado que sí existe una diferencia práctica entre estos conceptos.

La diferencia existente entre software libre y open source se da porque el software libre siempre será open source, pero el open source no necesariamente es software libre. Lo cual implica que el software puede ser open source sin necesidad de dar a sus usuarios permisos de libertades adicionales que sí se dan en software libre.

Por ejemplo, si Microsoft hubiera tenido verdaderamente la oportunidad de demandar exitosamente a usuarios Linux por supuestas violaciones a su propiedad intelectual, esto hubiera significado que Linux no sería software libre, porque los usuarios no tendrían la libertad de usarlo de la manera que deseen sin restricciones monetarias u otras restricciones. Sin embargo, seguiría siendo open source, porque al menos está disponible para inspeccionarlo y estudiarlo.

2.2.2.3 Introducción a Bases de Datos

2.2.2.3.1 ¿Qué son?

El término datos puede ser definido como un grupo de hechos aislados y no relacionados con un significado implícito. Cuando estos datos son procesados y convertidos en algo con sentido y que pueda usarse, se conoce como información. Una base de datos puede ser definida entonces como una colección de datos relacionados de la cual los usuarios pueden recoger información que deseen.

En una base de datos, los datos son integrados y compartidos. Por integrados entendemos que la base de datos puede ser interpretada como la unificación de varios archivos, con cualquier redundancia entre los mismos ya sea parcial o totalmente eliminada. Por compartida nos referimos a que la base de datos puede ser compartida entre diferentes usuarios, en el mismo sentido que diferentes usuarios pueden tener acceso al mismo contenido e información, incluso posiblemente hasta al mismo tiempo (acceso concurrente).

Una base de datos consiste tanto de hardware como de software. En el hardware podemos encontrar los discos de almacenamiento secundario (generalmente discos magnéticos) que son usados para mantener almacenada la información, junto con los dispositivos I/O (dispositivos de entrada y salida), controladores, canales, entre otros. También tenemos el procesador o procesadores y una memoria principal asociada, los cuales tienen la función de soportar la ejecución del software del sistema de la base de datos.

El software de una base de datos es la capa intermediaria entre los datos físicamente almacenados y los usuarios. Se lo conoce de algunas maneras; el administrador de la base de datos o el servidor de la base de datos, pero de manera más común como el sistema de administración de base de datos (DBMS). Todas las peticiones de acceso a la base son manejadas por el DBMS.

El sistema de manejo o administración de una base de datos, DBMS (*Figura 2.2-3*) por sus siglas en inglés “Database Management System”, tiene como objetivo principal proveer un método conveniente y efectivo de definición, almacenamiento, recolección y manipulación de datos contenidos en una base de datos. De manera que la base de datos en conjunto con el software de DBMS son conocidos como un sistema de base de datos. Los usuarios del sistema pueden realizar una variedad de operaciones con la base de datos como por ejemplo:

- Añadir nuevos archivos a la base de datos

- Insertar datos dentro de archivos ya existentes
- Recolectar datos de archivos existentes
- Eliminar datos de archivos existentes
- Cambiar los datos de archivos existentes
- Eliminar archivos existentes de la base de datos

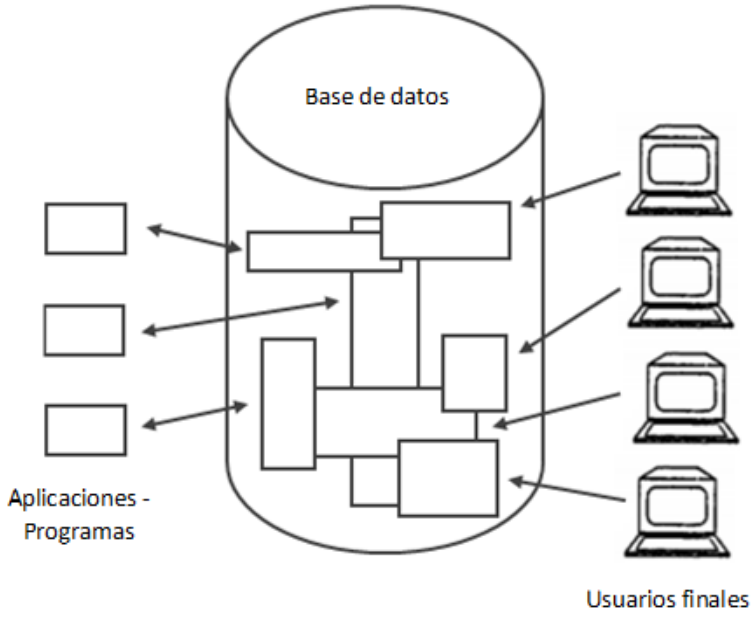

Figura 2.2-3 Imagen simplificada de DBMS (Date, 2004)

Se hizo referencia a "usuarios" en las bases de datos, pero, ¿qué son realmente los usuarios en este contexto? Los usuarios de una base de datos pueden dividirse en 3 ramas:

- Programadores de aplicaciones: Responsables de escribir aplicaciones de la base de datos en cualquier lenguaje de programación. Estos programas pueden acceder a la base de datos mediante las solicitudes correspondientes al DBMS.
- Usuarios finales: Usuarios que acceden a la base interactivamente mediante aplicaciones creadas para este fin.
- Administrador de la base de datos (DBA por sus siglas en inglés "Database Administrator"): Se encarga de crear la base de datos y poner en su lugar los controles técnicos necesitados para cumplir las políticas y decisiones impuestas por el administrador de datos o dueño de la base de datos. También se asegura de que el sistema opere con el rendimiento adecuado y que sea capaz de proveer una variedad de servicios técnicos. En resumen, es el encargado de administrar y mantener una base de datos confiable, consistente y segura.

2.2.2.3.2 Historia

De lo antiguo a lo moderno

- Las personas almacenaban información desde hace mucho tiempo.
- En tiempos antiguos, sistemas de bases de datos eran desarrollados por oficinas del gobierno, bibliotecas, hospitales y organizaciones de negocios.

- Algunos de los conceptos y principios siguen siendo usados en el presente.

1960

- La introducción del término "base de datos" coincide con la disponibilidad de almacenamiento de acceso directo (discos y tambores magnéticos) a mediados del año 1960.
- La primera generación de sistemas de base de datos era de navegación, es decir, las aplicaciones accedían a un dato mediante punteros, de un registro a otro.
- El usuario necesitaba conocer la estructura física del sistema para poder acceder a la información.
- IBM introdujo el sistema SABRE para ayudar a American Airlines a administrar su información de reservaciones. Aún es utilizado.

1970

- El modelo de base de datos relacional fue concebido por E.F. Codd en 1970.
- Este modelo establecía que las aplicaciones deberían buscar los datos por contenido en vez de buscarlos mediante el seguimiento de vínculos.
- Su sistema puede ser definido usando dos términos:
- Instancia y Esquema: Una tabla con filas y columnas donde se especifica la estructura, incluyendo nombre de la relación, nombre y tipo de cada columna
- El modelo de Codd se basó en ramas de la matemática llamada teoría de grupos y lógica del predicado.
- En su modelo el esquema de la base o la organización lógica se separa de del almacenamiento físico de la información, y esto se transformó en un principio estándar para sistemas de bases de datos.
- INGRES se desarrolló en la Universidad de California-Berkeley y se comercializó, seguido de POSTGRES el cual fue incorporado en Informix. Esto dio paso a Ingres Corp., Sybase, MS SQL Server, Britton-Lee, y Wang's PACE. Este sistema usaba QUEL como lenguaje de query.
- IBM desarrolló System R en San Jose y permitió que SQL/DS / DB2, ORACLE, HP's Allbase y Tandem's Non-Stop SQL DB2 se convirtieran en el primer producto DBMS basado en un modelo relacional.
- Un nuevo modelo llamado Entidad-Relación, o ER, fue propuesto por P. Chen en 1976. Este modelo hizo posible a los diseñadores enfocarse en aplicaciones de datos en vez de la estructura lógica.

1980

- La comercialización de sistemas relacionales se vuelve famosa en el campo de las bases de datos. SQL se convierte en el lenguaje estándar. DB2 se vuelve el producto principal de IBM. Modelos basados en redes y jerárquicos pierden popularidad.
- En este período aparece el concepto de bases de datos orientadas a objetos. Un objeto de base de datos es uno en el cual la información es representada en forma de objetos, así como son usados en la programación orientada a objetos. Las bases orientadas a objetos son diferentes de las bases relacionales y juntas forman parte de un sistema de administración de base de datos más amplio. Las bases de datos orientadas a objetos están designadas para trabajar correctamente en conjunto con lenguajes de programación orientados a objetos, **OODBMS**

(Object-Oriented Database Management System) tiene su propio lenguaje de programación y soporta modelación y creación de datos como objetos.

- OODBMS puede manejar eficientemente un número extenso de distintos tipos de datos. Los objetos de comportamientos complejos son fáciles de manejar usando **herencia** [9] y **polimorfismo**[10]. Esto ayuda a reducir un gran número de relaciones al crear objetos. El mayor problema con OODBMS fue intercambiar una base de datos existente a una base OODBMS, ya que la transición requiere un cambio completo desde cero y está ligado típicamente a un lenguaje de programación específico y a una API, lo cual reduce la flexibilidad de la base de datos. Para solucionar estos problemas se creó el Modelo de Bases de Datos Objeto-Relacional a inicios de 1990.

1990

- La mayoría del desarrollo en esta época se centró en herramientas para los clientes para el desarrollo de aplicaciones, por ejemplo **PowerBuilder** [11](Sybase), **Oracle Developer** [12]y **VB**[13] (Microsoft). El modelo cliente-servidor se convirtió en la norma para futuras decisiones de negocios. En este período también se marcó el desarrollo de herramientas de productividad personal, como por ejemplo **Excel**[14]**/Access** [15](MS) y **ODBC** [16](Open Database Connectivity).
- A mediados de 1990, apareció el Internet y consigo el acceso remoto a sistemas computacionales con información. De forma el modelo cliente-servidor alcanzó los escritorios de usuarios promedio con poca paciencia para cosas complejas, mientras la Web y las bases de datos crecía exponencialmente.
- Cómo se mencionó anteriormente, ORDBMS (Object Relational Database Managemente System) fue la solución a los problemas de OODBMS. El propósito principal de ORDBMS fue crear un puente entre los modelos de bases de datos relacionales y bases de datos orientadas a objetos.
- Otro desarrolló importante se dio en 1997, con la introducción de Extensible Markup Language (XML); lenguaje que define un grupo de reglas para codificar documentos en un formato entendible tanto para una persona como para la máquina.

Siglo XXI

- Tres compañías principales dominan el mercado de las bases de datos: IBM, Microsoft y Oracle.
- Las bases de datos hacen un excelente trabajo almacenando información y haciendo recuento, pero siguen siendo dependientes de la interacción humana o de algún programa externo para hacer correlaciones y decisiones basadas en la información. Por lo cual, aún ahora, hay una

[9] Herencia: Cuando en POO, una clase nueva se crea a partir de una clase existente. (Kioskea.net, 2014)
[10] Polimorfismo: El polimorfismo se refiere a la propiedad por la que es posible enviar mensajes sintácticamente iguales a objetos de tipos distintos. (Wikipedia, s.f.)
[11] PowerBuilder: Herramienta que permite el desarrollo de diferentes tipos de aplicaciones y componentes para ejecutar arquitecturas cliente/servidor, distribuidas y Web. (Wikipedia, s.f.)
[12] Oracle Developer: Entorno integrado de desarrollo para trabajar con SQL en bases de datos Oracle.
[13] VB: Visual Basic, lenguaje de programación, así como entorno integrado de desarrollo.
[14] Excel: Hoja de cálculo de Microsoft Office.
[15] Access: Gestor de base de datos de Microsot Ofice.
[16] ODBC: Open DataBase Connectivity. Estándar de acceso a bases de datos desarrollado por SQL Access Group. (Wikipedia, s.f.)

tendencia creciente para proveer lógica de programación más sofisticada en conjunto con la estructura de una base de datos.

- También apareció el concepto de **big data**, sistemas con terabytes de información que requieren novedosos conceptos para manejar y analizar una cantidad inmensa de datos, como por ejemplo bases de datos del **proyecto Genoma** [17] , proyectos geológicos, de seguridad nacional, de exploración de datos del espacio. En la actualidad **Data mining** [18] o minería de datos, **data warehousing**[19] y **data marts**[20] son técnicas comúnmente usadas.

.

2.2.2.3.3 Niveles de Abstracción de la arquitectura de un DBMS

La idea de tener una arquitectura en un DBMS es brindar un marco de trabajo en el cual se puedan construir niveles subsecuentes de una base de datos. La **arquitectura ANSI/SPARC** está dividida en tres niveles: nivel interno o físico, nivel externo o conceptual y nivel lógico (*Figura 2.2-4*).

El **nivel conceptual** muestra la estructura de una base para un usuario o grupo de usuarios individuales, que puede ser un programador de aplicación o un usuario final. Se encarga de manejar la manera en la que la información va a ser vista por usuarios individuales. De esta forma, el usuario tiene limitaciones con respecto a detalles de las estructuras físicas de almacenamiento.

El **nivel lógico** es una representación de todo el contenido de información de la base, este nivel tiene la intención de ser una vista de los datos "como realmente son", describiendo varios esquemas o vistas para usuarios.

El **nivel físico** es una representación de bajo nivel de la base de datos completa, consiste en varias estructuras de varios tipos de almacenamiento interno. Está descrito por concepto de esquemas internos, lo cual no sólo define los varios tipos de registros almacenados sino también qué índices existen, como se representan los campos, en que secuencia física se encuentran los registros almacenados, entre otros.

[17] Proyecto Genoma: Proyecto de investigación biomédica.
[18] Data Mining: Minería de datos. Extracción de información oculta y predecible de grandes bases de datos. (I., 2004)
[19] Data Warehousing: Colección de datos orientada a un determinado. (Ángel Sastre Castillo, 2009)
[20] Data Marts: Versión especial de almacén de data warehouse.

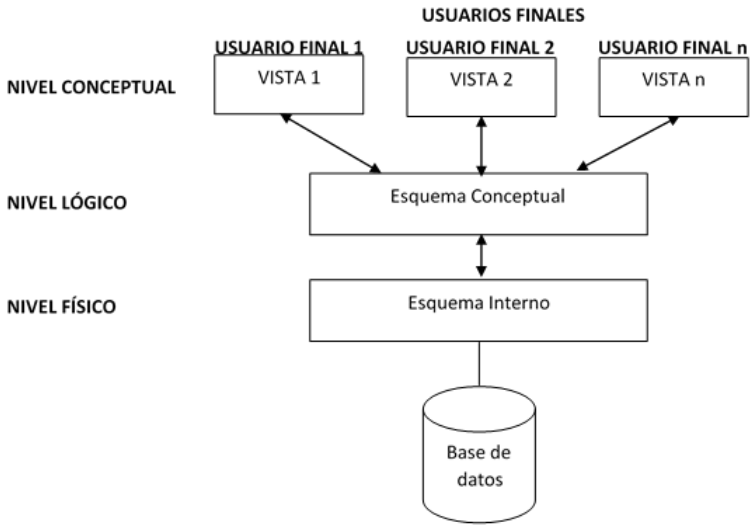

Figura 2.2-4 Arquitectura DBMS (ITL Education Solutions Limited, 2010)

2.2.2.3.4 Lenguajes de un DBMS

El principal objetivo de un DBMS es permitir a sus usuarios completar una serie de operaciones en la base de datos como insertar, borrar, obtener o actualizar información en términos abstractos, es decir, sin necesidad de saber las representaciones físicas de los datos. Para proveer estas facilidades a los distintos tipos de usuarios, un DBMS normalmente tiene uno o más lenguajes especializados de programación llamados **Lenguajes de Base de Datos o Database Languages**.

El DMBS usualmente trabaja con dos lenguajes:

- **Data Definition Language (DDL):** El lenguaje de definición de datos es usado para definir los esquemas físico y lógico de la base de datos. Sin embargo, en DBMS's en los cuales se mantiene una separación entre los niveles físico y lógico, el DDL es usado para especificar el nivel lógico únicamente; de manera que se usa un lenguaje alterno llamado Lenguaje de Definición de Almacenamiento, por sus siglas en inglés SDL "Storage Definition Language", para definir el esquema físico. Algunos de los DBMS's que están basados en la arquitectura ANSI/SPARC usan un tercer lenguaje llamado Lenguaje de Definición de Vistas, VDL "View Definition Language" para definir el esquema conceptual.
  Las sentencias DDL son usadas también para especificar las reglas de integridad (**constraints**) con el fin de mantener integridad en la base de datos. Las varias constraints de integridad son constraints de dominio, integridad referencial, confirmación y autorización.
  Como en cualquier otro lenguaje de programación, el DDL también acepta entradas o sentencias en forma de instrucciones y genera la descripción del esquema como salida. Esta salida es colocada en el **diccionario de datos** que es una tabla especial que contiene **metadatos**[21]. Hay que tener en cuenta que los usuarios de la base de datos no pueden modificar el diccionario de datos, este sólo es modificado por sí mismo por el sistema de la base de datos.

[21] Metadatos: Datos que describen otros datos.

- **Data Manipulation Languaje (DML):** Una vez que los esquemas de la base de datos son definidos y los datos iniciales son cargados en la base de datos, algunas operaciones como obtención, inserción, eliminación y modificaciones pueden ser aplicadas a la base. El DBMS provee un Lenguaje de Manipulación de Datos (DML) que permite a los usuarios obtener y manipular los datos. La sentencia que es usada para obtener la información se llama "**query**". El DML está dividido en dos tipos; procedural y no procedural.
  El DML no procedural, DML de alto nivel o DML declarativo permite expresar de manera concisa las operaciones complejas de la base de datos. Esto requiere que un usuario especifique que datos se requieren sin especificar cómo quiere recibir la información necesitada. Por ejemplo, **SQL (Structured Query Language)** es un DML no procedural que permite al usuario de manera fácil definir o modificar la estructura y los datos de la base. Las sentencias DML de alto nivel también pueden estar dentro de un lenguaje de programación de propósitos generales.
  Por otro lado el DML procedural o DML de bajo nivel requiere que el usuario especifique qué información necesita y cómo quiere acceder a los datos mediante un proceso paso por paso. Por ejemplo, el **álgebra relacional** es un DML procedural, el cual consiste en un grupo de operaciones como select, project, union, entre otros, para manipular los datos en la base.

Aparte de los lenguajes DDL y DML, también están los lenguajes Lenguaje de Control de Datos DCL, por sus siglas en inglés "Data Control Language", y Lenguaje de Control de Transacciones TCL, por sus siglas en inglés "Transaction Control Language". DCL es usado para crear nuevos roles, otorgar permisos y controlar el acceso a la base de datos mediante seguridad. Mientras que TCL sirve para manejar distintas transacciones que ocurren dentro de la base de datos.

2.2.2.3.5 Algunos DBMS

Actualmente existen más de 200 DBMS en el mercado, algunos con licencia gratuita y otros no. Según la página DB-Engines, el ranking de DBMS (actualizado mensualmente) está liderado por Oracle y la Tabla 2.2-1 muestra el Top 10 de los sistemas. El método de puntuación de los sistemas toma en cuenta:

- Número de menciones del sistema en sitios web
- Interés general en el sistema
- Frecuencia de discusiones técnicas acerca del sistema
- Número de ofertas de trabajo en las cuales el sistema es mencionado
- Número de perfiles en redes profesionales en las cuales el sistema es mencionado
- Relevancia en redes sociales

Es decir, se calcula el valor de popularidad del sistema estandarizando y promediando los parámetros individuales mencionados. Este ranking no mide el número de instalaciones de los sistemas, o su uso dentro de sistemas de TI (Tecnología e Información). Esta tendencia de popularidad se puede visualizar en la *Figura 2.2-5*

Tabla 2.2-1 Tabla Ranking DBMS (DB-Engines, 2016)

Ranking			DBMS	Modelo de la Base de Datos	Puntuación		
Febrero 2016	**Enero 2016**	**Febrero 2015**			**Febrero 2016**	**Enero 2016**	**Febrero 2015**
1	1	1	Oracle	DBMS Relacional	1476.14	-19.94	36.42
2	2	2	MySQL	DBMS Relacional	1321.13	21.87	48.67
3	3	3	Microsoft SQL Server	DBMS Relacional	1150.23	6.16	-27.26
4	4	4	MongoDB	Almacenamiento de documentos	305.6	-0.43	38.36
5	5	5	PostgreSQL	DBMS Relacional	288.66	6.26	26.32
6	6	6	DB2	DBMS Relacional	194.48	-1.89	-7.94
7	7	7	Microsoft Access	DBMS Relacional	133.08	-0.96	-7.47
8	8	8	Cassandra	Wide column store	131.76	0.81	24.68
9	9	9	SQLite	DBMS Relacional	106.78	3.04	7.22
10	10	10	Redis	Key-value store	102.07	0.92	2.86

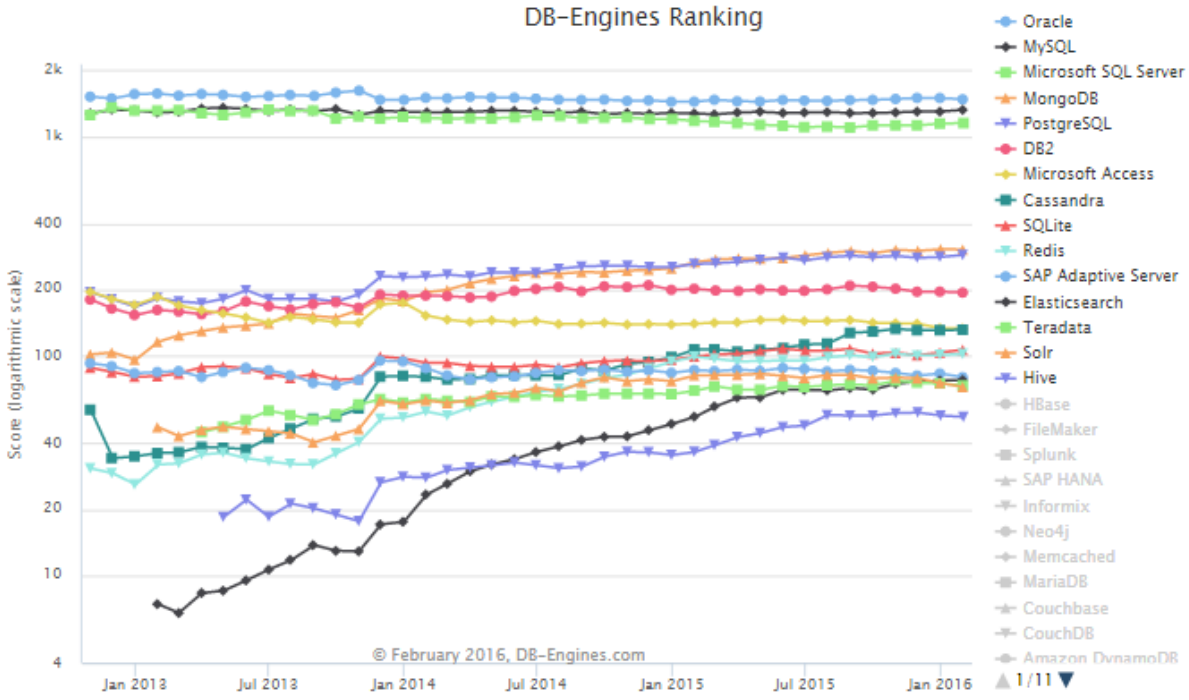

Figura 2.2-5 Tendencia de popularidad de los DBMS (DB-Engines, 2016)

Las características principales de los 10 DBMS mejor puntuados y su comparación se muestran en la Tabla 2.2-2.

Tabla 2.2-2 Comparación de Propiedades de los Sistemas (DB-Engines, 2016)

Nombre	Microsoft SQL Server X	MongoDB X	MySQL X	Oracle X	PostgreSQL X	Cassandra X	DB2 X	Microsoft Access X	SQLite X	Redis X
Descripción	RDBMS de Microsoft	Uno de los almacenadores más populares de documentos	DBMS Relacional, Open source usada mundialmente	DBMS Relacional usada mundialmente	Basada en el objeto relacional DBMS Postgres	Almacenamiento basado en ideas de BigTable y DynamoDB	Común en ambientes host de IBM, 2 versiones distintas para el host y para Windows/Linux	Microsoft Access combina RDBMS (JET / ACE Engine) con una GUI para manipular datos y queries	RDBMS in-process usado mundialmente	Estructura de almacenamiento de datos In-memory, usada como base de datos, cache y message broker.
Modelo de base de datos	DBMS Relacional	Almacenamiento de documentos	DBMS Relacional	DBMS Relacional	DBMS Relacional	Wide column store	DBMS Relacional	DBMS Relacional	DBMS Relacional	Key-value store
Desarrollador	Microsoft	MongoDB, Inc.	Oracle	Oracle	PostgreSQL Global Development Group	Apache Software Foundation	IBM	Microsoft	Dwayne Richard Hipp	Salvatore Sanfilippo
Versión Inicial	1989	2009	1995	1980	1989	2008	1983	1992	2000	2009
Versión Actual	SQL Server 2014, Abril 2014	3.2.1, Enero 2016	5.7.10, Diciembre 2015	12 Release 1 (12.1.0.2), Julio 2014	9.5, Enero 2016	3.2.1, Enero 2016	DB2 Data Server (10.5), Abril 2013	2016 (16.0.4229.1024), Septiembre 2015	3.10.2, Enero 2016	3.0.7, Enero 2016
Licencia	Comercial	Open Source	Open Source	Comercial	Open Source	Open Source	Comercial	Comercial	Open Source	Open Source

Nombre	Microsoft SQL Server X	MongoDB X	MySQL X	Oracle X	PostgreSQL X	Cassandra X	DB2 X	Microsoft Access X	SQLite X	Redis X
Base como Servicio (DBaaS)	no	no	no	no	no	no	no	no	no	no
Sistemas Operativos como Servidores	Windows	Linux OS X Solaris Windows	FreeBSD Linux OS X Solaris Windows	AIX HP-UX Linux OS X Solaris Windows z/OS	FreeBSD HP-UX Linux NetBSD OpenBSD OS X Solaris Unix Windows	BSD Linux OS X Windows	Linux Unix Windows z/OS	Windows	server-less	BSD Linux OS X Windows
Server-side scripts	Transact SQL and .NET languages	no	si	PL/SQL	Funciones definidas por el usuario	no	si	si	no	Lua
Esquema de datos	si	schema-free	si	si	si	schema-free	si	si	si	schema-free
Índices Secundarios	si	si	si	si	si	restringido	si	si	si	no
SQL	si	no	si	si	si	no	si	si	si	no

Nombre	Microsoft SQL Server X	MongoDB X	MySQL X	Oracle X	PostgreSQL X	Cassandra X	DB2 X	Microsoft Access X	SQLite X	Redis X
Lenguajes de Implementación	C++	C++	C and C++	C and C++	C	Java	C and C++	C++	C	C
APIs y otros métodos de acceso	OLE DB Tabular Data Stream (TDS)	proprietary protocol usando JSON	ADO.NET JDBC ODBC	ODP.NET Oracle Call Interface (OCI) JDBC ODBC	native C library streaming API for large objects ADO.NET JDBC ODBC	Proprietary protocol	JSON style queries XQuery ADO.NET JDBC ODBC	DAO OLE DB ADO.NET ODBC	ADO.NET JDBC ODBC	proprietary protocol
Concurrencia	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí
Durabilidad	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí
Métodos de Partición	Las tablas se pueden distribuir a lo largo de varios archivos (partición horizontal)	Replicación Master-slave	Partición horizontal, fragmentación con MySQL Cluster o MySQL Fabric	Partición horizontal	No, pero se puede realizar usando herencia de tablas.	Fragmen-tación	Fragmen-tación	Ninguna	Ninguna	Fragmen-tación

Nombre	Microsoft SQL Server X	MongoDB X	MySQL X	Oracle X	PostgreSQL X	Cassandra X	DB2 X	Microsoft Access X	SQLite X	Redis X
Métodos de replicación	Sí pero dependiendo de la edición de SQL-Server	si	Replicación Master-master Replicación Master-slave	Replicación Master-master Replicación Master-slave	Replica-ción Master-slave	Factor de replicación seleccio-nable	si	Ninguna	Ninguna	Replica-ción Master-slave
Lenguajes de Programación soportados	.Net Java PHP Python Ruby Visual Basic	ActionScript C C# C++ Clojure ColdFusion D Dart Delphi Erlang Go Groovy Haskell Java JavaScript Lisp Lua MatLab Perl PHP PowerShell Prolog Python R Ruby Scala SmallTalk JavaScript	Ada C C# C++ D Eiffel Erlang Haskell Java Objective-C OCaml Perl PHP Python Ruby Scheme Tcl	C C# C++ Clojure Cobol Eiffel Erlang Fortran Groovy Haskell Java JavaScript Lisp ObjectiveC OCalm Perl PHP Python R Ruby Scala Tcl Visual Basic	.Net C C++ Java Perl Python Tcl	C# C++ Clojure Erlang Go Haskell Java JavaScript Perl PHP Python Ruby Scala	C C# C++ Cobol Fortran Java Perl PHP Python Ruby Visual Basic	C C# C++ Java (JDBC-ODBC) VBA Visual Basic.NET	Actionscript Ada Basic C C# C++ D Delphi Forth Fortran Haskell Java JavaScript Lisp Lua MatLab Objective-C OCaml Perl PHP PL/SQL Python R Ruby Scala Scheme Smalltalk Tcl	C C# C++ Clojure Crystal D Dart Elixir Erlang Fancy Go Haskell Haxe Java JavaScript (Node.js) Lisp Lua MatLab Objective-C OCaml Perl PHP Prolog Pure Data Python R Rebol Ruby Rust Scala Scheme Smalltalk Tcl

Nombre	Microsoft SQL Server X	MongoDB X	MySQL X	Oracle X	PostgreSQL X	Cassandra X	DB2 X	Microsoft Access X	SQLite X	Redis X
Triggers	si	Sharding	si	si	si	si	si	si	si	no
MapReduce	no	Consistencia Eventual	no	no	no	si	no	no	no	no
Conceptos de consistencia	Inmediata	Inmediata		Inmediata	Inmediata	Eventual Inmediata				Eventual
Foreign keys	Sí	no	Sí	Sí	Sí	no	Sí	Sí	Sí	no
Conceptos de Transacción	ACID	no	ACID	ACID	ACID	no	ACID	ACID	ACID	Optimistic locking, ejecución atómica de bloques de comandos y scripts
Capacidades In-Memory		Sí	Sí	Sí	no				Sí	Sí
Conceptos de Usuarios	fine grained access rights according to SQL-standard	Access rights for users and roles	Users with fine-grained authorization concept	fine grained access rights according to SQL-standard	fine grained access rights according to SQL-standard	Access rights for users can be defined per object	fine grained access rights according to SQL-standard	no	no	Simple password-based access control

2.2.2.4 MySQL

2.2.2.4.1 ¿Qué es?

Una base de datos relacional (RDBMS) es una herramienta esencial en muchos sentidos, desde su uso en negocios, investigación y contextos educativos, hasta distribución de contenido en el Internet. Históricamente, los sistemas de bases de datos han sido recursos costosos, con tarifas por software, hardware y mantenimiento. Los Tiempos han cambiado tanto en hardware como en software.

Las bases de datos se han vuelto más accesibles y DBMS's open source están disponibles en el mercado. Uno de los más importantes DBMS's es MySQL, un sistema administrador de base de datos relacional cliente/servidor originado en Escandinavia. MySQL incluye un servidor SQL, programas para cliente para acceder al servidor, herramientas administrativas y una interface de programación para escribir programas propios.

Inicialmente MySQL se volvió popular por su rapidez y simplicidad. Pero también hubieron críticas: no tenía soporte de transacciones o **claves foráneas**[22]. De esta forma, MySQL continuó desarrollándose, no sólo añadiendo estas características de las cuales carecía, sino también otras como **replicación**[23], **subqueries**[24], **stored procedures**[25], **triggers**[26]y **vistas**[27].

Estas capacidades pusieron a MySQL en la cima de aplicaciones para empresas. Como consecuencia, las personas que consideraban sólo bases de datos de "peso pesado" para sus aplicaciones, ahora le dan una seria consideración a MySQL, que funciona en cualquier cosa, desde el más modesto hardware hasta grandes servidores de empresas.

2.2.2.4.2 Características Principales

- Velocidad: MySQL es rápido y cada vez más. Recientes mejoras, particularmente con InnoDB se han realizado y optimización de query.
- Fácil de usar: MySQL tiene un alto rendimiento, pero es un sistema relativamente simple y es mucho menos complejo de instalar y administrar que grandes sistemas.
- Soporte de lenguaje query
- Capacidad: El servidor de MySQL es multi-hebra (**multi-threaded**), lo que implica que varios clientes pueden conectarse al mismo tiempo. Cada cliente puede usar varias bases simultáneamente Se puede acceder a MySQL interactivamente usando varias interfaces que permitan poner queries y ver los resultados. Aparte, varias interfaces de programación están disponibles para muchos lenguajes como C, Perl, Java, PHP, Python y Ruby. También es accesible usando aplicaciones que soporten ODBC y .NET (protocolos desarrollados por Microsoft).

[22] Clave Foránea: Es una limitación referencial entre dos tablas. (Wikipedia, s.f.)
[23] Replicación: Es un conjunto de tecnologías destinadas a la copia y distribución de datos y objetos de base de datos desde una base de datos a otra. (Microsoft, s.f.)
[24] Subquery: Query dentro de un query.
[25] Stored Procedure: Elementos de una base de datos que corresponden a un conjunto de instrucciones SQL que pueden ser ejecutadas con una simple llamada a su nombre. (Groussard, 2013)
[26] Trigger: Es el nombre de un conjunto de comandos que se ejecutan antes o después de que se realiza un cambio en una tabla (INSERT, UPDATE y DELETE). (Ángel Arias, 2015)
[27] Vista: Es una consulta sobre unas tablas de base (SELECT) que se almacena en el Catálogo bajo un nombre que puede ser usado como una tabla más. (Enrique Rivero Cornelio, 2004)

- Conectividad y seguridad: MySQL está completamente estructurado en red y las bases de datos puede ser accedidas desde cualquier lugar en el Internet. Pero MySQL tiene control de acceso por lo que, si alguien no debe ver los datos e información de otra, no lo hará. Para proveer seguridad adicional, MySQL soporta conexiones encriptadas usando el **protocolo Secure Sockets Layer** (**SSL**).
- Portabilidad: MySQL se ejecuta en muchas variedades de Unix y Linux, así como en otros sistemas operativos como Windows. Tiene la capacidad de funcionar en diversos equipos desde pequeños dispositivos como **routers** [28] y computadoras personales hasta servidores de gama alta con muchos CPU's y grandes cantidades de memoria.

2.2.2.4.3 Historia

Las raíces de MySQL se originan en 1979, con la herramienta de base de datos UNIRG creada por Michael "Monty" Widenius para la compañía sueca TcX. En 1994, TcX comienza a buscar una RDBMS con interface SQL para usarse en el desarrollo de aplicaciones web, pero todos los servidores comerciales que se probaron eran muy lentos para las características que TcX necesitaba. Dados estos inconvenientes, Monty empezó a desarrollar un nuevo servidor.

En 1995, David Axmak de Detron HB empieza a presionar para que TcX terminará MySQL y lo publicara en Internet. David también trabajó en la documentación y en hacer que MySQL se construyera con auto herramientas de configuración de **GNU**[29].

MySQL 3.11.1 fue lanzado en el mundo en 1996 en forma de distribuciones binarias para Linux y Solaris. La compañía MySQL AB fue formada para proveer distribuciones de MySQL y ofrecer servicios comerciales. Para el 2008, Sun microsystems adquirió MySQL AB, y en el 2010, Oracle adquirió Sun. Hoy en día, MySQL está disponible tanto en forma binaria como en código fuente y trabaja en muchas más plataformas.

2.2.2.5 JavaScript Object Notation (JSON)

2.2.2.5.1 ¿Qué es?

Es un formato de texto usado para intercambiar datos entre plataformas. Otro formato de intercambio de datos que existe es XML. El mundo necesita formatos de intercambio de datos como XML y JSON, para cambiar datos entre diferentes sistemas.

JSON es un formato de intercambio de datos que muchos sistemas, no todos, han acordado usar para poder comunicar información. JSON hace el intercambio de datos entre sistemas o redes en ceros y unos. Estos ceros y unos son interpretados por un analizador sintáctico que consume los datos para que puedan ser leídos dentro del sistema al que entran. JSON es portable, lo cual implica que la transferencia de datos que realiza entre plataformas es de cierto modo compatible con ambos sistemas.

JSON, JavaScript Object Notation está basado en objetos de JavaScript donde "notación" se entiende por un sistema de caracteres para representar datos como números o palabras y "objeto" es un concepto

[28] Router: Su función principal consiste en enviar o encaminar paquetes de datos de una red a otra, es decir, interconectar subredes. (Wikipedia, s.f.)

[29] GNU: Sistema operativo de tipo Unix desarrollado por y para el Proyecto GNU y auspiciado por la Free Software Foundation. (Wikipedia, s.f.)

común de programación, en particular de POO. En programación el concepto de un objeto es similar a cómo se describe un objeto en la vida real. Como por ejemplo un par de zapatos; que tiene atributos o propiedades como color, estilo, marca y tipo de plantilla. Algunos de los valores de estos atributos pueden ser numéricos como el tamaño y otros pueden ser booleanos (verdadero/falso) como "tiene o no cordones".

El uso de JSON en el ejemplo de "como son mis zapatos este momento" se muestra en la *Figura 2.2-6*.

```
{
        "marca": "Crocs",
        "color": "rosado",
        "tamaño": 9,
        "tieneCordones": false
}
```

Figura 2.2-6 Ejemplo JSON (Bassett, 2015)

2.2.2.5.2 Características Principales

- JSON es un formato de intercambio de datos.
- Es un lenguaje de programación independiente (no es necesario el uso de JavaScript).
- Está basado en notación literal de objetos de JavaScript
- Representa datos de modo que sean amigables universalmente para conceptos de programación.
- Usa tipos de datos como números, strings, booleanos, null, colección o arreglos de valores y objetos.
- Fácil de escribir y leer para las personas.
- Fácil de analizar sintácticamente y generar para las máquinas
- Usa convenciones que son familiares para programadores de C, C++, C#, Java, JavaScript, Perl, Python y muchos otros. (JSON, s.f.)
- Construido en dos estructuras
    - Una colección de pares nombre/valor. En algunos lenguajes esto se entiende como un objeto, registro, diccionario, **tabla hash**[30], lista con claves o arreglo asociativo. (JSON, s.f.)
    - Una lista ordenada de valores. En muchos lenguajes esto se entiende como un arreglo, vector, lista o secuencia. (JSON, s.f.)

2.2.2.5.3 Estructura JSON

Objeto: Un objeto (*Figura 2.2-7*) es un set desordenado de pares nombre/valor. El objeto comienza con { (llave abierta) y termina con } (llave cerrada). Cada nombre está seguido por: (doble punto) y el par nombre/valor se separa por , (coma). (JSON, s.f.)

[30] Tabla Hash: Su objetivo es crear un vector en donde almacenar los elementos. (Carrillo, 2006)

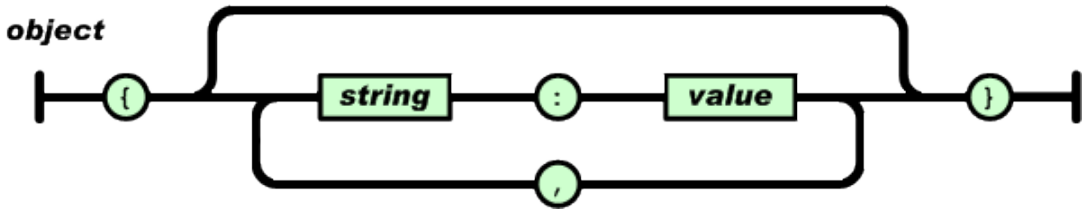

Figura 1.1-7 Estructura de un objeto JSON (JSON, s.f.)

Arreglo: Un arreglo (*Figura 2.2-8*) es una colección de valores. Un arreglo comienza con [(corchete izquierdo) y termina con] (corchete derecho). Los valores se separan por , (coma). (JSON, s.f.)

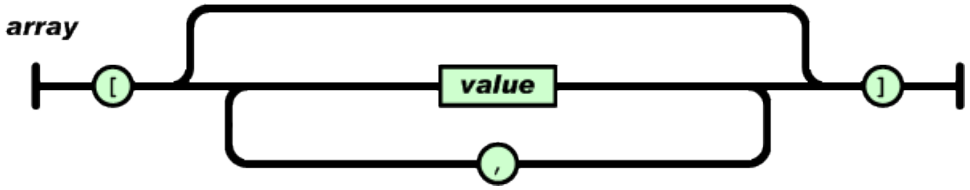

Figura 2.2-8 Estructura de un arreglo (JSON, s.f.)

Valor: Un valor (*Figura 2.2-9*) puede ser una cadena de caracteres con comillas dobles, o un número, o true o false o null, o un objeto o un arreglo. Estas estructuras pueden anidarse. (JSON, s.f.)

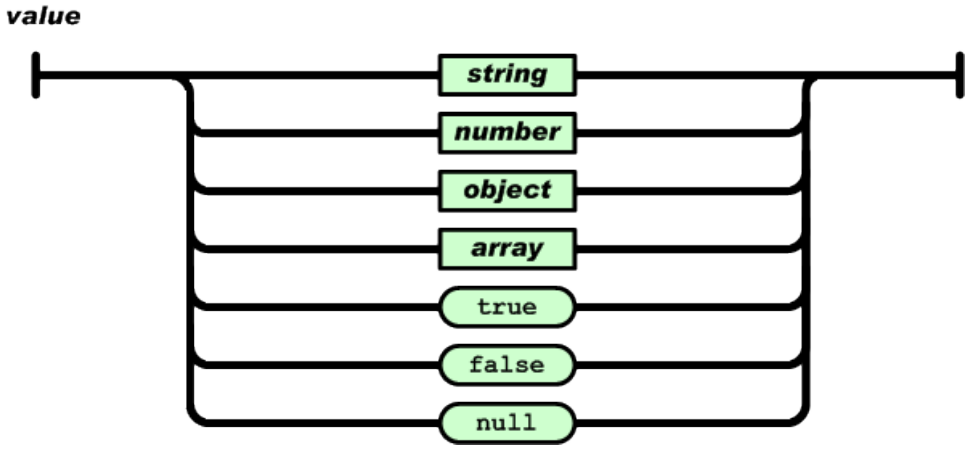

Figura 2.2-9 Estructura de un valor (JSON, s.f.)

String: Una cadena de caracteres (*Figura 2.2-10*) es una colección de cero o más caracteres Unicode, encerrados entre comillas dobles, usando barras divisorias invertidas como escape. Un carácter está representado por una cadena de caracteres de un único carácter. Una cadena de caracteres es parecida a una cadena de caracteres C o Java. (JSON, s.f.)

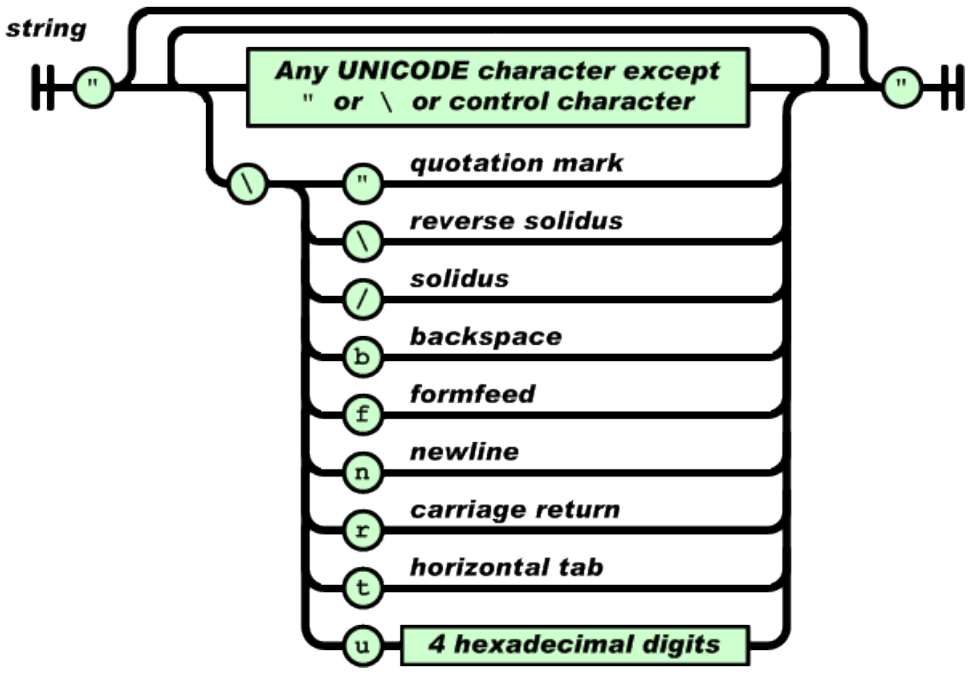

Figura 2.2-10 Estructura de una cadena de caracteres (JSON, s.f.)

Número: Un número (*Figura 2.2-11*) es similar a un número C o Java, excepto que no se usan los formatos octales y hexadecimales. (JSON, s.f.)

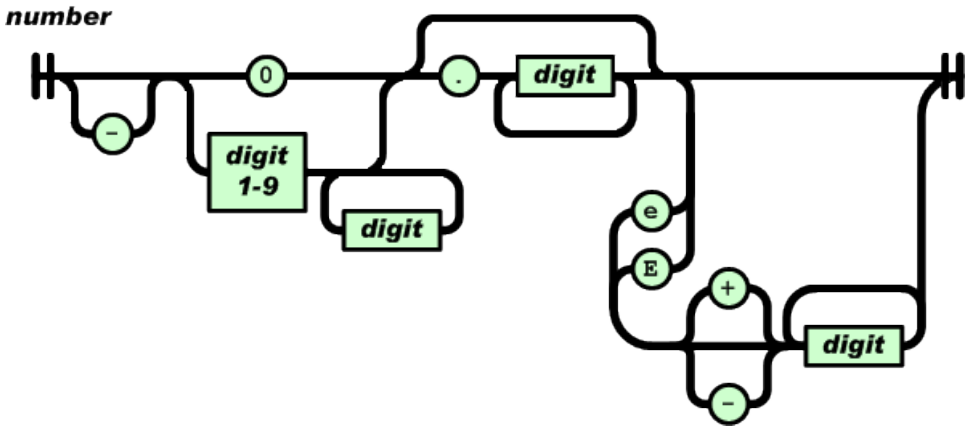

Figura 2.2-11 Estructura de un número (JSON, s.f.)

2.2.2.6 Ajax

2.2.2.6.1 ¿Qué es?

Ajax es una herramienta moderna usada para crear aplicaciones web amigables para el usuario, es como los materiales que se necesitan para hacer un edificio alto. Alturas grandes se logran con hierro mas no con madera, y sus extensiones son modernas y vistosas con metales y vidrio. La estructura básica continúa allí, las paredes se juntan tanto paralela como perpendicularmente unas con otras con ángulos de 90°, y todos los elementos básicos de la estructura, incluyendo la plomería, electricidad y luces, son los mismos; sólo están mejorados.

Eso es Ajax y su estructura, una aplicación Ajax se construye como estructura subyacente de XHTML[31] (Extensible HyperText Markup Language), que es una extensión de HTML. De tal modo que para construir aplicaciones web con Ajax es necesario considerar las siguientes herramientas:

- XHTML
- Documento Object Model (DOM)
- JavaScript
- Cascading Style Sheets (CSS)
- Extensible Markup Language (XML)

2.2.2.6.2 Características Principales

- Presentación basada en estándares usando XHTML y CSS
- Visualización dinámica e interacción usando DOM
- Intercambio y manipulación de datos usando XML y XSLT
- Recuperación de datos asíncronos usando XMLHttpRequest
- JavaScript poniendo todo junto

2.2.2.6.3 Modelo

Tradicionalmente, la mayoría de acciones del usuario en la interface desencadenaban una solicitud HTTP de vuelta a un servidor web. El servidor lo procesaba, recuperaba información, hacía cuentas, pasaba por varios sistemas heredados y finalmente retornaba la página HTML al cliente (*Figura 2.2-12 Izquierda*).

En cambio, Ajax elimina la naturaleza inicia-para-inicia-para de la interacción tradicional en la web mediante la introducción de un intermediario entre el usuario y el servidor; un motor Ajax. En donde en vez de cargar una página web al inicio de la sesión, el navegador carga el motor Ajax independientemente de la comunicación con el servidor, de manera que el usuario no tiene que esperar en una página tanto tiempo hasta que el servidor haga algo (*Figura 2.2-12 Derecha*). El motor mencionado está escrito en JavaScript y usualmente plegado lejos en un marco oculto. Este motor es el responsable tanto de renderizar la interface que el usuario ve y de comunicarse con el servidor en representación del usuario. Permite la interacción del usuario con la aplicación suceda de manera asíncrona.

[31] XHTML: Es un lenguaje de etiquetas destinado a la creación de páginas web. (Lancker, 2009)

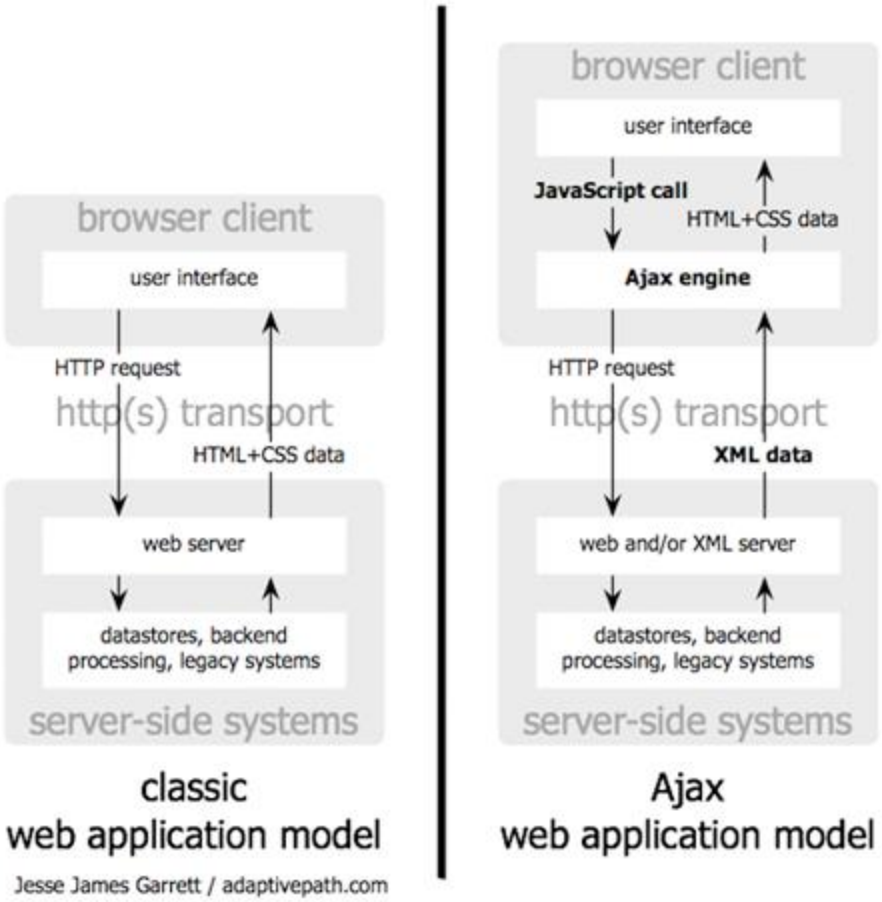

Figura 2.2-12 Comparación del modelo tradicional para aplicaciones web con el modelo Ajax (Garret, 2005)

2.2.2.6.4 Historia

El término de Ajax fue declarado como tal por Jesse James Garret el 18 de febrero del 2005 en uno de sus artículos: ""Ajax: A New Approach to Web Applicatios" o "Ajax: Un Nuevo Acercamiento a Aplicaciones Web" en español. Y se basó en técnicas usadas por Google.

Parecería que Ajax salió de la nada, pero en realidad ha estado presente por algún tiempo. Algunos años de trabajo alrededor de la Web se dieron crear herramientas y patrones que juntos formaron Ajax. (Larsen, 2011)

Durante el desarrollo de OWA (Outlook Web Access) por Alex Hopmann y su equipo en 1998, se evaluaron dos opciones: un cliente formado sólo por páginas HTML estáticas que se recargaban constantemente y un cliente realizado completamente con HTML dinámico o DHTML. Alex Hopmann pudo ver las dos opciones y se decantó por la basada en DHTML. Sin embargo, para ser realmente útil a esta última le faltaba un componente esencial: "algo" que evitara tener que enviar continuamente los formularios con datos al servidor. (Eguiluz, 2008)

Así Alex creó XMLHTTP, el cual fue usado por Microsoft para actualizar dinámicamente nuevas historias y frases en la página inicial de Internet Explorer. Google hizo una amplia implementación de estándares que se cumplan fusionando el buscador Ajax con Gmail (2004) y Google Maps (2005). (Wikipedia, s.f.)

2.2.2.7 EditableGrid

2.2.2.7.1 ¿Qué es?

EditableGrid es una librería JavaScript open source desarrollada por Webismymind, una compañía en Bélgica. Está orientada en transformar tablas en componentes totalmente editables. Su API está enfocada en la simplicidad: sólo unas pocas líneas de código son necesarias para tener una tabla editable lista y funcionando.

Originalmente, EditableGrid era un componente desarrollado para Hiflow Suite. Luego, se decidió factorizarla como una librería externa. Está publicada bajo **Licencia MIT**[32].

2.2.2.7.2 Características principales

- Múltiples formatos: Se pueden cargar datos desde XML, JSON o adjuntos de una tabla HTML existente.
- Ordenamiento de columnas: Se pueden ordenar las columnas de acuerdo al tipo sólo haciendo clic en la cabecera de la columna.
- Filtro: filtro de la tabla.
- Paginación: Paginación completa ya sea para el cliente o el servidor.
- Columnas tipificadas: Cada columna puede tener su propio tipo de dato (cadena de caracteres, enteros, doble, booleano, url, email, fecha).
- Editor de celda: Cada celda tiene un editor acorde al tipo de columna.
- Enumeración de valores: Cada celda puede tener la opción de escoger entre varios valores de un **drop down list**[33].
- Validación de datos por celda: Valida los datos de cada celda para ingresar datos correctos.
- Almacenamiento local.
- Posibilidad de introducir renderers de celda propios, editores y validadores.
- Vinculación con PHP: Construcción de una tabla propia en PHP para que el archivo XML se genere.
- Inserción y eliminación de filas.
- Soporte para grupos de opciones en columnas con valores enumerados.
- Integración de Open Flash Chart: Opción de renderizar gráfico de barras o pastel basado en los datos de la tabla.
- Stand-alone: No se necesitan dependencias.
- Soporte automático para interface de usuario jQuery.
- Soporte para autocompletar entradas jQuery (mediante una extensión).

Marcha atrás para todos los eventos: edición, inserción, ordenamiento, etc.

[32] Licencia MIT: Licencia pública del Instituto Tecnológico de Massachusets (MIT) que permite, pero no requiere, que el código fuente este abierto. (Valimaki, 2005)

[33] Drop Down List: Es un elemento de control gráfico, que permite al usuario elegir un valor determinado dentro de una lista. (Wikipedia, s.f.)

2.2.3 Criterios de selección de herramientas

2.2.3.1 Criterio Legal

Todo proyecto de software debe tener bases legales para poder construirlo y sobre todo para que sus usuarios y/o clientes puedan ponerlo en uso. Para que un sistema esté dentro de los parámetros legales tanto de desarrollo cómo de producción, debe requerir de licencias. Dado que el producto que se entregará será una donación, que las personas y monjitas que administran el plantel no cuentan con los recursos económicos suficientes para financiar costosos servidores, bases de datos y programas, y al existir varios tipos de licencias en el mercado, se han escogido licencias de software libres (Notepad++, MySQL, Linux).

2.2.3.2 Criterio Técnico

El enfoque técnico de este proyecto fue desarrollar un sistema en una plataforma web. Uno de los lenguajes más comunes y con mayor soporte técnico es PHP, el cual fue diseñado para trabajar en conjunto con el motor de base de datos MySQL. El servidor Linux también tiene una gran aceptación y soporte (en línea principalmente), sin embargo, su uso y manejo es mucho más complejo en relación a un servidor de Windows, ya que no dispone de una interface gráfica y toda operación es realizada a través de comandos.

2.2.3.3 Criterio Financiero

Debido a que el sistema es una donación, las herramientas que el sistema usa en producción tienen una licencia de software libre, es decir no tienen costo alguno.

2.2.3.4 Criterio Operativo

Las herramientas escogidas en su mayoría se han usado a lo largo de la carrera de Ingeniería en Sistemas, por lo cual trabajar con ellas no requirió de ninguna clase de capacitación extra. El uso de una planificación RUP se prefirió dado que no deja cabos sueltos a lo largo de la elaboración y producción del proyecto.

Por otro lado, las herramientas que se escogieron para diseñar la mayoría de interfaces con las que el usuario interactuará, fueron seleccionadas en función de la amigabilidad y facilidad de uso para quienes usen el sistema.

2.3 CICLO DE VIDA

El ciclo de vida es cómo se va a desarrollar el proyecto a lo largo del tiempo y en qué consiste cada fase del mismo. Al tener un ciclo de vida RUP, las iteraciones e hitos principales se proporcionan como se muestra en la *Figura 2.3-1*.

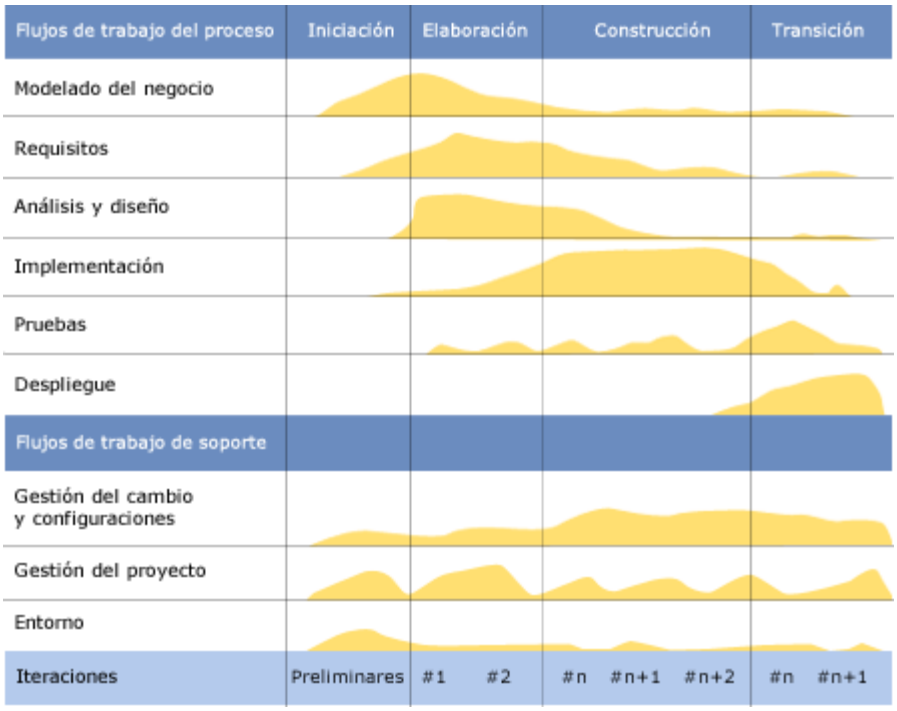

Figura 2.3-1 Ciclo de vida RUP (Wikipedia, 2016)

2.4 PATRONES DE DISEÑO

2.4.1.1 ¿Qué son?

Algunos programas reutilizan las mismas estructuras básicas o set de ideas, estas estructuras son regularmente usadas y llamadas Patrones de Diseño[34]. Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, y luego describe el set de soluciones al mismo, de manera que esta solución pueda ser utilizada un millón de veces, sin que necesariamente se tenga que hacer de la misma manera para cada vez que se usa.

Un patrón consta principalmente de cuatro elementos esenciales:

- **Nombre del patrón**: Describe el diseño, solución y consecuencias del patrón en una o dos palabras, permitiendo diseñar a un alto nivel de abstracción.
- **Problema**: Describe cuando se aplica un patrón, explicando el problema y su contexto. También, problemas de diseño específicos sobre cómo representar algoritmos como objetos o clases que son significativas en un diseño no-flexible.
- **Solución**: Describe los elementos que forman parte del diseño, sus relaciones, responsabilidades y colaboradores. No describe un diseño o implementación particular en concreto; sino que provee una descripción abstracta de un diseño de problema y cómo un grupo general de elementos puede resolverlo.

[34] Patrón de Diseño: "Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software." (Tedeschi, 2014)

- **Consecuencias**: Son los resultados y compensaciones de aplicar el patrón. Son críticas para evaluar alternativas de diseño y para comprender los costos y beneficios del uso del patrón.

2.4.1.1.1 Principales patrones de Diseño

- **Adapter Pattern**: Patrón de diseño que adapta una interfaz de un objeto a lo que otro objeto espera.
    - Problema: En una aplicación, se tienen objetos y clases funcionales y estables, pero al momento en que hay nuevos requerimientos, estos objetos y clases deben ser usados de una manera distinta de la que fueron originalmente diseñados
    - Solución: Construir otro objeto que funcione como intermediario entre el objeto original y la nueva funcionalidad, denominado “objeto adaptador”.
- **Builder Pattern**: Patrón de diseño que define el diseño de un objeto que maneja la compleja construcción de otro objeto.
    - Problema: Cuando un objeto es instanciado, técnicamente es un objeto completo, sin embargo, otros objetos pueden requerir la ejecución de métodos públicos adicionales para poder considerarse “completos” y disponibles para el resto de la aplicación.
    - Solución: Construir un nuevo objeto que sea responsable de interpretar todos los resultados de la “lógica del negocio” y llamar sólo a la funcionalidad requerida para construir un objeto completo.
- **Data Access Object Pattern**: Patrón de diseño que describe la creación de un objeto que provee acceso transparente a cualquier fuente de datos.
    - Problema: Repetición y abstracción de fuentes de datos.
    - Solución: Se crea un objeto que encapsula una manera inteligente de crear llamadas SQL, reduciendo la complejidad y repetición en los procesos de creación y actualización de datos.
- **Decorator Pattern**: Patrón de diseño que se encarga de decorar partes del contenido de un objeto existente sin modificar la estructura original.
    - Problema: Cuando un objeto requiere demasiados “hijos” para ser funcional, el código sacrifica la comprensión y mantenimiento que tiene el programador.
    - Solución: Aplicar cambios incrementales en un objeto base sin tener que sobrescribir en las funcionalidades existentes, de manera que se pueden insertar “decoradores” directamente en la corriente de ejecución del código, modificar o “decorar” el objeto y no afectar ninguna otra corriente de código.
- **Delegate Pattern**: Patrón de diseño que remueve decisiones y funcionalidades complejas del objeto núcleo, distribuyéndolas o delegándolas a otros objetos.
    - Problema: Programación de tipo procedural que se basa en control de flujos de declaraciones condicionales.
    - Solución: Se designa un objeto que se base en ejecutar funciones específicas mediante la evaluación de una declaración condicional, el objeto puede delegar la decisión a objetos “delegados”.
- **Interpreter Pattern**: Patrón de diseño que analiza una entidad en busca de elementos clave y provee su propia interpretación o acción correspondiente a cada clave.

    - Problema: Ciertos problemas ocurren repetidamente en un dominio correctamente establecido y definido, si el dominio fuese caracterizado por un "lenguaje", entonces los problemas serían más fácilmente resueltos con un "motor" de interpretación.
    - Solución: Modelar el dominio con gramática recursiva, cada regla en la gramática puede ser una regla que referencia otras reglas o un nodo en un árbol.
- **Iterator Pattern**: Patrón de diseño que ayuda a construir objetos que puedan proveer una interfaz estándar simple para iterar cualquier tipo de información contable.
    - Problema: Escribir múltiples queries[35] cada vez que se necesite obtener información (aunque se repitan).
    - Solución: Se crea un objeto iterativo que maneje los result set[36] el cual provee un query a la clase constructora y luego itera los result sets mediante la llamada a los métodos públicos de la clase.
- **Mediator Pattern:** Patrón de diseño que es usado para desarrollar un objeto que se comunique o que sea mediador de cambios para una colección de objetos similares, sin que estos interactúen directamente con los otros.
    - Problema: Un objeto que depende de otro, o que otros dependen de él, es actualizado o cambia su interface; de tal forma que los otros objetos dependientes también necesitan actualizarse o cambiarse.
    - Solución: Se crea un objeto mediador que tome el cambio de otro objeto y aplique los cambios a todos los demás objetos dependientes.
- **Observer Pattern**: Patrón de diseño que facilita la creación de objetos que observan el estado de un objeto específico y provee una funcionalidad de estado específico que se desacopla del objeto principal.
    - Problema: Las aplicaciones de Software necesitan constantemente nuevas características para mantenerse viables. Sin embargo, ¿cómo es posible añadir nuevas características sin tener que refactorar completamente algunos de los objetos principales
    - Solución: Se crea un objeto principal basado en el Patrón de diseño de Observador, el cual se torna responsable de comunicar todos los cambios de estado a otras clases que estén asignadas a observar y entender dichos cambios. De tal manera que las clases observadoras toman acciones externas o modifican el objeto principal.
- **Prototype Pattern:** Patrón de diseño que crea objetos de tal manera que un objeto inicial o prototipo puede ser copiado y clonado más eficazmente que creando una nueva instancia.
    - Problema: Costo de crear nuevos objetos complejos.
    - Solución: Crea una interface que preserva las propiedades principales de un objeto anterior mediante la duplicación.
- **Proxy Pattern:** Patrón de diseño que construye un objeto que es posicionado transparentemente junto a otros dos objetos, con el fin de interceptar la comunicación o acceso.
    - Problema: Otorgar acceso a un recurso específico y; verificación y validación de credenciales, identidades y ubicaciones.

[35] Query String: cadena de consulta, este término generalmente se utiliza para hacer referencia a una interacción con una base de datos. (Wikipedia, 2015)

[36] Result Set: set de filas de una base de datos, o metadata del query (nombres, tipos y tamaños de cada campo o columna)

    - Solución: Se crea un objeto proxy que intercepte comunicaciones entre dos o más objetos.
- **Singleton Pattern:** Patrón de diseño que se usa para restringir el número de veces que un objeto específico puede ser creado cada vez, proveyendo acceso a una instancia compartida propia.
    - Problema: Creación de varias instancias al mismo objeto.
    - Solución: Permitir una sola instancia de un objeto.
- **Strategy Pattern:** Patrón de diseño que ayuda a construir un objeto con una arquitectura que pueda hacer uso de un algoritmo en otro objeto en demanda en lugar de contener la lógica para sí mismo.
    - Problema: Existencia de objetos con exceso de código.
    - Solución: Invocar algoritmos de otras clases.
- **Template Pattern:** Patrón de diseño que crea un objeto abstracto que haga cumplir una serie de métodos y funcionalidades que serán usados en común por varias clases hijas, como una plantilla para su propio diseño.
    - Problema: Sobreescritura de métodos.
    - Solución: Crea una clase cuya intención es ser usada como un padre de otra clase, para que las clases hijo cumplan con métodos y funcionalidades específicos.
- **Visitor Pattern:** Patrón de diseño que construye distintos objetos que contienen un algoritmo, el cual, al ser consumido por un objeto padre de forma estándar, aplica el algoritmo a dicho objeto padre.
    - Problema: Sobre-escritura de código.
    - Solución: Se crea un objeto A que al ser instanciado por cualquier otro objeto B, cumple la funcionalidad en B como este lo requiera.
- **MVC[37]:** Hay tanto autores que lo consideran un patrón de diseño, como autores que lo consideran un tipo de arquitectura para la creación organizada de una aplicación; que consta de tres objetos: Modelo, Controlador y Vista.

  El controlador será el responsable de determinar qué tipo de acción la aplicación está tratando de lograr desde la petición web. Será el responsable de que la petición web tenga sentido y de ejecutar la acción apropiada del modelo y retornar una salida de esa acción.

  El objeto Modelo es el bloque creado para cada sección del sitio web. Contiene al menos un método público que será ejecutado por el controlador.

  El objeto Vista, como su nombre lo dice, se encarga de la parte visual o interfaz gráfica de usuario del sitio web.

2.5 ARQUITECTURA DE LA APLICACIÓN

La arquitectura de una aplicación es la descripción formal de un sistema, o un plan detallado del mismo a nivel de componentes, para guiar su implementación. (Open Group Standard, 2011)

[37] MVC: Modelo-Vista-Controlador. Arquitectura

2.5.1.1 Modelo-Vista-Controlador (MVC)

2.5.1.1.1 ¿Qué es?

Es un patrón de diseño o también se puede considerar una arquitectura que está construida en base a la interconexión de tres componentes principales, generalmente con un enfoque estricto sobre los paradigmas de la **Programación Orientada a Objetos** (**POO**[38]). Los tres componentes en general son modelos, vistas y controladores. La interacción de los componentes se muestra en la *Figura 2.5-1*.

- **Modelo:** El modelo es donde se encuentra toda la lógica de la aplicación. La lógica del negocio puede ser cualquier cosa específica en cuanto a cómo una aplicación almacena información o usa servicios de terceros, con el fin de que satisfaga todos los requerimientos del negocio. Si una aplicación necesita acceder a la información de una base de datos, el código para hacerlo estará en el modelo.
- **Vista:** La vista es donde se encuentran todos los elementos de la interfaz de usuario. Incluye marcado **HTML**[39], **estilos CSS (Cascading Style Sheet)**[40] y **archivos JavaScrip**t[41]. Todo lo que un usuario pueda ver o con lo que pueda interactuar se encuentran en la vista.
- **Controlador:** El controlador es el componente que conecta las vistas y modelos. El controlador separa la lógica del negocio de la interfaz de usuario, y maneja cómo va a responder la aplicación a la interacción del usuario con la vista. Son el primer punto de entrada, porque la petición para al controlador que luego la instancia en los modelos y las vistas.

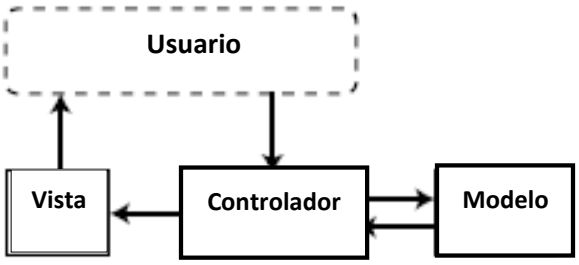

Figura 2.5-1 (Pitt, 2012)

2.5.1.1.2 Características Principales

- Su enfoque principal es implementar interfaces de usuario.
- No todas las peticiones de la aplicación necesitan un modelo o una vista.
- Los elementos que se carguen dependen del tipo de petición y recursos que se requieran.
- Un controlador puede servir sólo para alternar el estado de una aplicación, o para retornar información no analizada directamente desde un servicio de terceros.

[38] Programación Orientada a Objetos: El diseño orientado a objetos consiste en averiguar cuáles son los objetos de un sistema, las clases en las que se pueden agrupar y las relaciones entre objetos. (Durán, 2007)
[39] HTML: Es un estándar que, en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, etc. (Quesada., s.f.)
[40] CSS: "Cascade Style Sheet", permite otorgar atributos a los elementos de los documentos HTML y una separación del diseño de los contenidos de las páginas web. (Enrique E. Condor Tinoco, 2014)
[41] JavaScript: Es un lenguaje interpretado usado para múltiples propósitos. (Gauchat, 2012)

2.5.1.1.3 Beneficios y desventajas

Hay muchos aspectos que tomar en cuenta al realizar una aplicación Web. Está el diseño, que afecta directamente a los intereses del usuario en el producto. También está la lógica del negocio, necesaria para hacer cosas prácticas cómo procesos de venta de ítems y facturación. Y está el proceso de mejora continua; actualizando, solucionando errores y puliendo la aplicación en general.

Al aplicar MVC nos podemos encontrar con los siguientes inconvenientes y beneficios:

- Ventajas
    - Escalabilidad.
    - Inclusión de librerías.
    - Abstracción de datos.
    - Código entendible.
    - Uso de frameworks.
    - Control de recursos.
- Desventajas
    - Soporte técnico escaso en relación a otras arquitecturas.
    - No es muy recomendable para aplicaciones pequeñas.
    - Mayor exigencia de programación orientada a objetos.
    - No es fácil de manejar en todos los lenguajes de programación.

2.5.1.1.4 Historia

Se dice que Ada Lovelace escribió el primer programa de computación para la **Máquina Analítica** a mediados del siglo XIX, el primer prototipo de computadora mecánica creado por Charles Babbage; de tal manera que el desarrollo de software no es una idea nueva. Ha pasado un tiempo considerable desde entonces y el desarrollo de Software ha crecido y es ahora algo que se puede considerar una de las mayores contribuciones al desarrollo de nuestra especie.

MVC fue concebido por primera vez en 1978 como un diseño de solución para un problema en particular, inventado por Trygve Reenskaug durante su cargo como científico de visita en el grupo de Smalltalk en el famoso Centro de Investigación Xerox Palo Alto. El objetivo principal era resolver el problema de representar (modelar) sistemas complejos del mundo real como por ejemplo “el diseño y construcción de un puente principal, una estación de poder o una plataforma de producción de petróleo”. Una persona tiene un modelo mental de estos sistemas y una computadora tiene un modelo digital. ¿Cómo crear un puente que elimine la “distancia” entre estos dos modelos? Originalmente los Modelos, Vistas y Controladores fueron definidos de la siguiente manera:

> Modelos – Representación del conocimiento. Puede ser un objeto simple, o puede alguna estructura de objetos.
>
> Vistas – Representación visual del modelo. Destacaría atributos específicos del modelo y suprimiría otros. Por tanto, actúa como un filtro de presentación.
>
> Controladores – Enlace entre el usuario y el sistema. Recibe el output del usuario y lo traduce en los mensajes apropiados para que pasen a una o más vistas.

Es importante saber que el patrón MVC fue creado mucho antes que el primer navegador Web. Fue implementado como parte de la librería de Smalltalk-8º. Y fue originalmente usado como un patrón arquitectónico para crear interfaces de usuario gráficas (**GUIs**).

El concepto original (*Figura 2.5-2*) para el patrón MVC fue muy distinto al concepto asociado con el patrón actual. En el contexto de una Interface Gráfica de Usuario, el MVC era interpretado de la siguiente manera:

Modelo – Una pieza particular de datos representada por una aplicación. Por ejemplo, lectura de temperatura de una estación de clima.

Vista – Una representación de datos del modelo. El mismo modelo podría tener múltiples visas asociadas con este. Por ejemplo, una lectura de temperatura puede ser representada por una etiqueta por un gráfico de barras. Las vistas están asociadas con un modelo particular mediante la relación Observador.

Controlador – Recoge las entradas del usuario y modifica el modelo. Por ejemplo, el controlador toma el número de clics y tecleos y actualiza el modelo.

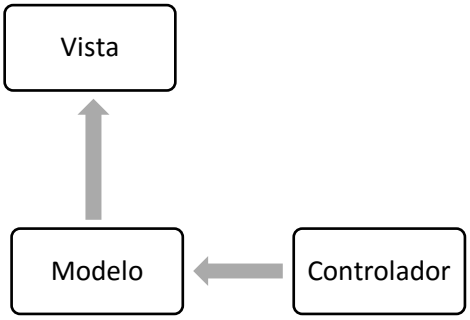

Figura 2.5-2 (Walther, 2008)

El concepto de MVC cambió drásticamente (*Figura 2.5-3*) con la introducción de **JavaServer Pages (JSP[42])**, en donde MVC venía dado por los siguientes términos:

Modelo – Lógica del negocio y uno o más fuente de datos como una base de datos relacional.

Vista – La interface de usuario que despliega información acerca del modelo.

Controlador – El mecanismo de flujo de control que define por dónde el usuario interactúa con la aplicación.

[42] JSP: "Java Server Pages" permite crear páginas web dinámicas basadas en HTML, XML, etc. (Cardador Cabello, 2014)

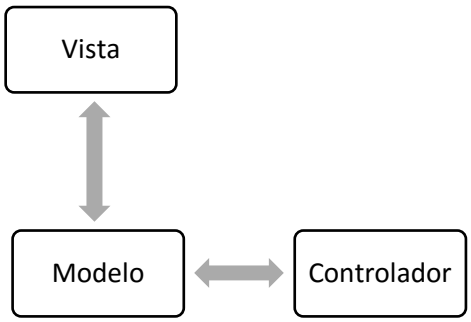

Figura 2.5-3 (Walther, 2008)

Capítulo 3. CASO DE ESTUDIO

La Unidad Educativa Ángel de La Guarda actualmente cuenta con tan sólo manejo de notas mediante hojas de cálculo en Excel. Todos los archivos de control de registros son accesibles para todos los docentes y administrativos en una sola computadora.

Físicamente el centro cuenta con infraestructura suficientemente óptima para conectar todas las computadoras necesarias en red para que interactúen con un sistema individual de manejo académico.

En este capítulo se incluirá detalladamente las fases de Inicio, Elaboración, Construcción y Transición del proceso, incluyendo los flujos de trabajo del proceso (elementos esenciales para el desarrollo y comprensión del proyecto) y los flujos de trabajo de soporte (Administración de cambio y configuración, logística y entorno) para cada uno.

Cabe recalcar que en algunas secciones se encuentran únicamente fragmentos de la documentación, cómo descripción general para sus similares. El material completo se puede encontrar en la sección de Anexos, dentro del CD del proyecto.

3.1 INICIO

3.1.1 Flujos de Trabajo del Proceso

3.1.1.1 Requerimientos

3.1.1.1.1 Requerimientos Funcionales

1. Ingreso al sistema por verificación.
2. Administrar años lectivos.
3. Administrar registro de personal (usuarios).
4. Administrar asignación de materias a docentes.
5. Administrar registro de estudiantes.
6. Administrar registro de calificaciones.

3.1.1.1.2 Requerimientos No Funcionales

1. Debe ser de fácil navegabilidad, interacción y uso.
2. Posee ingreso a información por usuarios.
3. Debe ser seguro y mantener la integridad de la información almacenada.
4. Debe acceder de forma correcta y rápida a la información.
5. Debe soportar plataformas web.
6. Debe ser programado en PHP.
7. Debe contener una base de datos con tablas para Año Lectivo, Notas, Materias, Estudiantes, Cursos, Asignación de Cursos-Materias-Profesores, Personal y Tipo de Personal, en MySQL.
8. Las Notas de cada Materia para cada Año Lectivo se dividen en:
    a. 2 Quimestres: Cada quimestre tiene su nota final, la cual es resultado del promedio entre 3 notas parciales y un examen quimestral.
    b. El promedio de la nota final de los 2 quimestres, es el resultado de la nota final.

c. Si la nota final es inferior a 7, hay la posibilidad de tomar 3 exámenes supletorios para aprobar la materia (promediados con la nota final deben resultar mayores a 7).

9. Las materias de Computación, Arte y Música se promedian en una sóla llamada Educación Estética.
10. Las materias de Proyectos y Proyectos-inglés se califican cualitativamente, de la misma manera que las conductas.
11. Cada aporte tiene una nota de conducta/comportamiento calificada del 1 al 10 y varios aspectos de conducta y aspecto personal calificados con letras. (S – Siempre, CS – Casi siempre, R – Regularmente, CN – Casi Nunca, N – Nunca), que le corresponden sólo al tutor.

3.1.1.2 Modelado del Negocio

3.1.1.2.1 Subsistemas de la Institución

La *Figura 3.1-1* muestra cómo se divide la institución con respecto a los sistemas que del proyecto.

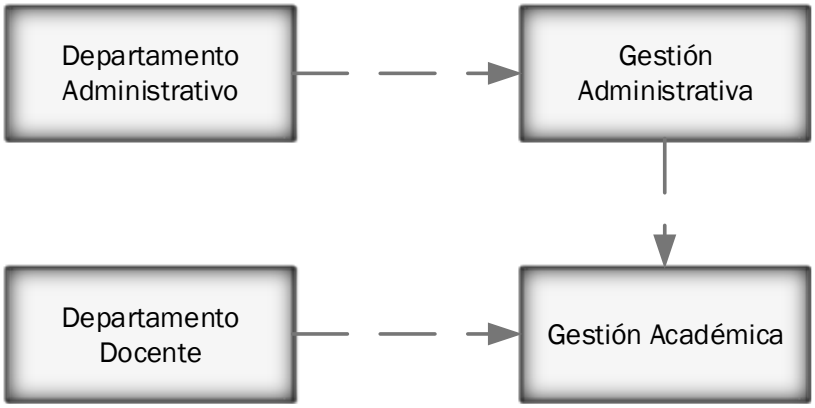

Figura 3.1-1 (Wendy Jaramillo, 2016)

3.1.1.2.2 Modelo de Casos de Uso del Negocio

El Sistema de Gestión Académica interactúa directamente con distintos actores, entre ellos:

- Administrador: encargado de la administración completa del sistema y con permisos para gestión académica como se muestra en la *Figura 3.1-2*.
- Profesor: encargado de administración de calificaciones con permisos únicamente de gestión académica cómo se muestra en la figura.

Los actores indirectos son los alumnos y sus representantes legales.

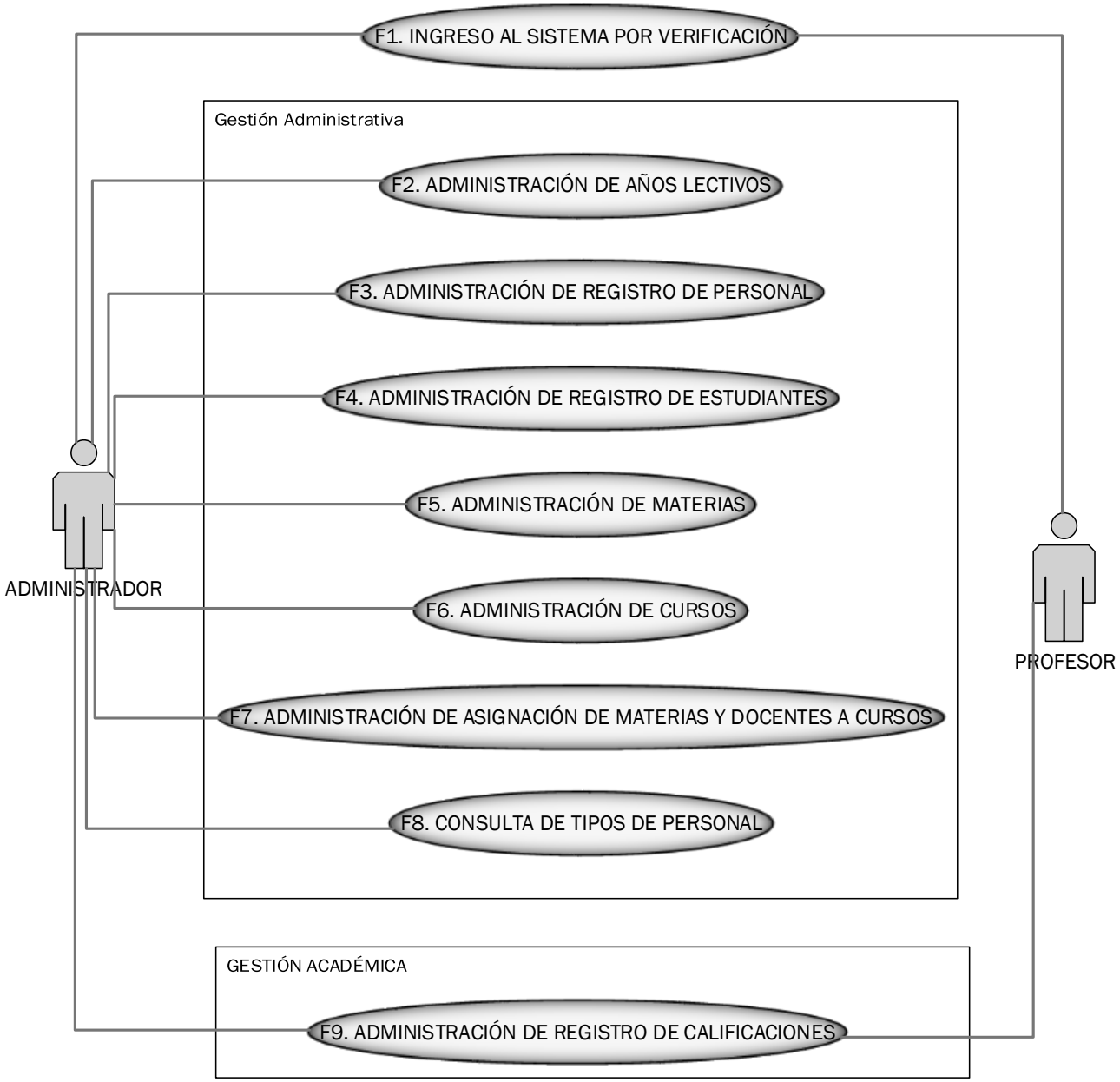

Figura 3.1-2 (Wendy Jaramillo, 2016)

3.1.1.2.2.1 Subprocesos y Diagramas

3.1.1.2.2.1.1 Subproceso F1, Diagrama de Secuencia y Diagrama de Colaboración

La *Figura 3.1-3* detalla los subprocesos.

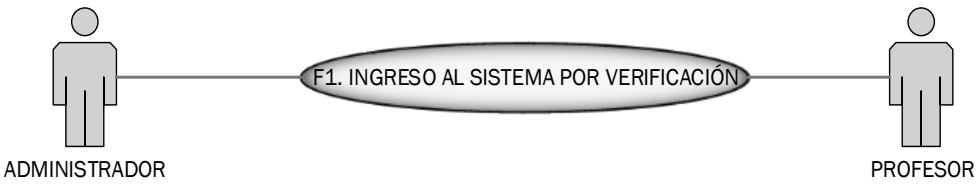

Figura 3.1-3 Subproceso F1 Ingreso al Sistema por Verificación. (Wendy Jaramillo, 2016)

1. Actores: Administrador, Profesor
2. Flujo Principal
    a. Sistema muestra página de inicio para llenar campos de usuario y contraseña en caso de que no haya una sesión iniciada. (E3)
    b. Actor llena campos de usuario y contraseña.
    c. Actor presiona botón ingresar.
    d. Sistema verifica que los campos obligatorios hayan sido llenados. (E1)
    e. Sistema verifica autenticación. (E2) (E3)
    f. Sistema redirecciona a pantalla de inicio con todas las páginas disponibles.
3. Flujo Secundario: ninguno
4. Excepciones
    a. E1
        i. Los campos obligatorios no fueron llenados.
        ii. Los campos fueron llenados con tipos de dato incorrectos
    b. E2
        i. Sistema no encontró datos en la base de datos.
        ii. Error de conexión con la base de datos.
    c. E3
        i. Error de conexión con el servidor

3.1.1.2.2.1.2 Subproceso F2

La Figura 3.1-4 detalla los subprocesos.

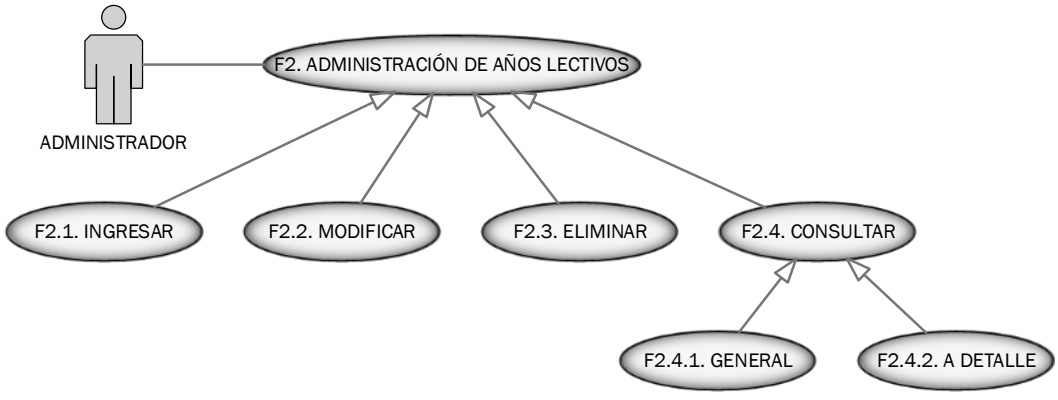

Figura 3.1-4 Subproceso F2: Administración de Año Lectivo

1. F2.1. Ingresar
    a. Actores: Administrador
    b. Flujo Principal

i. El actor presiona el botón “Añadir Nuevo Año Lectivo”.
ii. El sistema abre una ventana con campos de fechas de inicio y fin de año lectivo, número de días laborables y vacaciones, y fechas para registrar calificaciones. (E1) (E2)
iii. El actor llena los campos.
iv. El actor presiona el botón “Apply”. (FS1)
v. El sistema verifica los campos (E3).
vi. El sistema cierra la ventana de ingreso, añade el año lectivo en la base de datos y actualiza la página de Administración de Años Lectivos. (E2)

c. Flujo Secundario
   i. FS1
      1. El actor presiona el botón “Cancelar”
      2. El sistema cierra la ventana de ingreso sin añadir datos a la base de datos.

d. Excepciones
   i. E1
      1. El sistema termina una sesión por inactividad en cierto período de tiempo.
      2. Error de conexión con el servidor
   ii. E2
      1. Error de conexión con la base de datos.
   iii. E3
      1. Los campos obligatorios tienen valor nulo.
      2. Los campos fueron llenados con tipos de dato incorrectos

2. F2.2. Modificar
   a. Actores: Administrador
   b. Flujo Principal
      i. El actor selecciona la celda que quiere modificar.
      ii. El sistema verifica la celda modificada (E3).
      iii. El sistema modifica el registro correspondiente en la base de datos y actualiza la página de Administración de Años Lectivos. (E2)
   c. Flujo Secundario: Ninguno
   d. Excepciones
      i. E2
         1. Error de conexión con la base de datos.
      ii. E3
         1. Los campos obligatorios tienen valor nulo.
         2. Los campos fueron llenados con tipos de dato incorrectos
3. F2.3. Eliminar
   a. Actores: Administrador
   b. Flujo Principal
      i. El actor presiona el ícono de eliminación del registro que quiere eliminar.
      ii. El sistema verifica si se puede eliminar el registro.
      iii. El sistema elimina el registro correspondiente en la base de datos y actualiza la página de Administración de Años Lectivos. (E2)

   c. Flujo Secundario: Ninguno
   d. Excepciones
      i. E2

1. Error de conexión con la base de datos.

4. F2.4.1. Consulta General
    a. Actores: Administrador
    b. Flujo Principal
        i. El sistema verifica la existencia de datos en la base de datos. (E2)
        ii. El sistema despliega los datos en la página de Administración de Años Lectivos.
    c. Flujo Secundario: Ninguno
    d. Excepciones
        i. E2
            1. Error de conexión con la base de datos.
5. F2.4.2. Consulta A Detalle
    a. Actores: Administrador
    b. Flujo Principal:
        i. El actor escribe su criterio de búsqueda en el campo de texto de filtro de búsqueda.
        ii. El sistema verifica los registros que coinciden con el criterio de búsqueda. (E2)(E3)
        iii. El sistema actualiza la tabla y despliega todos los registros que coincidan con el término de búsqueda.
    c. Flujo Secundario: Ninguno
    d. Excepciones
        i. E2
            1. Error de conexión con la base de datos.
        ii. E3
            1. El término de búsqueda no existe.

El resto de subprocesos no se describen se pueden encontrar en la sección de anexos. No se describen en este documento por su similitud con el subproceso previamente descrito.

3.1.2 Flujos de Trabajo de Soporte

3.1.2.1 Gestión de Configuración y Cambios

La Gestión de Configuración llevará un seguimiento de los objetos generados y sus versiones, así como solicitudes de cambios y las modificaciones respectivas. La línea base está especificada a detalle a continuación.

3.1.2.2 Gestión del Proyecto

3.1.2.2.1 Propósito, Alcance y Objetivos

El propósito de este proyecto es facilitar la administración de calificaciones a los docentes, mediante el uso de herramientas tecnológicas que provean maneras más rápidas, automáticas y seguras de realizar sus procesos internos.

La herramienta desarrollada no debe simplemente solucionar los problemas con respecto a la automatización de Gestión Académica que incluye la Administración de Calificaciones, sino que también debe proveer una Gestión de Administración que incluya:

- Administración de Años Lectivos
- Administración de Estudiantes
- Administración de Personal
- Administración de Roles
- Administración de Cursos
- Administración de Materias
- Administración de Asignación de Materias y Docentes por Curso

3.1.2.2.2 Suposiciones y Restricciones

La información recolectada permitió obtener las siguientes suposiciones y restricciones en relación al sistema que se implementará:

- En general, el sistema será implementado en un ambiente de red local.
- El sistema es
- Gestión de Administración
    - Administración de Años Lectivos
        - Cada año lectivo detalla las fechas de inicio y fin de año lectivo, número de días laborables y vacaciones y fechas para que los docentes registren las calificaciones, teniendo un plazo de 3 días a partir de esa fecha.
    - Administración de Estudiantes
        - Existen 3 tipos de estudiantes: Graduados, Retirados y Cursando.
        - El sistema asciende automáticamente a los estudiantes a su siguiente curso una vez que se crea un nuevo año lectivo.
        - Si los estudiantes ya estaban en último nivel (10mo de básica), cuando se crea un nuevo año lectivo, el sistema automáticamente los pone como “Graduados”
        - Si un alumno se retira, el sistema deja de tomarlo en cuenta en los siguientes años lectivos, pero no lo elimina.
        - Antes de inscribir un alumno, el año lectivo debe ser creado primero, para que se puedan gestionar las calificaciones de manera correcta.
    - Administración de Personal
        - Al igual que los estudiantes, la lista de personal no se elimina. Quedan registradas las personas que trabajaron y trabajan en el plantel.
    - Administración de Roles
        - Los roles de profesor y administrador son predefinidos, permanentes y no editables; con sus respectivos permisos en el sistema.
    - Administración de Cursos
        - Existe un número definido de cursos. Dado que los niveles se extienden desde “Inicial” hasta “10mo de básica” existen 11 cursos con 4 paralelos cada uno.
        - Es opcional usar todos los paralelos existentes.
    - Administración de Materias
        - Los tipos de comportamiento o conductas, se gestionan como materias.

3.1.2.2.3 Entregables del Proyecto

Los entregables del proyecto componen el grupo de objetos que conforman la configuración RUP y que serán generados y utilizados.

- Anexos
    - Producto
    - Modelos
    - Especificaciones
    - Casos de Pruebas
    - Material de Apoyo
- Manual de Usuario
- Manual Técnico
- Manual de Configuración
- Producto

3.1.2.2.4 Estimaciones del Proyecto

Las estimaciones totales del proyecto en relación a recursos humanos se calcularon mediante la siguiente fórmula: **Total horas efectivas * $10**. El resto de estimaciones incluyen uso de software, hardware y varios como luz, internet, entre otros. La *Tabla 3.1-1* muestra el costo real del producto (lo que fue cobrado a los usuarios) vs. El costo ideal que es su valor verdadero.

Tabla 3.1-1

Descripción	Costo Ideal	Costo Real
Recursos Humanos	$3600	$0
Software	$0	$0
Hardware	$400	$400
Varios	$400	$400
Total	$4400	$800

3.1.2.2.5 Cronograma del proyecto

La *Figura 3.1-5* detalla el cronograma y planificación del proyecto durante todas las fases de documentación y desarrollo.

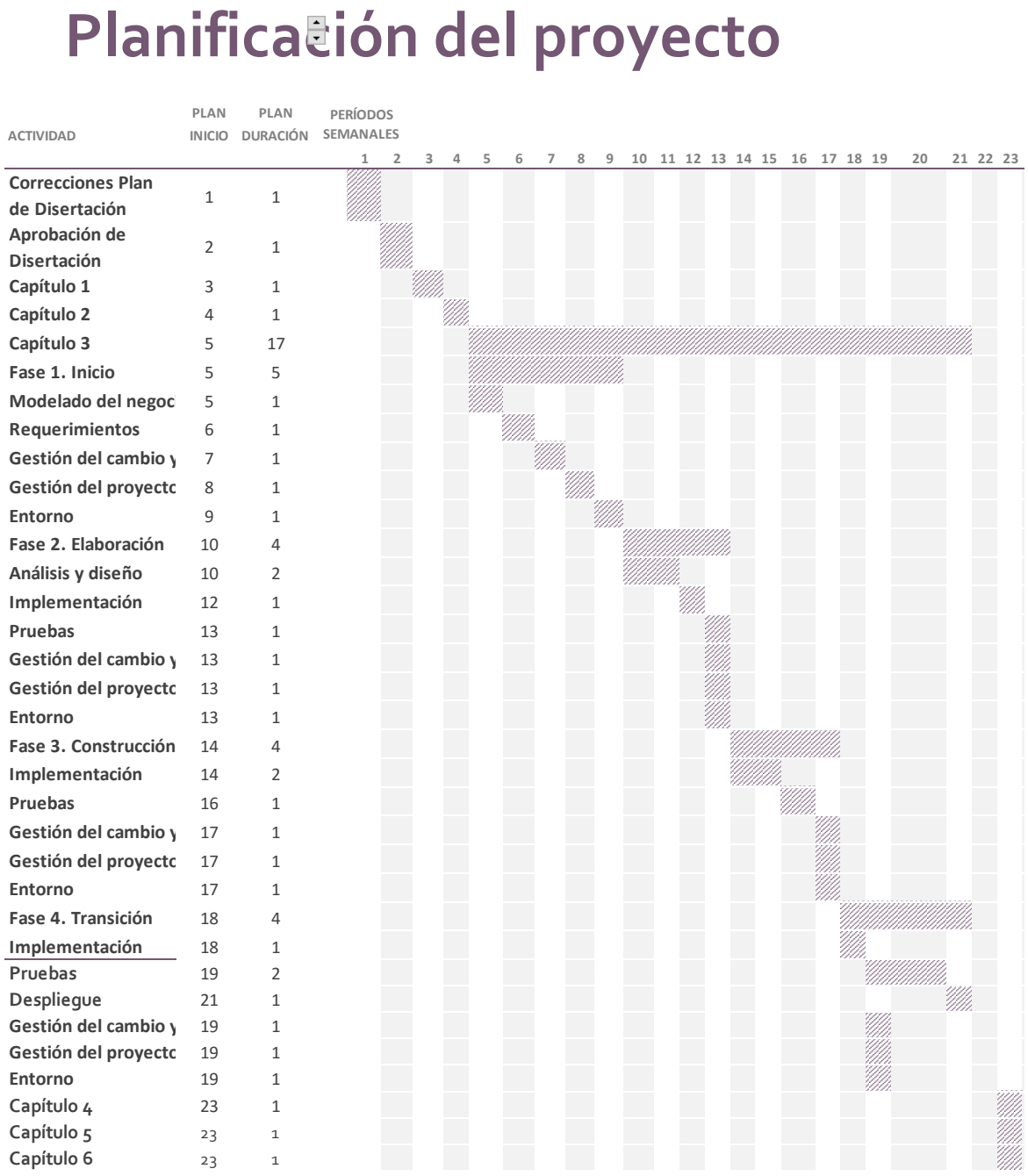

Figura 3.1-5 (Wendy Jaramillo, 2016)

3.1.2.3 Entorno

No hay un número exacto de usuarios que interactúen con el sistema, ya que al ser una institución educativa, podrán ser algunas las personas que realicen operaciones simultánea o individualmente. La velocidad de respuesta del sistema variará dependiendo de los procesos concurrentes que existan en el servidor.

El sistema se desarrolla en una plataforma Web y por lo tanto no importa el sistema operativo en el que trabajen los usuarios, siempre y cuando lo hagan en el explorador Google Chrome de manera que las extensiones visuales funcionen correctamente.

En un **Estándar CMM**[43], el proyecto se puede posicionar en el nivel 2 de forma natural.

[43] CMM: "Capability Maturity Model" es un marco de referencia para la evaluación y mejora de procesos de software. (Tuya, 2007)

3.2 ELABORACIÓN

3.2.1 Flujos de trabajo del proceso

3.2.1.1 Análisis y diseño

3.2.1.1.1 Diagrama de Clases

El diagrama de clases detalla cómo se relacionan los objetos del sistema y sus atributos y funciones correspondientes. La *Figura 3.2-1* detalla el diagrama de clases de manera conceptual (sin funciones ni atributos) mientras que la *Figura 3.2-2* detalla el diagrama de clases completo.

3.2.1.1.1.1 Diagrama de Clases Conceptual

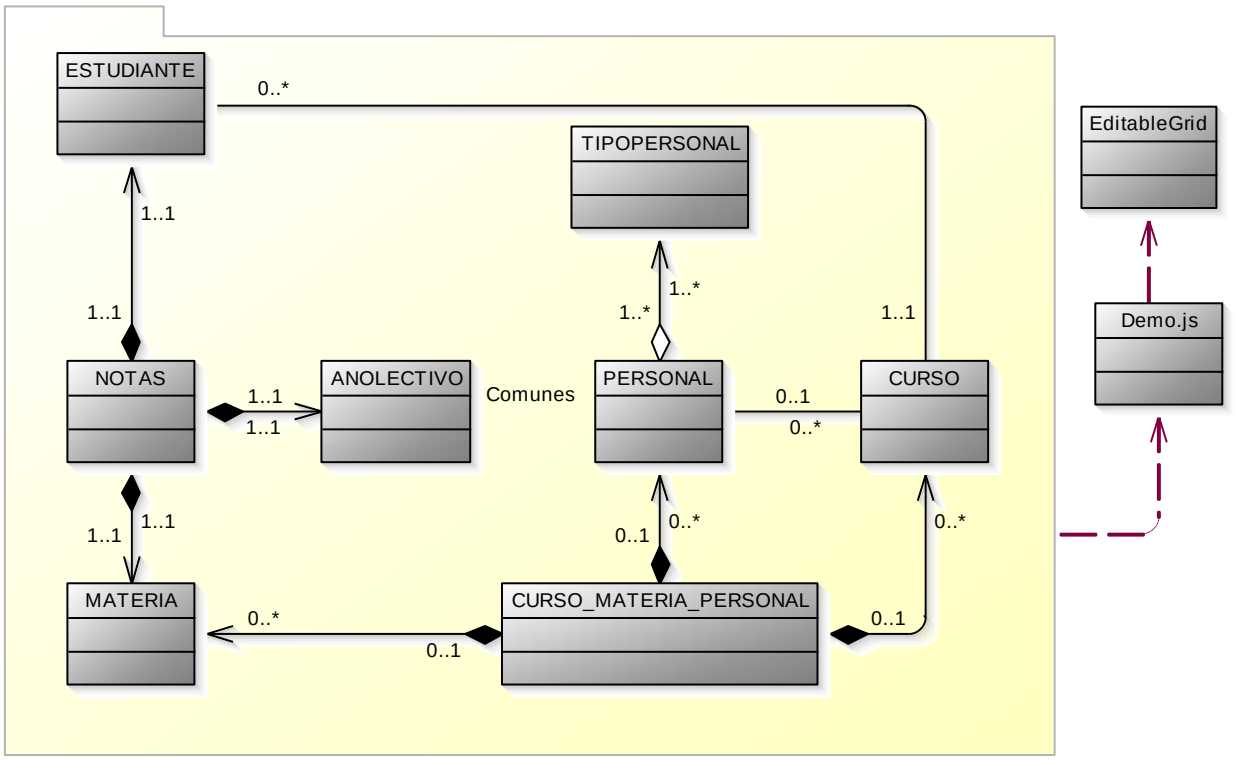

Figura 3.2-1 (Wendy Jaramillo, 2016)

3.2.1.1.1.2 Diagrama de Clases Funcional

En este diagrama se muestran las clases existentes en este proyecto, con sus respectivos atributos y funciones.

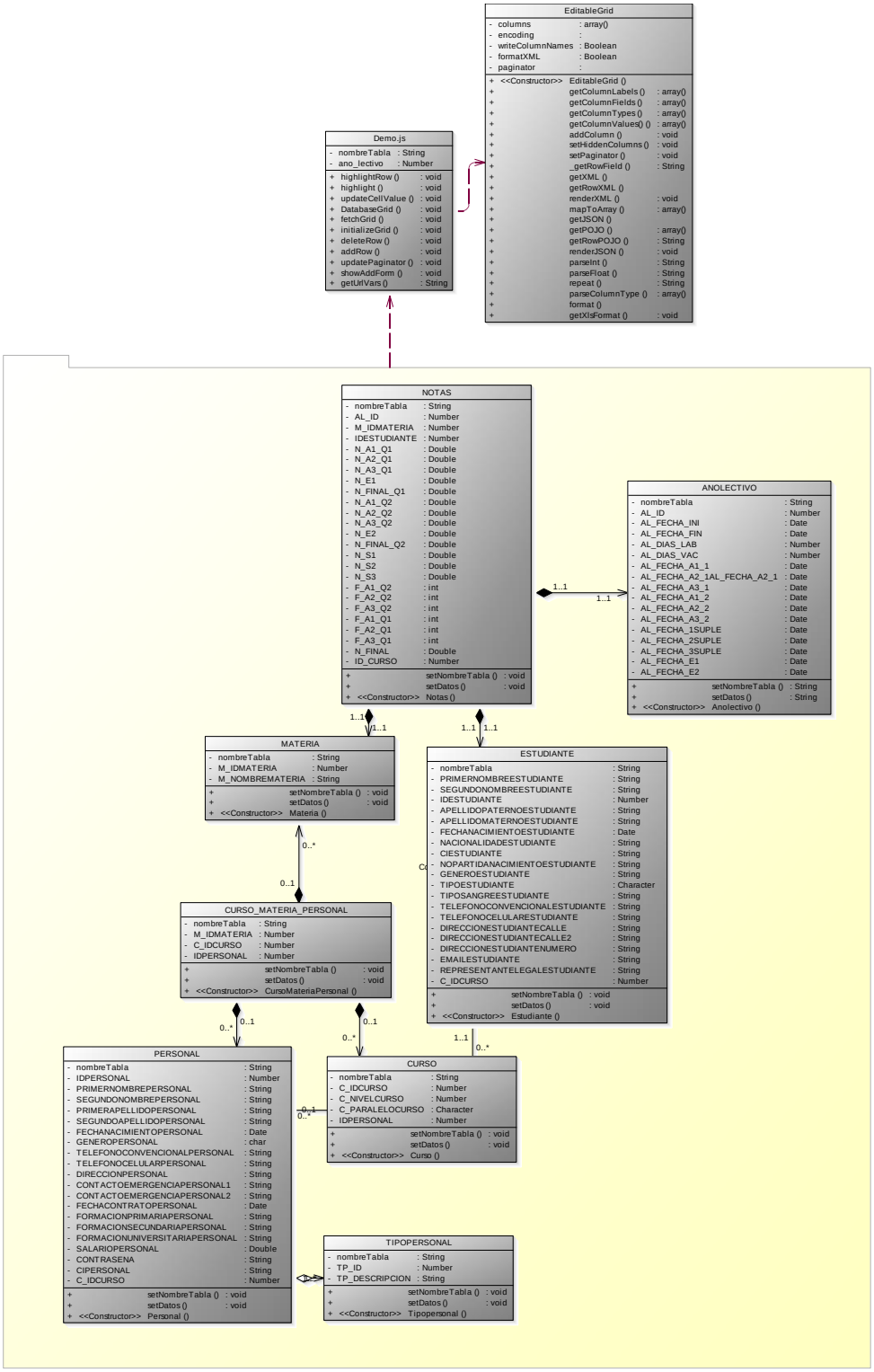

Figura 3.2-2 (Wendy Jaramillo, 2016)

3.2.1.1.2 Diagramas de la Base de Datos

Los diagramas de bases de datos muestran cómo se interrelacionan las distintas tablas-objetos del sistema. La *Figura 3.2-3* muestra el diseño conceptual, la *Figura 3.2-4* muestra el diseño lógico y la *Figura 3.2-5* muestra el diseño físico.

3.2.1.1.2.1 Conceptual

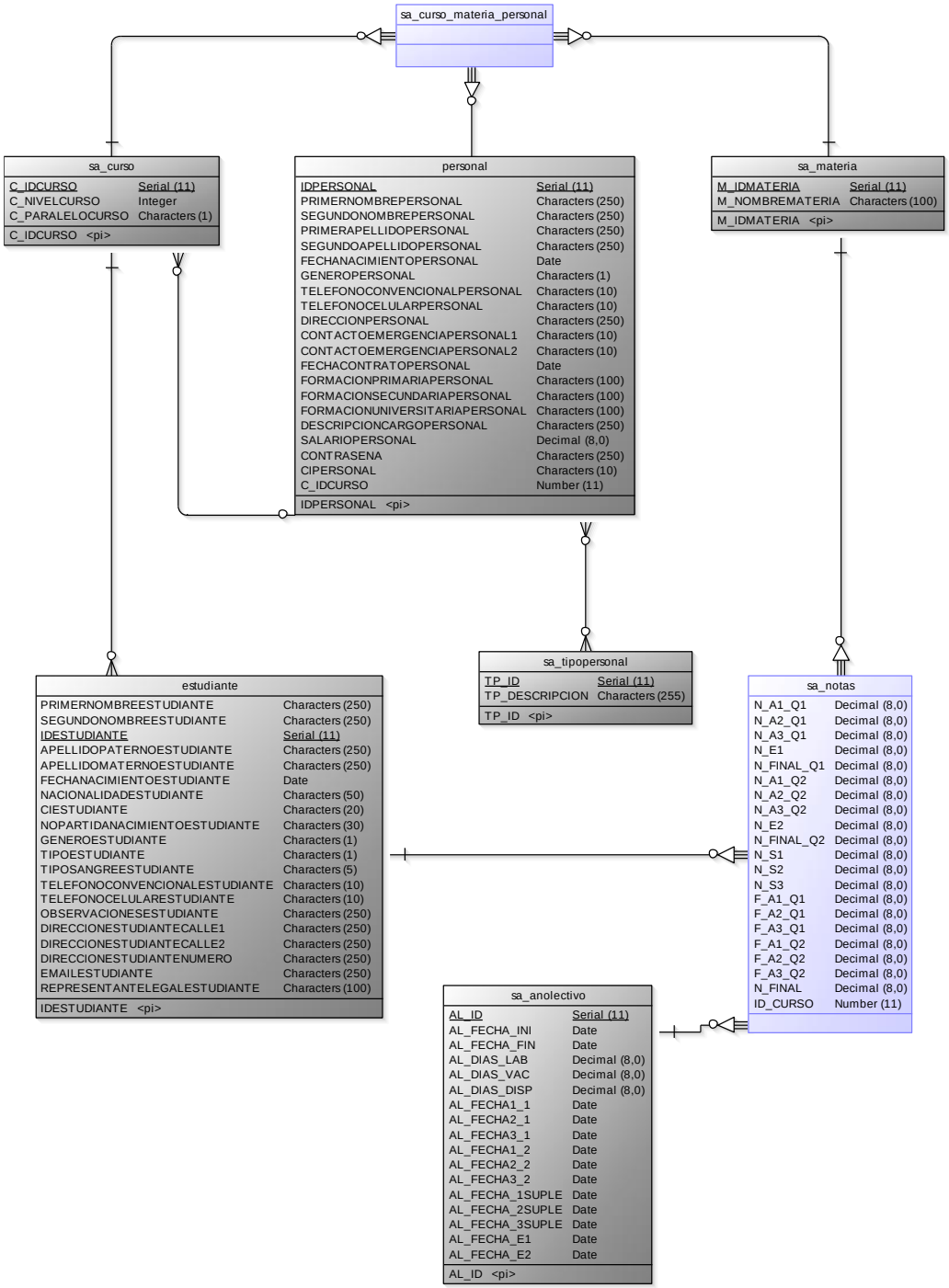

Figura 3.2-3 (Wendy Jaramillo, 2016)

3.2.1.1.2.2 Lógico

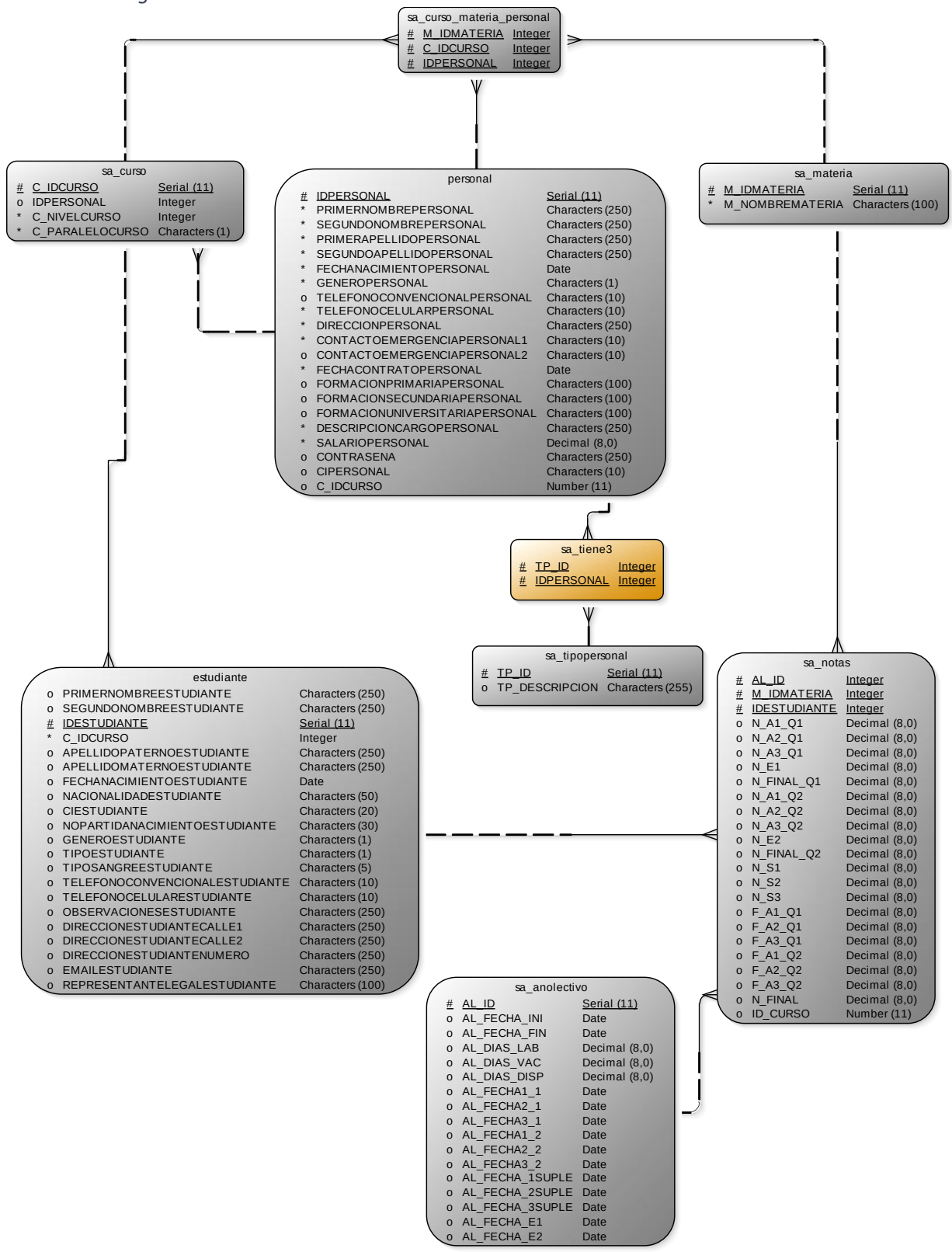

Figura 3.2-4 (Wendy Jaramillo, 2016)

3.2.1.1.2.3 *Físico*

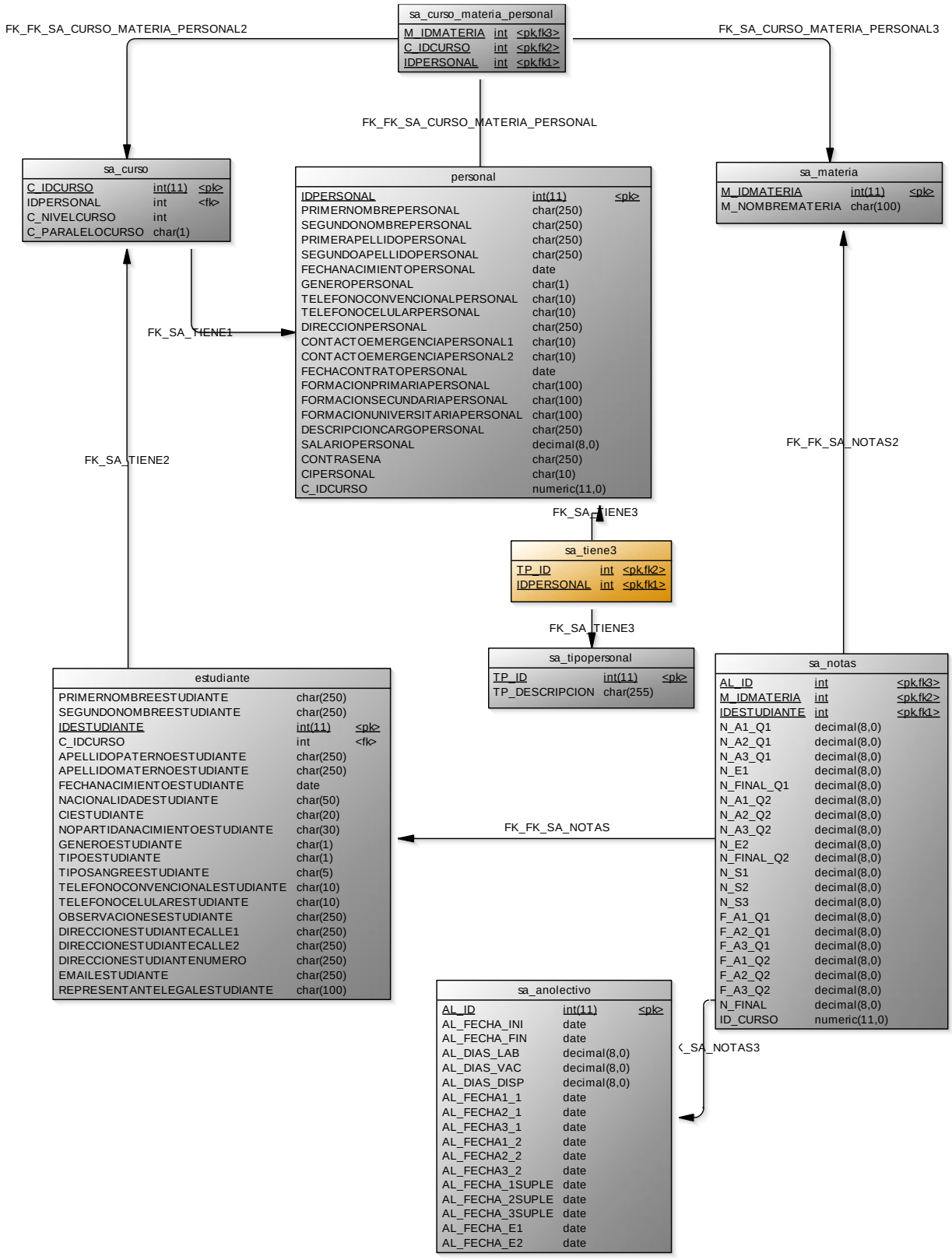

Figura 3.2-5 (Wendy Jaramillo, 2016)

3.2.1.2 Implementación

3.2.1.2.1 Arquitectura del SW

La arquitectura de SW se muestra en la *Figura 3.2-6* (Ojeda, 2015)

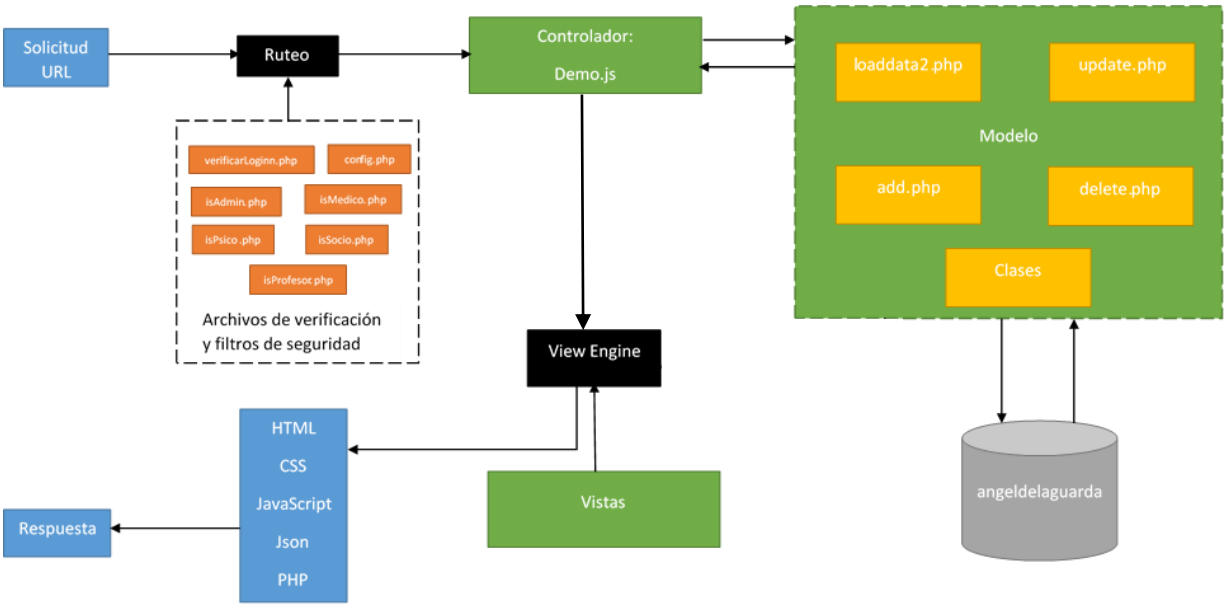

Figura 3.2-6 (Wendy Jaramillo, 2016)

3.2.1.2.2 Prototipo Arquitectónico Ejecutable

Las figuras *Figura 3.2-7*, *Figura 3.2-8*, *Figura 3.2-9*, *Figura 3.2-10*, *Figura* y *3.2-11* muestran los diagramas de flujo para la administración de materias; Inserción, Eliminación, Actualización, Consulta General y Consulta a Detalle respectivamente, similar a las demás administraciones encontradas en la sección de anexos.

Inserción

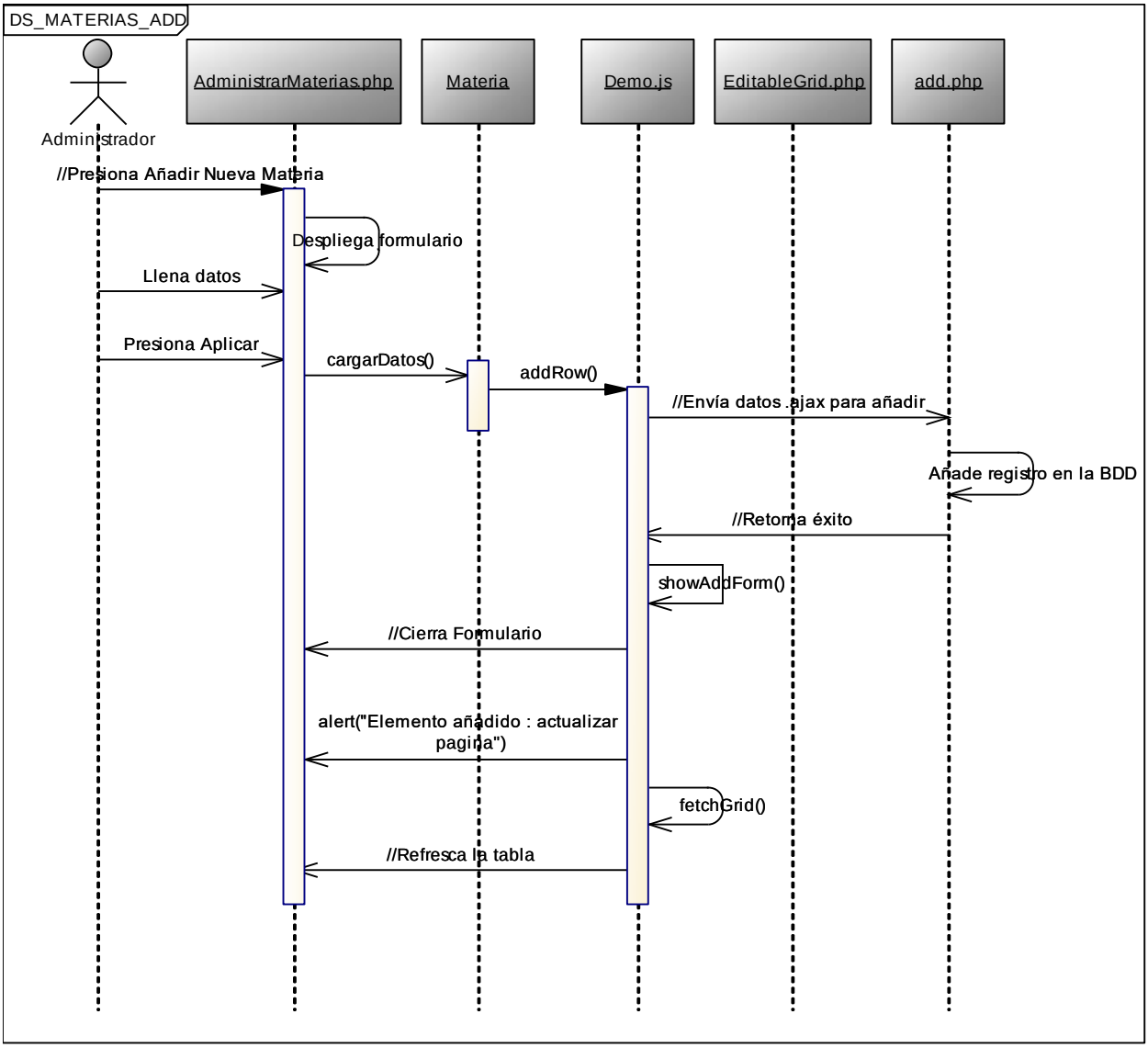

Figura 3.2-7 (Wendy Jaramillo, 2016)

Eliminación

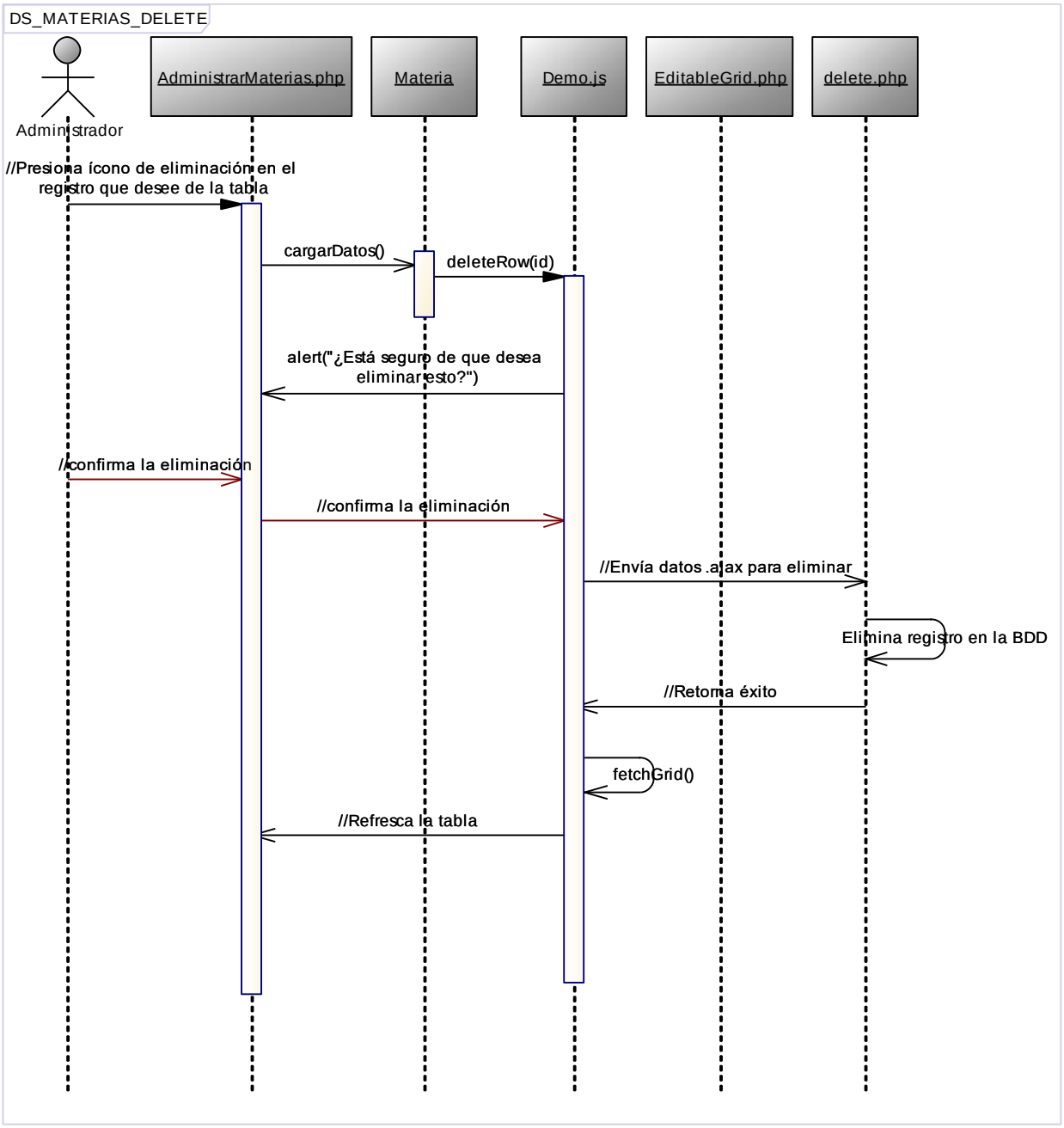

Figura 3.2-8 (Wendy Jaramillo, 2016)

Actualización

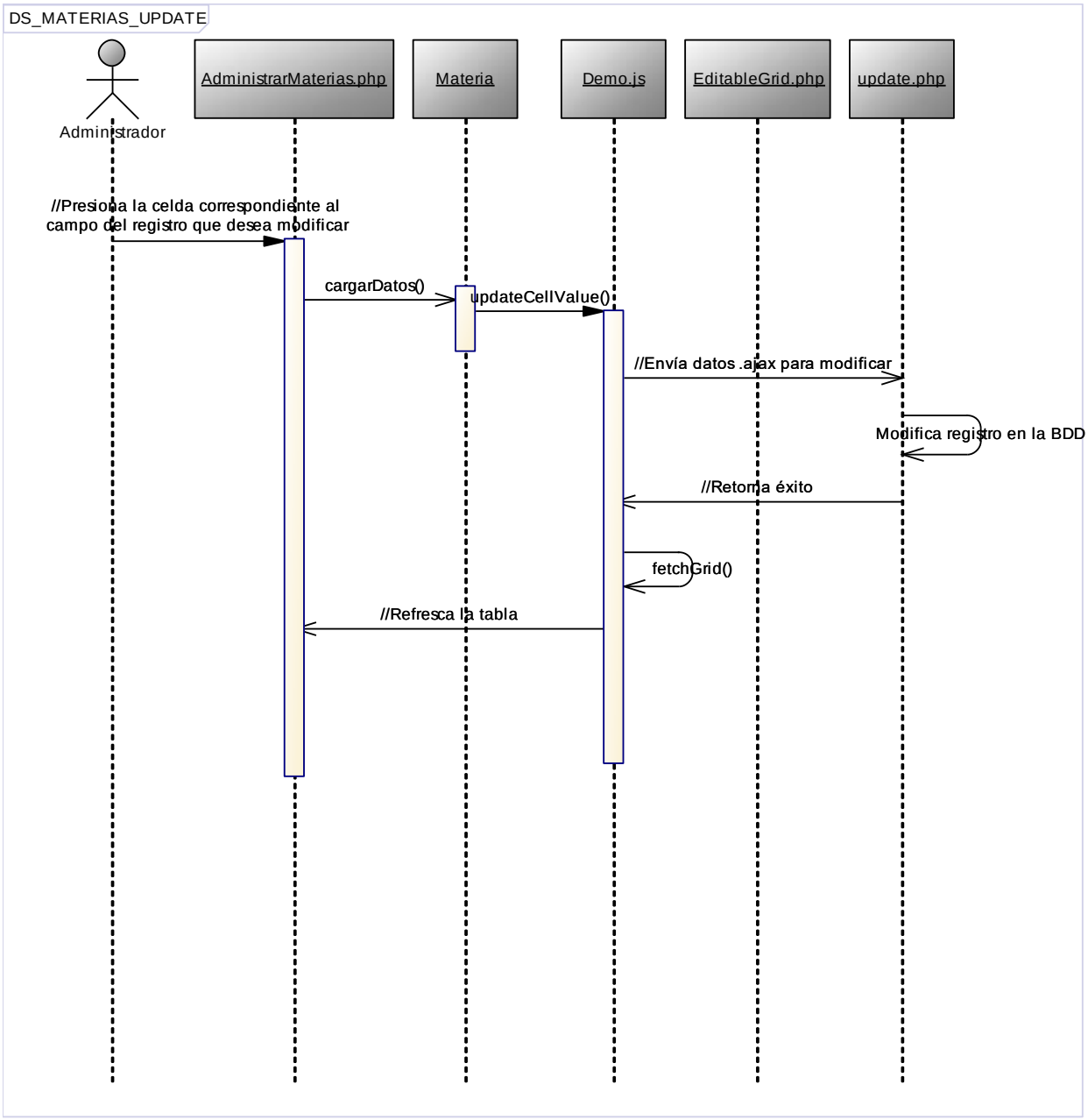

Figura 3.2-9 (Wendy Jaramillo, 2016)

Consulta General

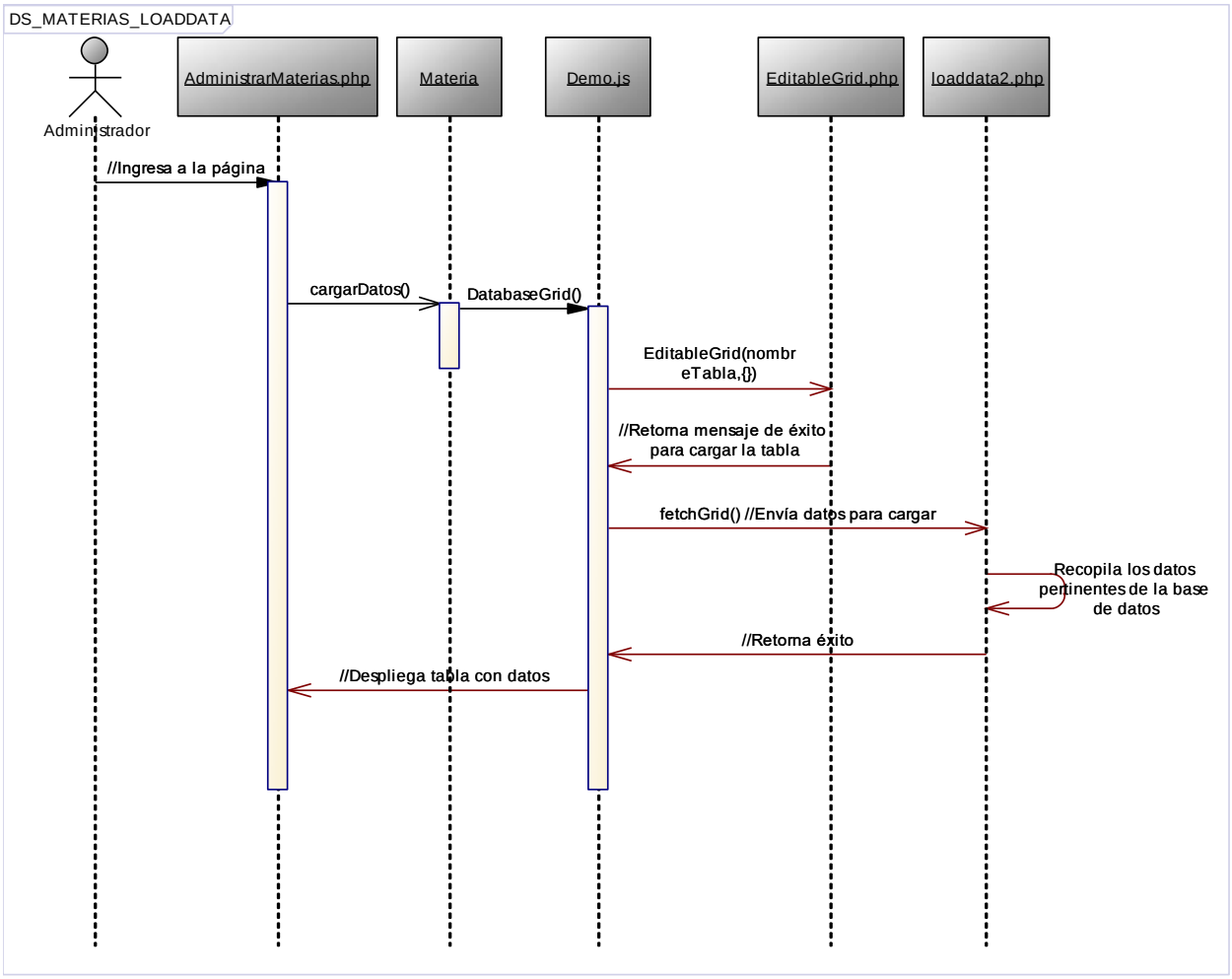

Figura 3.2-10 (Wendy Jaramillo, 2016)

Consulta Específica

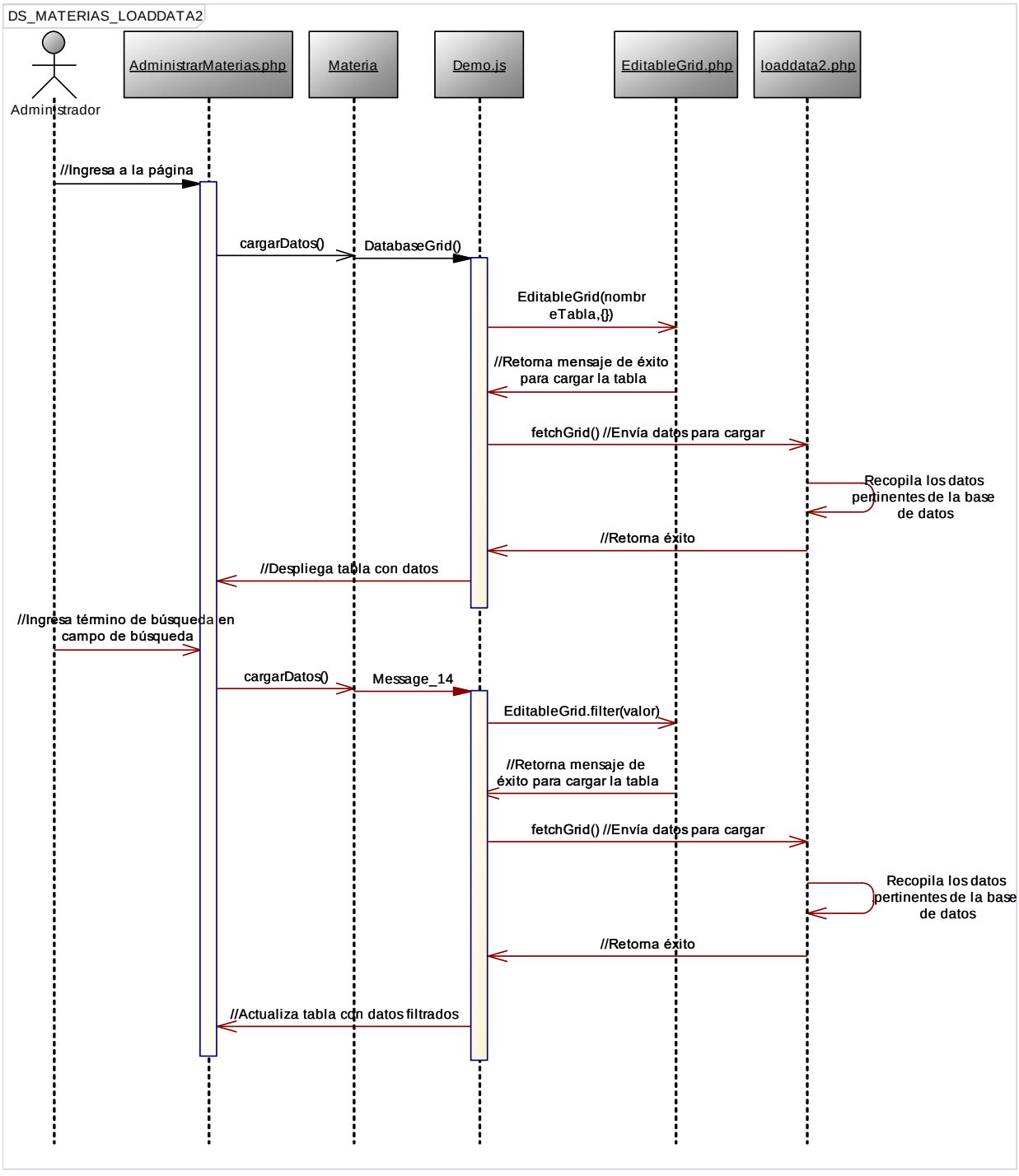

Figura 3.2-11 (Wendy Jaramillo, 2016)

El prototipo funcional muestra la Administración de Materias, el resto de procesos son similares, por lo cual no se detallan, pero están como anexo en el CD del proyecto.

1. **Página de Administración de Materias (*Figura 3.2-12*)**

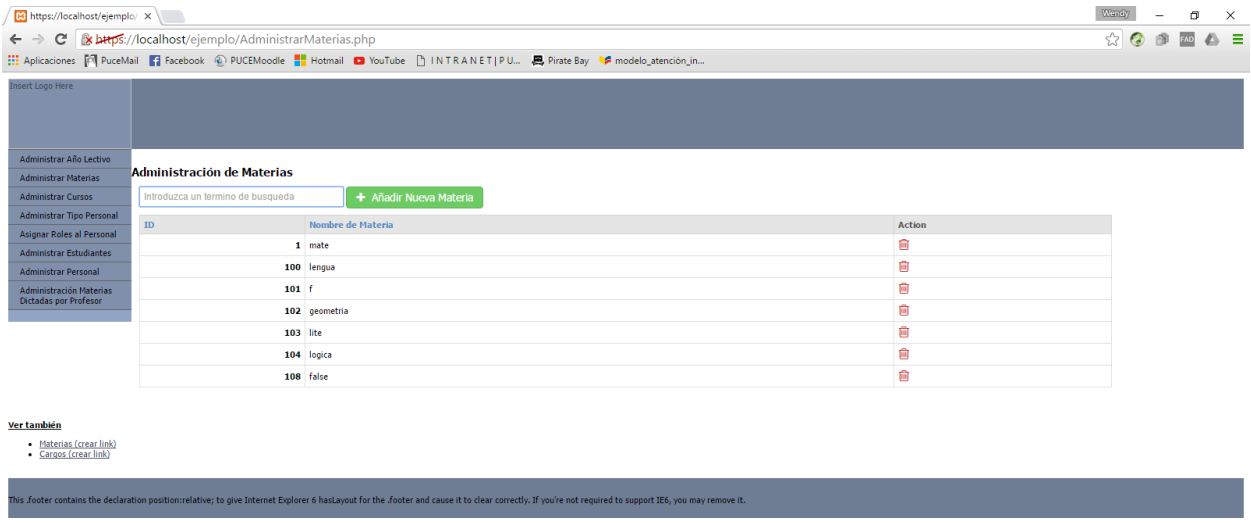

Figura 3.2-12 (Wendy Jaramillo, 2016)

2. **Búsqueda personalizada, por filtro. Criterio de búsqueda: m -> Todas los registros que contengan "m" (*Figura 3.2-13*)**

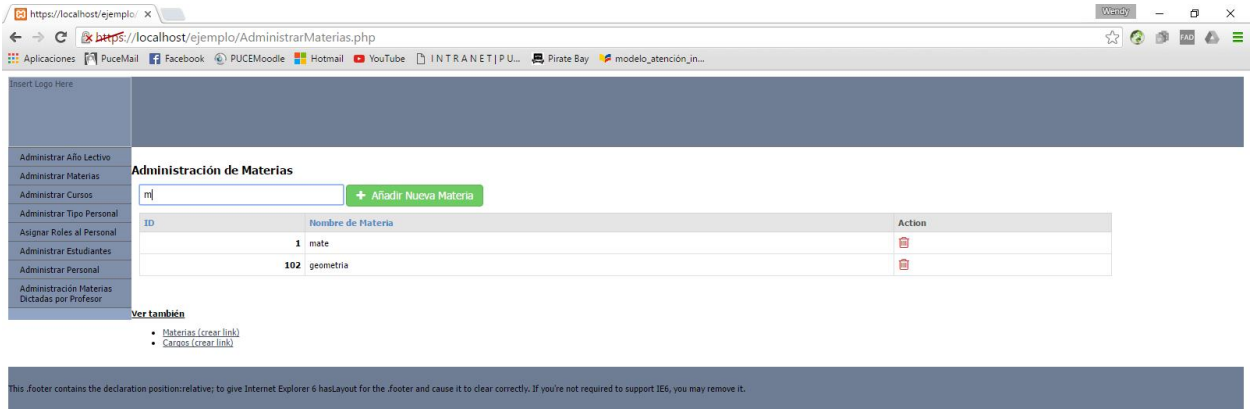

Figura 3.2-13 (Wendy Jaramillo, 2016)

3. **Edición de registros (*Figura 3.2-14*)**

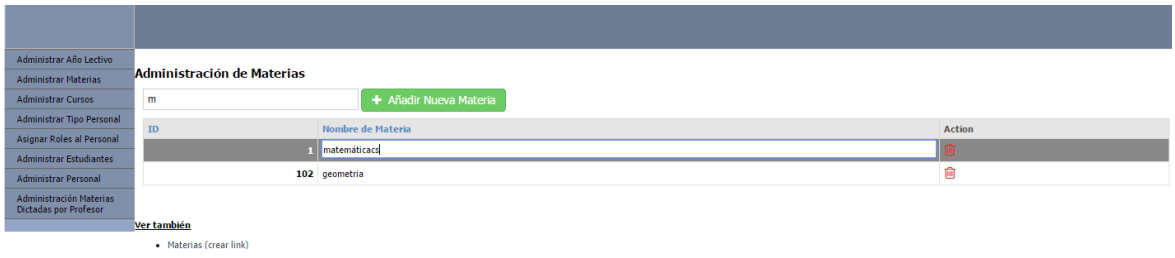

Figura 3.2-14 (Wendy Jaramillo, 2016)

4. **Eliminación (*Figura 3.2-15*)**

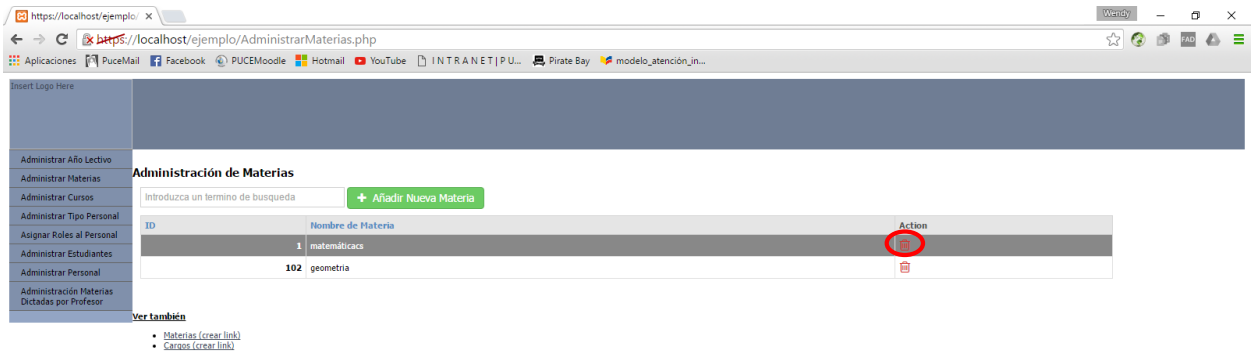

Figura 3.2-15 (Wendy Jaramillo, 2016)

5. **Inserción (*Figura 3.2-16*, *Figura 3.2-17*)**

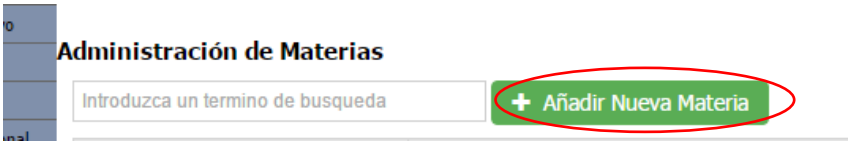

Figura 3.2-16 (Wendy Jaramillo, 2016)

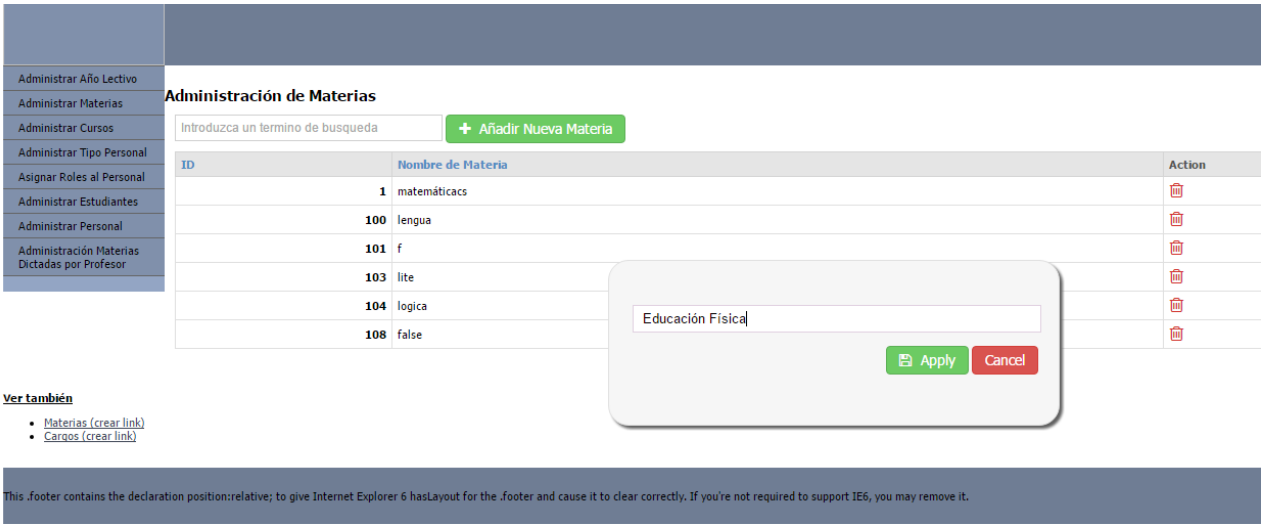

Figura 3.2-17 (Wendy Jaramillo, 2016)

3.2.1.3 Pruebas

Las pruebas se realizarán en base a pruebas de integración basada en hilos, es decir, cada hilo se integra y prueba individualmente y se emplea pruebas de regresión para asegurar que no existan problemas posteriores.

En la fase de elaboración, se propone un plan de pruebas para la gestión de materias. Descrito de la siguiente manera:

1. Plan Prueba – F5.1 Ingreso (*Tabla 3.2-1*)

Precondiciones: Ingreso al sistema exitoso

Tabla 3.2-1 (Wendy Jaramillo, 2016)

Entradas	Resultados Esperados	Funcionalidad
Selecciona "Agregar Nueva Materia".	El sistema muestra el formulario para añadir una nueva materia.	F5.1
El actor llena los campos. Y presiona guardar	El sistema valida y muestra un aviso si los campos llenados son incorrecto. El sistema valida los datos y guarda en la base. Se actualiza la página con el nuevo registro.	F5.1

Postcondiciones: La tabla de consulta general de materias debe actualizarse automáticamente y mostrar la materia nueva ingresada. Base de Datos actualizada.

2. Plan Prueba – F5.2 Modificación (*Tabla 3.2-2*)

Precondiciones: Ingreso al sistema exitoso

Tabla 3.2-2 (Wendy Jaramillo, 2016)

Entradas	Resultados Esperados	Funcionalidad
El actor cambia el campo del registro que desee cambiar y presiona Enter.	El sistema valida los datos y guarda en la base. Se actualiza la página.	F5.2

Postcondiciones: La tabla de consulta general de tareas de restauración debe actualizarse automáticamente y mostrar la tarea modificada. Base de Datos actualizada

3. Plan Prueba – F5.3 Eliminación (*Tabla 3.2-3*)

Precondiciones: Ingreso al sistema exitoso

Tabla 3.2-3 (Wendy Jaramillo, 2016)

Entradas	Resultados Esperados	Funcionalidad
El usuario presiona el ícono de eliminación correspondiente a la fila del registro que desee eliminar	El Sistema valida y verifica el registro en la Base de Datos y elimina. Se actualiza la página.	F5.3

Postcondiciones: La tabla de consulta general de tareas de restauración debe actualizarse automáticamente y no mostrar la tarea eliminada. Base de Datos actualizada.

4. Plan Prueba – F5.4.1 Consulta General (*Tabla 3.2-4*)

Precondiciones: Deben existir registros almacenados en la base de datos

Tabla 3.2-4 (Wendy Jaramillo, 2016)

Entradas	Resultados Esperados	Funcionalidad
Ninguna	Despliegue de información de materias.	F5.4.1

Post-condiciones: ninguna

5. Plan Prueba – F2.4.2 Consulta a detalle (*Tabla 3.2-5*)

Precondiciones: 1. Deben existir registros almacenados en la base de datos

Tabla 3.2-5 (Wendy Jaramillo, 2016)

Entradas	Resultados Esperados	Funcionalidad
Ingreso de parámetro de búsqueda en el campo para filtrar registros.	Despliegue de la información de todas las materias que cumplan con el parámetro	F5.4.2

Post-condiciones: ninguna

3.2.2 Flujos del trabajo de soporte

3.2.2.1 Gestión del cambio y configuraciones

En la fase de elaboración se dieron varias solicitudes de cambios tanto en la base de datos como en los requerimientos del programa (*Tabla 3.2-6*).

Tabla 3.2-6 (Wendy Jaramillo, 2016)

PETICIÓN DE CAMBIO DE CONFIGURACIÓN TSPi			
Nombre:	Wendy Jaramillo	**Fecha:**	Diciembre 2015
Información del Producto			
Nombre del Producto:	Sistema de Gestión Académica para la Unidad Educativa Ángel de la Guarda	**Dueño del Producto:**	Wendy Jaramillo - PUCE
Tamaño del Producto/Cambio:	4	**Medida del Tamaño:**	Niveles
Información del Cambio			
Razón para el cambio			
• El cliente solicitó que en cada aporte, cada responsable de dictar una materia tenga la opción de registrar las faltas de los alumnos.			

• Para las notas, se debe saber a qué curso corresponden, además del estudiante, materia y año lectivo. • Las conductas se califican cómo que fuesen materias. • Un curso puede tener un solo tutor, pero un profesor puede ser tutor de varios cursos, de manera que la tabla cursos debe tener un profesor asignado. • No se usa un campo de días disponibles para calificar para los docentes, ya que este es fijo: 3.
Beneficios del Cambio:
• Mayor eficiencia de la base de datos. • Cumplimiento de requerimientos del cliente.
Impacto del Cambio
Alto: Cambios a nivel de base de datos, modelo, vista y controlador.

Descripción del Cambio:	(Para fuente del código adjunte lista; para cambios del código incluya el número de defectos (de existir) y la lista de segmentos del programa que han sido cambiados o no)

- Cambio en tabla SA_NOTAS
  - Se añaden campos F_A1_Q1, F_A2_Q1, F_A3_Q1, F_A1_Q2, F_A2_Q2, F_A3_Q2, ID_CURSO.
  - Se elimina campo N_CONDUCTA y N_FALTAS
- Cambio en tabla SA_ANOLECTIVO
  - Se elimina campo AL_DIAS_DISP
- Cambio en la tabla SA_CURSO
  - Se añade campo ID_PERSONAL

- Se realizan los cambios correspondientes en:
  - Páginas de administración
  - loaddata2.php
  - add.php
  - delete.php
  - update.php
  - demo.js

Estado:					
Aprobado:	✓	Información adicional:	Ninguna	Desaprobado:	

Información requerida
Requerimientos del cliente

3.2.2.2 Gestión del Proyecto

3.2.2.2.1 Situación del Proyecto

Durante esta fase se concluyó que el proyecto es completamente viable dadas las estimaciones de costos y tiempos. Los cambios no son radicales, de manera que no arriesgan la entrega del producto final.

Los riesgos que surgieron en esta fase fueron:

- Pérdida de energía eléctrica en donde se encuentra instalado el servidor local.
- Pérdida de información del servidor local.
- Encendido/Apagado incorrecto del servidor local.

Y las soluciones propuestas fueron respectivamente:

- Instalación de un **UPS** [44]junto con el servidor.
- Respaldo de la base de datos hacia una memoria externa conectada al servidor.
- Instalación del servidor en un cuarto restringido, donde sólo personal autorizado y capacitado pueda manejar el servidor. Capacitación.

3.2.2.2.2 Plan de desarrollo

El plan de desarrollo ha cambiado con respecto al original de la siguiente manera (*Figura 3.2-18*), dados ciertos retrasos e inconvenientes:

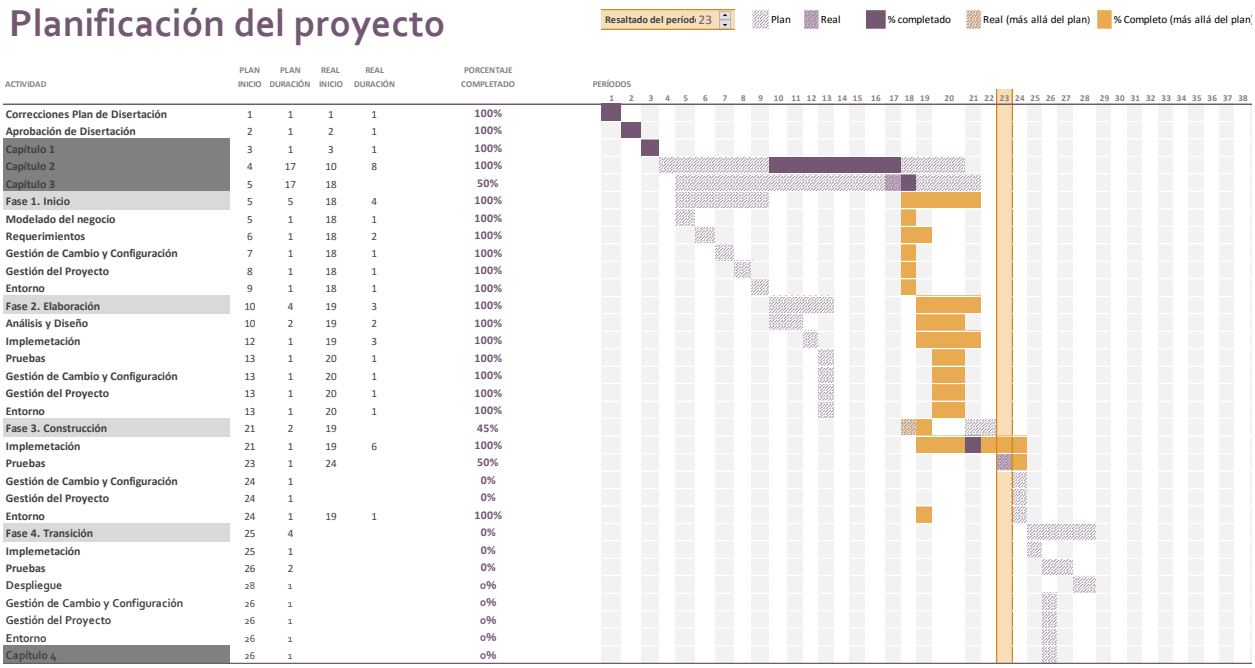

Figura 3.2-18 (Wendy Jaramillo, 2016)

3.2.2.3 Entorno

Para la fase de elaboración, el entorno en el que se desarrolló la aplicación fue sobre un servidor local (Apache) mediante la herramienta **XAMPP**[45].

[44] UPS: "Uninterruptible power supply" es un dispositivo puede proporcionar energía eléctrica limitada durante un apagón eléctrico a todos los dispositivos que tenga conectados. (Wikipedia, s.f.)
[45] XAMPP: Es un servidor independiente de plataforma, software libre. (Wikipedia, s.f.)

3.3 CONSTRUCCIÓN

3.3.1 Flujos de trabajo del proceso

3.3.1.1 Implementación

En la fase de construcción el producto ya está completamente integrado, con estilos visuales fijos cómo se muestra en la Figuras.

Pantalla de Inicio (*Figura 3.3-1*)

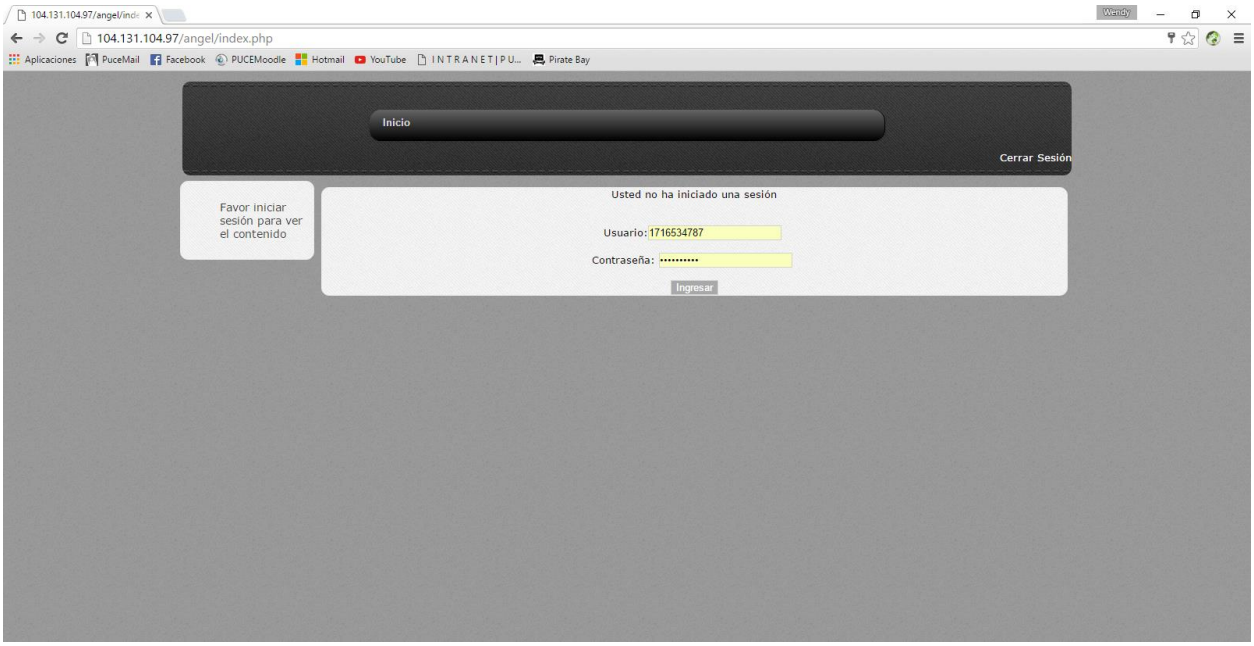

Figura 3.3-1 (Wendy Jaramillo, 2016)

Pantalla de Inicio de Administración (*Figura 3.3-2*)

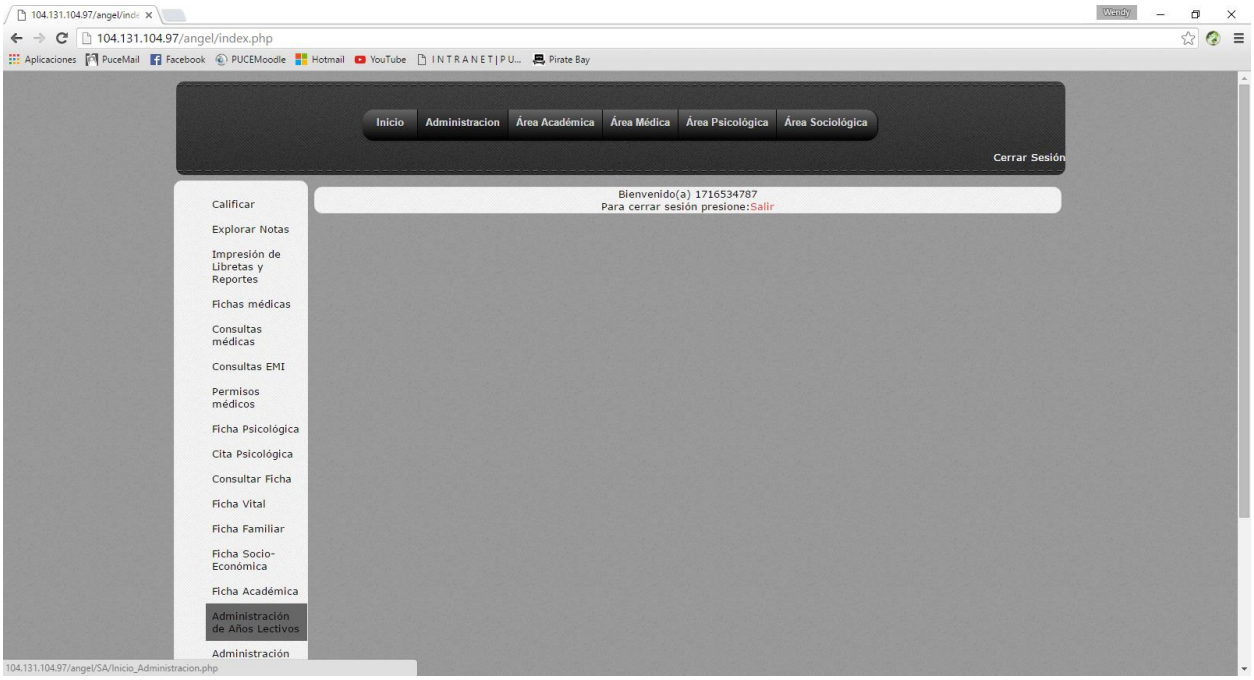

Figura 3.3-2 (Wendy Jaramillo, 2016)

Pantalla de Inicio de Área Académica (*Figura 3.3-3*)

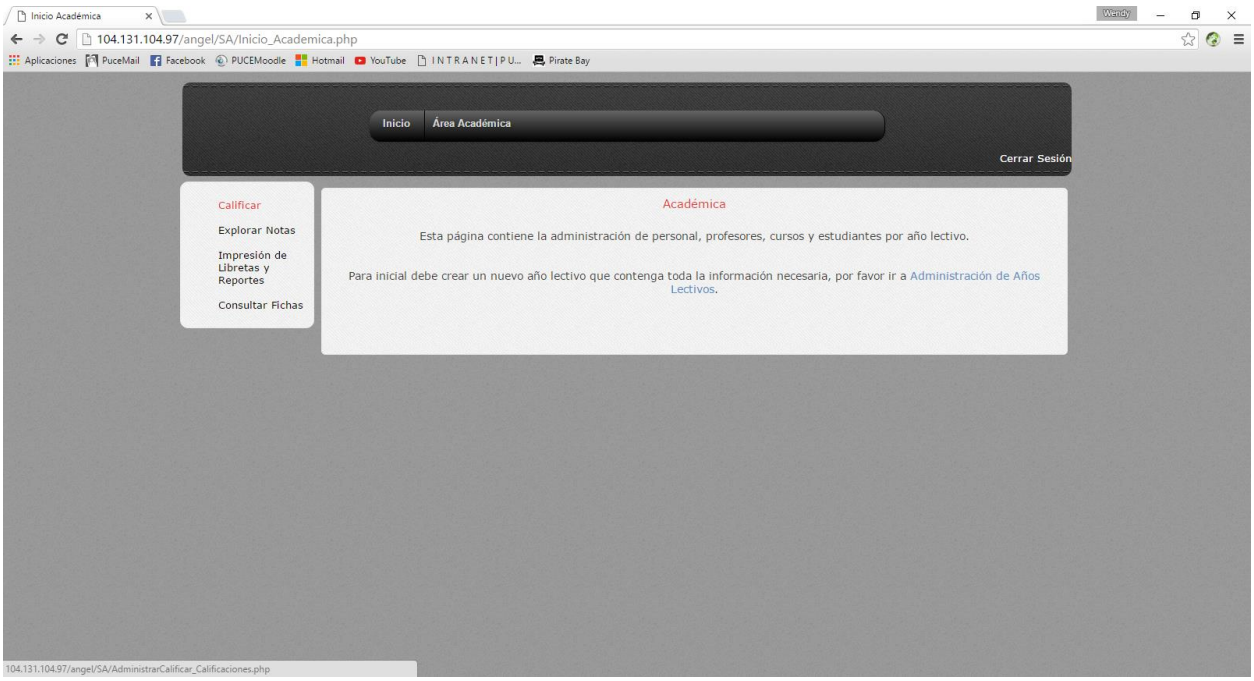

Figura 3.3-3 (Wendy Jaramillo, 2016)

3.3.1.2 Pruebas

En la fase de construcción, las pruebas realizadas fueron tanto de **caja blanca**[46], como de **caja negra**[47]. Las pruebas de caja blanca consistieron en revisar prioritariamente, parte por parte los cálculos que se realizaban para obtener promedios y notas finales mediante triggers y stored procedures de la base de datos. Mientras tanto, las pruebas de caja negra siguieron el plan de pruebas descrito en la *sección 3.2.1.3*, mediante un proceso de entradas y salidas.

Los resultados de las pruebas fueron favorables. Se ejecutaron pruebas con valores tanto correctos como incorrectos y el sistema validó correctamente estos valores, restringiendo cualquier operación en caso de que existan pruebas con valores incorrectos.

Existieron errores con respecto a la codificación y decodificación de caracteres especiales, debido a que el sistema tenía Latin1 como codificación y la base de datos UTF-8.

3.3.2 Flujos del trabajo de soporte

3.3.2.1 Gestión del cambio y configuraciones

Para esta fase hubieron cambios mínimos relacionados a la parte visual-funcional del programa (*Tabla 3.3-1*).

Tabla 3.3-1

PETICIÓN DE CAMBIO DE CONFIGURACIÓN TSPi			
Nombre:	Wendy Jaramillo	**Fecha:**	Diciembre 2015
Información del Producto			
Nombre del Producto:	Sistema de Gestión Académica para la Unidad Educativa Ángel de la Guarda	**Dueño del Producto:**	Wendy Jaramillo - PUCE
Tamaño del Producto/Cambio:	7	**Medida del Tamaño:**	Líneas de Código
Información del Cambio			
Razón para el cambio			
• Las tablas no despliegan caracteres especiales de la base de datos (tildes, ñ's, etc). • No se cargan las tablas que tengan listas en sus celdas.			
Beneficios del Cambio:			
• Despliegue correcto de registros con todos sus caracteres. • Cumplimiento de requerimientos del cliente.			
Impacto del Cambio			
Medio: Cambios a nivel de modelo.			
Descripción del Cambio:	(Para fuente del código adjunte lista; para cambios del código incluya el número de defectos (de existir) y la lista de segmentos del programa que han sido cambiados o no)		

[46] Pruebas de Caja Blanca: Pruebas estructurales que se centran en los detalles procedimentales del software.
[47] Pruebas de Caja Negra: Pruebas funcionales o Entrada/Salida.

- Cambio en loaddata.php
  - Toda tabla que utilice una lista en cualquiera de sus campos y se cargue mediante un query, cambiará su query de manera que este codifique y decodifique correctamente de Latin1 a UTF-8. Por ejemplo:
  - Antes:
    fetch_pairs($mysqli,'SELECT M_IDMATERIA, M_NOMBREMATERIA FROM sa_MATERIA where M_IDMATERIA>10')
  - Ahora:
    fetch_pairs($mysqli,'SELECT M_IDMATERIA, convert(cast(convert(M_NOMBREMATERIA using utf8) as binary) using latin1) FROM sa_materia where M_IDMATERIA>10')
- Cambio en EditableGrid.php
  - Se modifica la función de codificación
  - Antes (línea 87, función getRowField($row, $field)):
    return is_string($value) ? @iconv($this->encoding, "utf-8//IGNORE", $value) : $value;
  - Después(línea 87, función getRowField($row, $field)):
    return is_string($value) ? @iconv("iso-8859-1", "utf-8//IGNORE", $value) : $value;

Estado:					
Aprobado:	✓	Información adicional:	Ninguna	Desaprobado:	

Información requerida
Requerimientos del cliente

3.3.2.2 Gestión del proyecto

3.3.2.2.1 Situación del Proyecto

El proyecto en este punto sigue siendo completamente viable. Debido a que las mayores preocupaciones tanto del cliente como del desarrollador ya han sido atendidas, el sistema ya ejecutó exitosamente el plan de pruebas propuesto en la fase de elaboración.

Dados estos logros, se plantearon los primeros Manuales de Usuario y Configuración/Instalación que explican el funcionamiento y uso del sistema y cómo instalarlo, respectivamente.

3.3.2.2.2 Plan de desarrollo

El plan de desarrollo se ha cumplido en el tiempo estimado y no ha tenido cambios como se muestra en la *Figura 3.3-4*

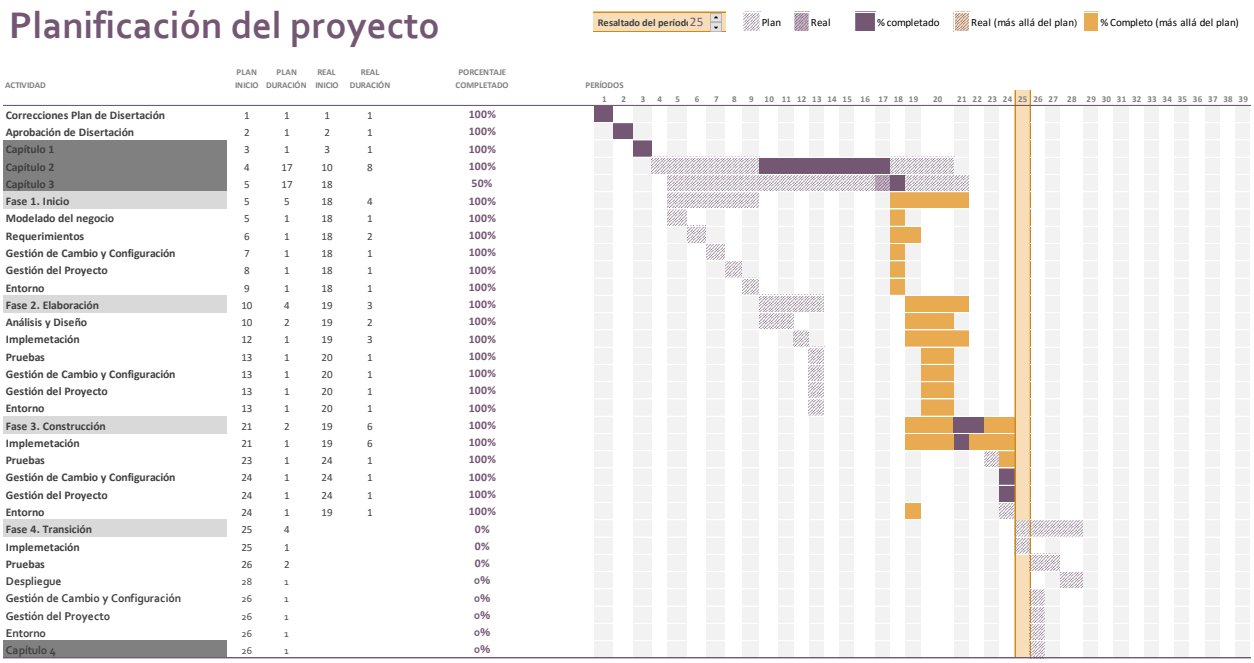

Figura 3.3-4 (Wendy Jaramillo, 2016)

3.3.2.3 Entorno

En la fase de construcción el sistema se desarrolló y probó en una plataforma Linux en un servidor de prueba en la nube. Dado que no era conveniente hacer las pruebas directamente en el servidor de producción (que también está configurado en una plataforma Linux).

También se aplicaron los patrones de diseño:

- Singleton: se usó para la conexión a la base de datos (*Figura 3.3-5*), en vez de realizar todo el proceso de conexión varias veces, sólo se crea un objeto de conexión. Se usó para restringir el número de instancias que se crean para conectarse a la base de datos.

```php
class database {

    private $dbName = null, $dbHost = null, $dbPassword = null, $dbUser = null;
    private static $instance = null;

    private function __construct($config = array()) {
        // Please note that this is Private Constructor
        $this->dbName = $config['db_name'];
        $this->dbHost = $config['db_host'];
        $this->dbUser = $config['db_user'];
        $this->dbPassword = $config['db_password'];

        // Your Code here to connect to database //
        $mysqli = mysqli_init();
        $mysqli->options(MYSQLI_OPT_CONNECT_TIMEOUT, 5);
        $mysqli->real_connect($config['db_host'],$config['db_user'],$config['db_user'],$config['db_name']);
    }

    public static function connect($config = array()) {
        // Check if instance is already exists
        if(self::$instance == null) {
            self::$instance = new database($config);
        }
        return self::$instance;
    }
}
```

Figura 3.3-5 (Wendy Jaramillo, 2016)

- Iterator: se usó para los dropdown list, los cuales se usaban en varias páginas. No se repitió el query en cada página, sólo se instanció “selectCursos.php”(*Figura 3.3-6*).

selectCursos.php

```php
<?php
require_once('../config.php');
$mysqli = mysqli_init();
$mysqli->options(MYSQLI_OPT_CONNECT_TIMEOUT, 5);
$db1 = database::connect($config);
$mysqli->set_charset("utf8");
ini_set('default_charset', 'utf-8');

    $sql = "SELECT * FROM angeldelaguarda.sa_curso";
$result = $mysqli->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo '<option value ="' . $row['C_IDCURSO'] .'" '. '>' .
        "Nivel ".$row['C_NIVELCURSO'] .": Paralelo " .$row['C_PARALELOCURSO'].'</option>';
    }
} else {
    echo "0 results";
}

?>
```

Figura 3.3-6 (Wendy Jaramillo, 2016)

- Mediator: Debido a que cada objeto se maneja la final como un objeto EditableGrid, cada vez que cambia algo en el objeto EditableGrid (*Figura 3.3-7*), también cambia en el objeto original mediante “demo.js” (*Figura 3.3-8*).

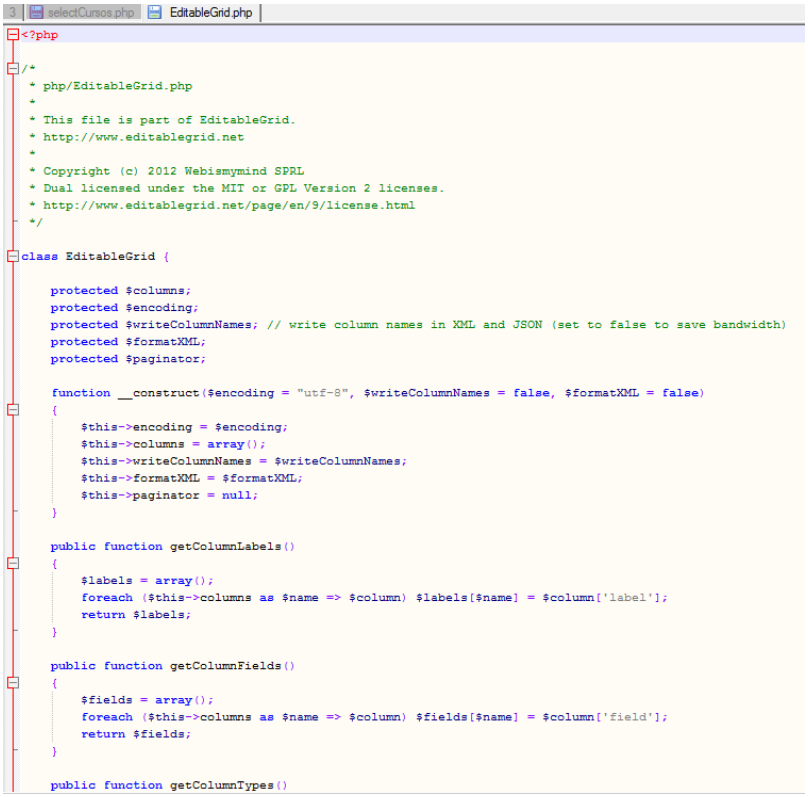

Figura 3.3-7 (Wendy Jaramillo, 2016)

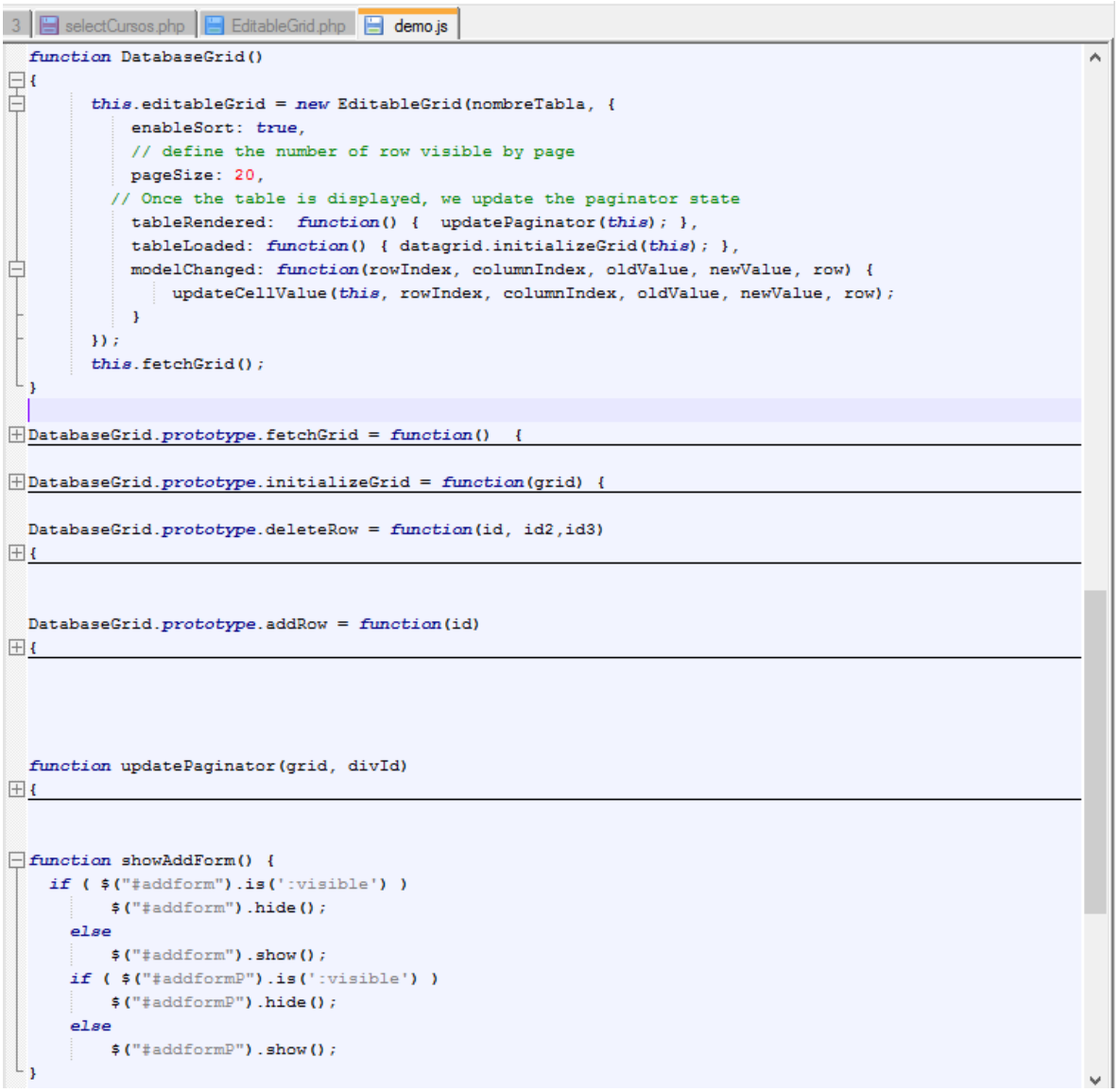

Figura 3.3-8 (Wendy Jaramillo, 2016)

- Prototype: la clase EditableGrid puede clonarse con distintas tablas instanciándola y llamando al constructor con el nombre de la tabla que se desee en "demo.js" (*Figura 3.3-9*)

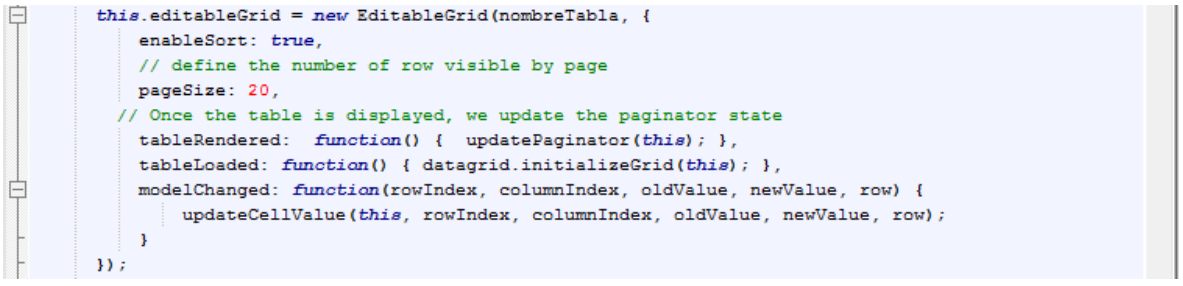

Figura 3.3-9 (Wendy Jaramillo, 2016)

3.4 TRANSICIÓN

3.4.1 Flujos de trabajo del proceso

3.4.1.1 Implementación

En esta fase se instaló el servidor con el que el plantel trabajará y estará en producción hasta que haya que migrarlo o cambiarlo. La Unidad Educativa Ángel de la Guarda recibió la donación de un CPU, configurado como servidor Linux, un UPS para garantizar el correcto funcionamiento en caso de fallas eléctricas y la verificación de más de 10 máquinas conectadas en red para que puedan hacer uso del sistema en paralelo.

La base de datos en MySQL y el sistema como tal, fueron instalados y probados en el servidor mencionado. Se dio capacitación a los usuarios y clientes, y se definieron y entregaron los documentos respectivos al manual de usuario y manual de instalación, los cuales se pueden encontrar en los anexos de este proyecto.

3.4.1.2 Pruebas

Se realizaron pruebas reales o "pruebas beta" en la Unidad Educativa Ángel de la Guarda simultáneamente con la capacitación del personal del plantel educativo que haría uso del sistema. El sistema funcionó correctamente en todos sus procesos.

Estas pruebas fueron 100% de caja negra, se constató paralelamente con la capacitación, que el sistema emite mensajes de error al aplicar valores incorrectos en cada registro, así como mensajes de éxito en caso de ingresar valores correctos.

Las pruebas de caja negra se hicieron para dos cursos completos: 7mo de básica "B" y 5to de Básica "B", para los cuales se ingresaron todos sus estudiantes, todas las materias correspondientes, profesores y notas registradas hasta la fecha.

El procedimiento en orden fue:

1. Creación de año lectivo.
2. Creación de usuarios y asignación como profesores.
3. Creación de materias.
4. Asignación de materias y docentes a los cursos 7mo de Básica "B" y 5to de Básica "B".
5. Ingreso de Estudiantes (en donde se les asignó el curso al que pertenecen).
6. El sistema autogeneró correctamente en la base de datos los registros para las notas por cada creación de alumno.
7. Algunos docentes y administradora (durante la capacitación) ingresaron todas las notas de los alumnos ingresados hasta la fecha.
8. Se imprimieron los primeros bosquejos de libretas.

3.4.1.3 Despliegue

El personal capacitado demostró que puede usar el sistema sin ninguna dificultad. El certificado de finalización del proyecto (*Figura*) se puede encontrar en los anexos del proyecto. Dado que el proyecto ha finalizado, el período de servicio de soporte gratuito ofrecido al plantel es de 1 año, contando desde la fecha de finalización establecida en el certificado.

Quito, 2016 -04-20

CERTIFICADO

Quiero manifestar mi profundo agradecimiento a la labor que realizó la señorita Wendy Jaramillo, estudiante de la Carrera de Ingeniería en Sistemas de la Pontifica Universidad Católica del Ecuador; quien propuso, desarrolló e implementó una solución integrada y funcional para la problemática de Gestión Académica de nuestra institución, siguiendo un plan estructurado que cubre las expectativas principales de nosotros como clientes y usuarios.

Certifico que la señorita **Wendy Elizabeth Jaramillo Wilches con CI: 1725031439**; instaló, probó, capacitó al personal de la Unidad Educativa "Ángel de la Guarda" y entregó satisfactoriamente el Sistema de Gestión Académica, tomando como cursos piloto al 5to año de educación básica "B" y 7mo año de educación básica "B"; en Abril del presente año.

Se expide la presente a solicitud de la interesada para los fines que crea convenientes, siempre y cuando esté en honor a la verdad.

Atentamente,

ESCUELA PARTICULAR
"Ángel de la Guarda"
Vela 147 y Exposición
Telefax: 02282698

Hna. Hilda Aguinsaca
Representante Legal de la Unidad Académica "Ángel de la Guarda"
CI: 1102564265

Figura 3.4-1 (Wendy Jaramillo, 2016)

3.4.2 Flujos del trabajo de soporte

3.4.2.1 Gestión del cambio y configuraciones

No existieron cambios ni configuraciones en esta fase.

3.4.2.2 Gestión del proyecto

El proyecto fue finalizado y entregado en su totalidad exitosamente.

3.4.2.3 Entorno

El entorno del sistema en la fase de transición fue el entorno definitivo.

- Servidor Ubuntu LAMP
    - Memoria RAM de 2GB.
    - Disco duro de 500GB.
- Base de datos MySQL
- Sistema basado en PHP en su mayoría.

La documentación completa se encuentra en la sección de Anexos dentro del CD de este proyecto.

Capítulo 4. CONCLUSIONES Y RECOMENDACIONES

4.1 DE LA DISERTACIÓN

Luego de haber aplicado la metodología RUP, se puede concluir que esta es una de las metodologías más completas tanto para el desarrollador como para el cliente, en tanto y en cuanto a documentación y seguimiento del sistema se trata, ya que brinda la posibilidad de ir armando una aplicación o sistema desde el nivel de abstracción más bajo (conceptos claros para el cliente) hasta el nivel más alto (programación, diagramas de funcionamiento, etc). Y en cada fase del ciclo de vida hubieron hitos claves en los cuales el cliente y el desarrollador acordamos cómo continuar con el sistema de la mejor manera para las dos partes.

El sistema fue exitoso tanto en su desarrollo como en producción. El proyecto implementó algunos de los patrones de diseño propuestos en el capítulo 2, los cuales fueron de gran ayuda para optimizar y ahorrar líneas de código.

Por ejemplo, se implementó el patrón de diseño Singleton para la conexión a la base de datos, todas las páginas hacen referencia a una sola instancia. También se puede encontrar el patrón Iterador, Mediador y Prototipo, los cuales se encargan de construir objetos que puedan proveer una interfaz estándar simple para iterar cualquier tipo de información contable, desarrollar un objeto que se comunique o que sea mediador de cambios para una colección de objetos similares y crear objetos de tal manera que un objeto inicial o prototipo puede ser copiado y clonado más eficazmente que creando una nueva instancia, respectivamente; estos patrones son usados en su mayoría por la clase EditableGrid y todos los componentes que esta conlleva.

El uso del framework EditableGrid facilitó el desarrollo del sistema, por lo cual, personalmente, aconsejo hacer uso de herramientas ya existentes en el mercado. Y qué mejor si estas herramientas son gratuitas y el uso que les demos puede ser un aporte a la comunidad.

Hubo dificultades principalmente con la comunicación con el cliente, ya que como el proyecto se desarrolló en medio de dos años lectivos, algunos requerimientos cambiaron en la fase de construcción. Por esto se recomienda estar en contacto con los requerimientos del cliente por lo menos una vez en cada fase del proyecto, de esta manera los cambios pueden realizarse a tiempo.

Es recomendable que la documentación (fase inicial y de elaboración principalmente) antecedan la programación del proyecto, así también se evitan cambios innecesarios del proyecto y se mantiene una visión más abstracta para enseñar al cliente.

Para las personas que tengan la oportunidad de leer este trabajo de disertación y usarla para sus trabajos y posibles mejores sistemas, yo recomiendo especialmente poner atención en la relación que se establece con el cliente, para poder entregar un producto de calidad y en el tiempo establecido, es de vital importancia saber exactamente los requerimientos del cliente durante todo el proceso, ya que siempre habrán malos entendidos o cambios de último momento. De manera que los entregables para cada fase descritos en la metodología, son la mejor forma de acercarnos a quien nos contrata y verificar cómo estamos haciendo las cosas

4.2 De la Carrera

En este proyecto se aplicaron muchos conceptos vistos, estudiados y trabajados a lo largo de la carrera de Ingeniería en Sistemas. Sobre todo, el área de Base de Datos, Programación e Ingeniería de Software. Por tanto, trabajar con todas las herramientas implementadas no supuso un reto mayor. Por otro lado, lo que sí supuso un reto grande, fue haber trabajado directamente con los clientes, haber experimentado individualmente la relación cliente-desarrollador de la cual vimos mucho en clases, pero por primera vez pude aplicar.

Recomiendo a mis futuros colegas, Ingenieros en Sistemas, que una vez que decidan estudiar esta carrera estén preparados para abrir su mente a muchas áreas, porque en la actualidad prácticamente todo funciona con sistemas, y los profesionales en esta rama tenemos que convertirnos en "todólogos" para poder cumplir con las expectativas del cliente y la sociedad como tal.

4.3 De la PUCE

La Pontificia Universidad Católica del Ecuador ha sido un gran soporte para la realización de este proyecto. La biblioteca y los laboratorios de computación del edificio de la Facultad de Ingeniería han tenido sus puertas abiertas siempre para brindar toda la ayuda e información que tienen.

Es un lugar maravilloso en el que un estudiante puede desenvolverse y crecer para ser un futuro profesional de éxito, siempre y cuando se comprometa y se lo proponga.

4.4 De la Organización

La Unidad Educativa Ángel de la Guarda es administrada por monjitas, es una institución sin fines de lucro a la cual llegué a conocer muy bien; y debo decir, a la cual admiro y respeto por la labor que cumple con sus estudiantes y con la sociedad.

Las personas que trabajan aquí, quienes hacen uso del sistema fueron siempre personas atentas y sí se interesaron en cómo manejar el sistema y brindar toda la información necesaria para que este proyecto tenga frutos.

Personalmente, recomiendo que se siga prestando atención a esta institución, ya que están en vías de crecimiento y aún les falta mucho por prosperar.

4.5 De la Sociedad

Este proyecto no fue destinado para ser usado por una sóla institución, sino para poder ser aplicado y mejorado en muchas más instituciones educativas del país. Para que estas unidades educativas puedan crecer y desarrollarse a la altura de otras que manejan todo mediante automatización.

El crecimiento de una sociedad empieza por dar de nuestro potencial a esas personas e instituciones que aún no han podido crecer.

Índice/Glosario

A

B

C

D

E

G

H

I

L

M

N

O

P

Q

R

S

Secure Sockets Layer (SSL)
Protocolo que soporta conexiones encriptadas……**27**

SimpleXML
rápido y fácil de aprender modo de interactuar con documentos XML…….. **10**

Singleton Pattern
Es un patrón que se usa para restringir el número de veces que un objeto específico puede ser creado cada vez, proveyendo acceso a una instancia compartida propia. .. 38

SQL (Structured Query Language)
Es un DML no procedural que permite al usuario de manera fácil definir o modificar la estructura y los datos de la base……..**18**

SQLite
Sistema de gestión de base de datos ligero (~275Kb)……..**10**

Stored Procedure
Elementos de una base de datos que corresponden a un conjunto de instrucciones SQL que pueden ser ejecutadas con una simple llamada a su nombre. **(Groussard, 2013)**……..**26**

Strategy Pattern
Es un patrón de diseño que ayuda a construir un objeto con una arquitectura que pueda hacer uso de un algoritmo en otro objeto en demanda en lugar de contener la lógica para sí mismo. **38**

Subqueries
Query dentro de un query. ……..**26**

T

Template Pattern
Es un patrón de diseño que crea un objeto abstracto que haga cumplir una serie de métodos y funcionalidades que serán usados en común por varias clases hijas, como una plantilla para su propio diseño. **38**

Trigger
Es el nombre de un conjunto de comandos que se ejecutan antes o después de que se realiza un cambio en una tabla (INSERT, UPDATE y DELETE). **(Ángel Arias, 2015)** …… **26**

U

UML
Es un lenguaje estándar que permite expresar claramente requerimientos, arquitectura y diseños. Fue originalmente creado por Rational Software y ahora es mantenido por la organización de estándares Object Management Group (OMG). 4

UPS
"Uninterruptible power supply" es un dispositivo que gracias a sus baterías u otros elementos almacenadores de energía, puede proporcionar energía eléctrica por un tiempo limitado y durante un apagón eléctrico a todos los dispositivos que tenga conectados. (Wikipedia, s.f.)……**68**

V

Versión Beta: Software técnicamente acabado, lo cual significa que no se le añadirán de momento más funciones, y presumiblemente será lo suficientemente estable para trabajar con normalidad. (Ordenadores y Portátiles, s.f.)……**7**

Visitor Pattern
Es un patrón de diseño que construye distintos objetos que contienen un algoritmo, el cual, al ser consumido por un objeto padre de forma estándar, aplica el algoritmo a dicho objeto padre.....**38**

Vista
Es una consulta sobre unas tablas de base (SELECT) que se almacena en el Catálogo bajo un nombre que puede ser usado como una tabla más. (Enrique Rivero Cornelio, 2004).....**26**

X

XAMPP
Es un servidor independiente de plataforma, software libre, que consiste principalmente en el sistema de gestión de

Z

Bibliografía

Ángel Arias, E. F. (2015). *Curso de Programación con iOS: Apps iPhone.* IT Campus Academy.

Ángel Sastre Castillo, M. Á. (2009). *Diccionario de economía y empresa.* Madrid: Ecobook-Editorial del Economista.

Bassett, L. (2015). *Introduction to JavaScript Object Notation.* Gravenstein: O'Reilly Media, Inc.

Cardador Cabello, A. L. (2014). *Implantación de aplicaciones web en entornos internet, intranet y extranet. IFCD0210.* Antequera, Málaga: IC Editorial.

Carrillo, A. G. (2006). *Abstracción y estructuras de datos en C++.* Madrid: Delta Publicaciones.

Cunningham & Cunningham, I. (16 de Enero de 2001). *Cunningham & Cunningham, Inc.* Obtenido de Model View Controller History: http://c2.com/cgi/wiki?ModelViewControllerHistory

Date, C. J. (2004). *An Introduction To Database Systems.* Pearson Education, Inc.

DB-Engines. (Febrero de 2016). *DB-Engines*. Obtenido de DB-Engines Ranking: http://db-engines.com/en/ranking

DB-Engines. (Febrero de 2016). *DB-Engines*. Obtenido de DB-Engines Ranking - Trend: http://db-engines.com/en/ranking_trend

DB-Engines. (Febrero de 2016). *DB-Engines*. Obtenido de System Properties Comparison: http://db-engines.com/en/system

DuBois, P. (2013). *MySQL - Fifth Edition.* Pearson Education, Inc.

Durán, F. (2007). *Programación orientada a objetos con Java.* Madrid: Paraninfo.

Eguiluz, J. (2008). *Introducción a AJAX.* www.librosweb.es.

Enrique E. Condor Tinoco, I. S. (2014). *Programación Web con CSS, JavaScript, PHP y AJAX.* Iván Soria Solís.

Enrique Rivero Cornelio, C. G. (2004). *Bases de datos relacionales: diseño físico: Orientada al DB2 para z/OS de IBM).* Univ Pontifica Comillas.

Erich Gamma, R. H. (1998). *Elements of Reusable Object-Oriented Software.* Reading, Mass.: Addison Wesley Longman.

Free Software Foundation, Inc. (2 de Enero de 2016). *El Sistema operativo GNU*. Obtenido de ¿Qué es el software libre?: http://www.gnu.org/philosophy/free-sw.es.html

Gallardo, J. A. (2015). *Aplicaciones microinformáticas e Internet para consulta y generación de documentación. IFCT0310.* Antequera, Málaga: IC Editorial.

Garret, J. J. (18 de Febrero de 2005). *Adaptive Path*. Obtenido de Ajax: A New Approach to Web Applications: http://adaptivepath.org/ideas/ajax-new-approach-web-applications/

Gauchat, J. D. (2012). *El gran libro de HTML5, CSS3 y Javascript.* Barcelona: Marcombo.

Groussard, T. (2013). *Visual Basic 2012 (VB.NET): Los fundamentos del lenguaje - Desarrollar con Visual Studio 2012.* Cornellà de Llobregat (Espagne): Editores ENI.

Hudson, P. (2005). *PHP in a Nutshell.* Sebastopol: O'Reilly Media, Inc.

I., F. G. (2004). *Conocimientos y aplicaciones tecnológicas para la dirección comercial.* Pozuelo de Alarcón: ESIC Editorial.

Ian Sommerville, M. I. (2005). *Ingeniería de Software.* Madrid: Pearson Educación.

ITL Education Solutions Limited. (2010). *Introduction To Database Systems.* Pearson Education India.

JSON. (s.f.). *JSON*. Obtenido de Introducing JSON: http://www.json.org/

Kioskea.net. (Junio de 2014). *CCM*. Obtenido de POO - Herencia: http://www.citationmachine.net/items/confirm

Kristi L. Berg, T. S. (2013). History of Databases. *History of Databases*. Minot, Estados Unidos de América.

Lancker, L. V. (2009). *XHTLM y CSS: Los nuevos estándares del código fuente.* Barcelona: Ediciones ENI.

Larsen, R. (22 de Agosto de 2011). *IBM developerWorks*. Obtenido de Una introducción a Ajax: http://www.ibm.com/developerworks/ssa/web/library/wa-aj-ajaxhistory/

López Illescas Diana Cristina, P. A. (2004). *DESARROLLO DE SOFTWARE UTILIZANDO LA METODOLOGÍA RUP (Rational Unified Process) Caso Práctico: "Sistema Escolástico Parametrizable".* Latacunga.

Microsoft. (s.f.). *Microsoft Developer Network*. Obtenido de Replicación de SQL Server: https://msdn.microsoft.com/es-es/library/ms151198(v=sql.120).aspx

Miguel, J. T. (2015). *MF0493_3 - Implantación de aplicaciones web en entorno internet, intranet y extranet.* Ediciones Parainfo.

Ojala, A. (2013). Software-as-a-Service Revenue Models. *IT Professional 15(3)*, 54-59.

Ojeda, J. (12 de 03 de 2015). *Arquitectura de Software en ASP.NET MVC*. Obtenido de La Arquitectura de mis Proyectos MVC: http://msaspnetmvc.blogspot.com/2015/03/la-arquitectura-de-mis-proyectos-mvc.html#more

Open Group Standard. (2011). *Open Group.* Obtenido de http://pubs.opengroup.org/architecture/togaf9-doc/arch/index.html

Ordenadores y Portátiles. (s.f.). *¿Qué es una versión beta?* Obtenido de ¿Qué significa beta en el mundo de la informática?: http://www.ordenadores-y-portatiles.com/beta.html

Pitt, C. (2012). *Pro PHP MVC.* ApressAccess.

Quesada., P. J. (s.f.). *Práctica de Lenguaje de Marcas*. Obtenido de Práctica de Lenguaje de Marcas: http://dis.um.es/~lopezquesada/documentos/IES_1314/LMSGI/curso/xhtml/xhtml7/index.html

Rational - The Software Development Company. (03 de Julio de 2001). *Rational Unified Process - Best Practices for Softwre Development Teams.* Obtenido de IBM: https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

Reenskaug, T. (2003). *The Model-View-Controller (MVC) Its Past and Present.* Oslo: University of Oslo.

Reenskaug, T. (12 de Febrero de 2007). *University Of Oslo.* Obtenido de The original MVC reports - Trygve Reenskaug - Dept. of Informatics - University of Oslo: http://heim.ifi.uio.no/~trygver/2007/MVC_Originals.pdf

Romero, J. L. (2015). *Instalación y configuración del software de servidor Web. IFCT0509.* Antequera, Málaga: IC Editorial.

Tedeschi, N. (2014). *Microsoft Developer Network.* Obtenido de ¿Qué es un Patrón de Diseño?: https://msdn.microsoft.com/es-es/library/bb972240.aspx

The Linux Information Project. (Enero de 2007). *The Linux Information Project.* Obtenido de Open Source Definition: http://www.linfo.org/open_source.html

Tuya, J. (2007). *Técnicas cuantitativas para la gestión en la ingeniería del software.* Oleiros, La Coruña: Netbiblo.

Valimaki, M. (2005). *The rise of Open Source Licensing.* Helsinki: Turre Publishing.

Vineet Kumar Sharma, D. S. (s.f.). Distributed Co-ordinator Model for Optimal Utilization of Software and Piracy Prevention. *International Journal of Computer Science and Security, Volume (3)*, 550-558.

Walther, S. (24 de Agosto de 2008). *Stephen Walther - ASP .Net, Metro, and Html5.* Obtenido de The Evolution of MVC: http://stephenwalther.com/archive/2008/08/24/the-evolution-of-mvc

Webismymind. (s.f.). *Editablegrid Make your HTML tables editable!* Obtenido de Editablegrid Make your HTML tables editable!: http://www.editablegrid.net/en/

Wendy Jaramillo. (2016). *Aplicación de la metodología RUP y el patrón de diseño MVC en la construcción de un sistema de gestión académica para la Unidad Educativa Ángel De La Guarda.* Quito.

Wikipedia. (s.f.). Obtenido de GNU: https://es.wikipedia.org/wiki/GNU

Wikipedia. (s.f.). Obtenido de Drop-down List: https://en.wikipedia.org/wiki/Drop-down_list

Wikipedia. (13 de 04 de 2015). *Wikipedia.* Obtenido de Query String: https://es.wikipedia.org/wiki/Query_string

Wikipedia. (2016). *Wikipedia*. Obtenido de Proceso Unificado de Rational: https://es.wikipedia.org/wiki/Proceso_Unificado_de_Rational

Wikipedia. (s.f.). *Wikipedia*. Obtenido de Ajax (programming): https://en.wikipedia.org/wiki/Ajax_(programming)

Wikipedia. (s.f.). *Wikipedia*. Obtenido de Interfaz de programación de aplicaciones: https://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones

Wikipedia. (s.f.). *Wikipedia*. Obtenido de Polimorfismo (informática): https://es.wikipedia.org/wiki/Polimorfismo_(inform%C3%A1tica)

Wikipedia. (s.f.). *Wikipedia*. Obtenido de PowerBuilder: https://es.wikipedia.org/wiki/PowerBuilder

Wikipedia. (s.f.). *Wikipedia*. Obtenido de Open Database Connectivity: https://es.wikipedia.org/wiki/Open_Database_Connectivity

Wikipedia. (s.f.). *Wikipedia*. Obtenido de Clave foránea: https://es.wikipedia.org/wiki/Clave_for%C3%A1nea

Wikipedia. (s.f.). *Wikipedia*. Obtenido de Sistema de alimentación ininterrumpida: https://es.wikipedia.org/wiki/Sistema_de_alimentaci%C3%B3n_ininterrumpida

Wikipedia. (s.f.). *Wikipedia*. Obtenido de XAMPP: https://es.wikipedia.org/wiki/XAMPP

Wikipedia. (s.f.). *Wikipedia*. Obtenido de Router: https://es.wikipedia.org/wiki/Router

Wikipedia. (s.f.). *Wikipedia*. Obtenido de AOLserver: https://es.wikipedia.org/wiki/AOLserver

Zenón J. Hernández Figueroa, F. J. (2009). *Iniciación a la programación. Ada como primer lenguaje 2da Edición.* Lulu.com.