

Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Centro de Computación Paralela y Distribuida - CCPD

Desarrollo del Módulo de Gestión de Trámites de Profesores del Sistema de Gestión de la Programación Docente para el Departamento de la Escuela de Computación de la Universidad Central de Venezuela

Trabajo Especial de Grado presentado ante la Ilustre
Universidad Central de Venezuela por el Bachiller
Daniel Enrique Hernández Guzmán (C.I. V-20.594.114)
para optar al título de Licenciado en Computación

Tutores: Prof. Carlos Acosta y Profa. Yusneyi Carballo

Caracas, 23 de Octubre de 2015

ACTA

Quienes suscriben, miembros del jurado designado por el Consejo de la Escuela de Computación, para examinar el Trabajo Especial de Grado titulado **"Desarrollo del Módulo de Gestión de Trámites de Profesores del Sistema de Gestión de la Programación Docente para el Departamento de la Escuela de Computación de la Universidad Central de Venezuela"** y presentado por el Bachiller **Daniel Enrique Hernández Guzmán C.I. V-20.594.114**, a los fines de optar al título de **Licenciado en Computación,** dejamos constancia de lo siguiente:

Leído como fue dicho trabajo, por cada uno de los miembros del jurado, se fijó el día Viernes, 23 de Octubre de 2015, a las 3:30 pm horas, para que el autor lo defendiera en forma pública, lo que este hizo en la Sala PA-III de la Escuela de Computación, mediante una presentación oral de su contenido, luego de lo cual respondió a las preguntas formuladas. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobar con la nota de _____ puntos.

En fe de lo cual se levanta la presente Acta, en Caracas el día 23 de Octubre de 2015.

_____	_____
Prof. Carlos Acosta	Profa. Yusneyi Carballo
Tutor	Cotutora
_____	_____
Prof. Robinson Rivas Suárez	Prof. Marcel Castro
Jurado Principal	Jurado Principal

AGRADECIMIENTOS

Primeramente a Dios que me ha acompañado en todo momento, y me ha dado la fuerza, perseverancia y sabiduría necesarias para salir adelante en los momentos de mayor dificultad.

A mis padres y hermano quienes han sido mi fuerza y motivación en todo momento, y me han guiado siempre por el mejor camino además de contar con su apoyo incondicional en las buenas y en las malas.

A mis amigos, compañeros de trabajo y demás familiares que me han demostrado su interés y su apoyo en todo momento.

A mis tutores Yusneyi Carballo y Carlos Acosta, por guiarme de la mejor manera y prestarme toda su colaboración en la elaboración de este proyecto.

A todas las personas mencionadas y faltantes, muchísimas gracias.

Por el Br. Daniel Enrique Hernández Guzmán



Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Centro de Computación Paralela y Distribuida – CCPD

Desarrollo del Módulo de Gestión de Trámites de Profesores del Sistema de Gestión de la Programación Docente para el Departamento de la Escuela de Computación de la Universidad Central de Venezuela

Autor: Hernández, Daniel
C.I: V-20.594.114
danielhg7@gmail.com
Tutores: Prof. Carlos Acosta
cacostal@yahoo.co.uk
Profa. Yusneyi Carballo
yusneyi.carballo@ciens.ucv.ve

Fecha: 23 de Octubre de 2015

RESUMEN

Actualmente, en el Departamento de la Escuela de Computación de la Universidad Central de Venezuela el proceso de gestión de la Programación Docente presenta una desactualización significativa, debido a que no se lleva un control del flujo del proceso, las actividades en su mayoría se realizan de forma manual, y en general presenta una serie de carencias que motivan el desarrollo de este proyecto. Por consiguiente el presente Trabajo Especial de Grado se enfoca en la automatización de los procesos de dicha gestión a través del desarrollo de un sistema compuesto por varios módulos, que permita ofrecer un mejor desempeño y minimizar los costos y recursos para la conformación de la Programación Docente. Para ello se utilizará la metodología Scrum, que permitirá garantizar un proceso de desarrollo adaptable a los cambios, y por otro lado, se hará uso de los marcos de desarrollo (*frameworks*) Laravel (PHP) y AngularJS (JavaScript), SASS para las hojas de estilo, el manejador de versiones Git, Bootstrap para el diseño de las interfaces y PostgreSQL como sistema de gestión de la base de datos.

Palabras Clave: Tecnología, Aplicación Web, Gestión de procesos, Automatización de Procesos, Programación Docente.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En los últimos años se ha podido observar cómo la tecnología se ha convertido en una herramienta principal al momento de llevar a cabo tareas tan simples como ver noticias, comunicarse o difundir información. Actualmente el acceso a una computadora e Internet ofrece un inmenso conjunto de opciones a los usuarios para realizar distintas actividades, facilitar procesos, además de reducir gastos y esfuerzo.

Por ello en la actualidad muchas personas continúan desarrollando aplicaciones y creando herramientas que permitan realizar procesos complejos de una manera más sencilla, buscando minimizar costos y ahorrar recursos. Las universidades no se escapan de esta realidad, las cuales se han visto en la necesidad de crear software que les facilite muchos procedimientos que se realizan normalmente, como consultar información sobre las materias, realizar inscripciones, consultar notas, conocer el grupo docente asignado, entre otras.

El Departamento de la Escuela de Computación de la Facultad de Ciencias de la Universidad Central de Venezuela, como ente velador de una educación de primera calidad, debe mantenerse actualizado para desarrollar un buen desempeño en los servicios que ofrece al personal docente y administrativo, así como a la comunidad estudiantil.

Actualmente los procesos de gestión de la planificación docente del Departamento de Computación presentan una serie de carencias que no permiten que haya un mejor desempeño en su ejecución, destacando entre las limitaciones actuales:

1. No se utiliza la tecnología en un gran porcentaje del proceso. La mayoría de las actividades son realizadas manualmente.

2. No hay organización ni control en el flujo de las actividades.

3. Existen varios factores que pueden generar retrasos en el proceso de planificación.

4. información que no está consolidada en una base de datos.

5. Los profesores y otros participantes del proceso deben ir muchas veces a la Escuela para realizar trámites o asistir a reuniones sobre la gestión de la programación docente.

Por lo tanto, se genera la iniciativa y el interés de optimizar los métodos usados a través del desarrollo de un sistema que mejore el rendimiento de las actividades a realizar.

El presente documento de trabajo de investigación se encuentra estructurado de la siguiente manera:

Capítulo I

Se presenta el contexto del problema de investigación relacionado al actual proceso de conformación de la planificación docente. Puntualmente se describe la situación actual de los procesos del Departamento, planteamiento del problema, justificación, la importancia y se expone, a su vez, el objetivo general y los objetivos específicos del Trabajo Especial de Grado.

Capítulo II

Se presentan los antecedentes que sirven de apoyo al presente trabajo de investigación y el marco conceptual, el cual define los conceptos necesarios para modelar la propuesta de aplicación y las herramientas que se usarán para la implementación del mismo.

Capítulo III

Se muestra una descripción de la metodología de desarrollo de software a implementar.

Capítulo IV

Constituye el marco aplicativo, donde se describe la solución planteada en detalle a través de los objetivos, alcance, descripción de la propuesta y resultados obtenidos para culminar el Trabajo Especial de Grado.

Finalmente se agregan los resultados obtenidos para culminar el Trabajo Especial de Grado así como las conclusiones, recomendaciones, referencias bibliográficas y digitales, y anexos con información adicional.

CAPÍTULO I. PLANTEAMIENTO DEL PROBLEMA

1.1 Problema o necesidad

En los últimos años la automatización de procesos administrativos en las universidades ha tomado un mayor auge, debido a la necesidad de optimizar la gestión de los procesos de la organización, minimizar actividades y el uso de recursos e incrementar la eficiencia en general. Uno de estos procesos es el relacionado a la programación docente la cual involucra la planificación previa de recursos de infraestructura, personal docente y demás recursos con el fin de administrarlos y poder proporcionar al estudiantado una oferta académica adecuada a las necesidades y recursos disponibles, sin menoscabo de la calidad.

Actualmente, en el Departamento de la Escuela de Computación de la Facultad de Ciencias para conformar la programación docente lleva un largo camino dado que dicha gestión de los procesos implica varias actividades que ameritan información sobre profesores, preparadores, aulas, asignaturas obligatorias y electivas, horarios y demás recursos necesarios que no están consolidados en una base de datos, más aún todo esto se lleva en muchos casos de forma manual y en otros se utilizan rudimentarias hojas Excel con alguna información disponible. Adicionalmente, no existe un sistema que permita la gestión, control y seguimiento de todas las actividades involucradas que se han ido realizando y completando y no se tiene información que apoye la toma de decisiones.

Basado en estas premisas, es de gran utilidad el desarrollo de una aplicación web que de alguna forma automatice la gestión y permita ejecutar los procesos referentes a los trámites del personal docente para la conformación de la planificación docente, con el objetivo de implementar una mejora en los procesos de gestión del Departamento.

1.2 Situación actual

Mediante información suministrada por el Profesor Carlos Acosta y la Secretaria del Departamento de Computación Señora Gleidys Caraballo se pudo obtener una descripción del flujo de actividades que siguen los principales procesos y servicios que actualmente presta el Departamento.

Teniendo dicha información se pudo llegar a la conclusión de que el proceso actualmente posee una desactualización significativa debido a que no posee una plataforma que evite la ambigüedad o duplicidad de la información, no favorece el ahorro de recursos, no posee control del flujo de los procesos. En general carece de

automatización, pese a que se manejan hojas de cálculo con la herramienta Microsoft Excel y la comunicación se realiza a través de correo electrónico. Todo esto trae como consecuencia mucha carga laboral, lentitud en el flujo de actividades, falta de coordinación y puede generar problemas de gestión de la información.

Por lo tanto se considera que, contando con una aplicación sobre plataforma web se facilitan las actividades de los actores involucrados, se reducen los tiempos de respuesta, se controlan y coordinan las actividades que se van ejecutando, y se disminuye la inversión económica necesaria para la realización del proceso.

1.3 Objetivos del Trabajo Especial de Grado

1.3.1 Objetivo General

Desarrollar un sistema web de gestión que permita llevar el control y seguimiento del proceso de planificación docente de forma automatizada, optimizando las actividades y el uso de recursos con el fin de hacer más eficiente la gestión de la programación docente.

1.3.2 Objetivos Específicos

1. Hacer uso de una adaptación de la metodología Scrum para equipos de trabajo de una persona, para la creación del Sistema de Gestión de la Programación Docente.

2. Realizar los diagramas de casos de uso y diagramas de secuencia de la aplicación en función de la información de los procesos suministrada en el Departamento.

3. Establecer un diseño lógico y físico de la base de datos, a fin de soportar la información que se desea almacenar.

4. Aplicar estándares, patrones y lineamientos de usabilidad en el diseño de la aplicación, para que este diseño de interfaces sea claro y limpio.

5. Desarrollar funcionalidades para la gestión de la planificación docente.

6. Realizar las pruebas de las funcionalidades desarrolladas y evaluación integral del módulo.

1.4 Alcance y restricciones

El Módulo estará acotado por los trámites que realiza el personal docente para la conformación de la planificación docente del Departamento de la Escuela de Computación. Los servicios que prestará dicha aplicación incluyen las siguientes actividades:

- Proveer una interfaz para que los profesores puedan llenar y enviar sus propuestas para la planificación docente.
- Gestionar las propuestas docentes de los centros de investigación de la Escuela de Computación.
- Gestionar estados de las propuestas enviadas por los profesores o coordinadores.
- Generar archivo con la programación docente propuesta por el Jefe de Departamento de la Escuela.
- Gestionar los procesos de comunicación entre los actores.

CAPÍTULO II. MARCO TEÓRICO

2.1 Antecedentes

Los antecedentes, son todos aquellos trabajos de investigación que preceden al que se realiza, indagaciones previas que sustentan el estudio, tratan sobre el mismo problema o se relacionan de alguna manera con el problema de estudio.

En tal sentido, se destacan los siguientes antecedentes, que son comunes además al Módulo de Gestión de Trámites de Preparadores y Auxiliares Docentes:

- "***Automatización de procesos relacionados con las solicitudes estudiantiles y actividades administrativas y de docencia de la Facultad de Ciencias***" por los licenciados Boyer C. Yurbelis G. y Méndez D. Néstor A. y los tutores Prof. Jossie Zambrano y Prof. Concettina Di Vasta (2008), cuyo objetivo general es: Desarrollar una aplicación Web que automatice el conjunto de procesos relacionados a las diversas solicitudes estudiantiles y actividades que realizan las escuelas y dependencias de la Facultad de Ciencias, en relación a la DCE, dando soporte al personal involucrado aumentando así su satisfacción, y al mismo tiempo contribuir en la disminución de los costos y los tiempos de respuesta.

- "***Prototipo de Sistema de Gestión de Preparadores de la Escuela de Computación, Módulos Gestión de Retiros y Administrativo***" por las licenciadas Aguilar M. Iris H. y Bellet L. Isabelle M. y los tutores Prof. Marcel Castro y Prof. Robinson Rivas (2009), cuyo objetivo general es: Desarrollar los módulos de Gestión de Retiros y Administrativo del sistema de gestión de preparadores de la Escuela de Computación.

- "***Prototipo de Sistema de Gestión de Preparadores de la Escuela de Computación, Módulos Gestión de Concursos y Nombramientos***" por los licenciados Gouveia C. Rosa M. y Ruiz B. Isaura L. y los tutores Prof. Marcel Castro y Prof. Robinson Rivas (2009), cuyo objetivo general es: Desarrollar los módulos de gestión de concursos y nombramientos del sistema de gestión de preparadores de la Escuela de Computación, según el Workflow propuesto para estos procesos.

- "***Desarrollo de Módulos para la Gestión de Trámites Administrativos y Solicitudes de la Escuela de Computación de la Universidad Central de Venezuela***" por las licenciadas Montenegro Z. Hailyx N. y Zapata B. Isthar L. Y. y la tutora Prof. Carballo B. Yusneyi Y. (2014), cuyo objetivo general es:

Desarrollar un sistema compuesto por módulos, para facilitar la realización de tareas de forma automatizada, asociados a la gestión de trámites administrativos y solicitudes docentes.

- "***Automatización de los procesos asociados a la Planificación Docente en CONEST***" por la licenciada Josefina J. Montilla M. y los tutores Prof. Jossie Zambrano y Prof. Sergio Rivas (2014), cuyo objetivo general es: Automatizar los procesos asociados a la Planificación Docente, con la finalidad de generar un impacto positivo en los tiempos de respuesta y reducir la brecha de comunicación entre las dependencias de las escuelas y la DCE (División de Control de Estudios).

2.2 Conceptos básicos y tecnologías de desarrollo

2.2.1 Aplicación web

Una aplicación web es un programa informático que en lugar de ejecutarse de manera local (como las aplicaciones de escritorio), se ejecuta parcialmente en un servidor remoto, al que se accede a través de Internet por medio de un navegador web. (Moreira, 2011)

2.2.2 Lenguaje de Marcado de Hipertexto (HTML)

HTML, *Hyper Text Markup Language* (Lenguaje de marcación de Hipertexto) es un lenguaje de etiquetas que se emplea para dar formato a los documentos que se desean publicar en la web. Los navegadores pueden interpretar las etiquetas y muestran los documentos con el formato deseado.

HTML se basa en el *Standard Generalized Markup Language* (SGML), un sistema mucho más completo y complicado de procesamiento de documentos que indicar como organizar y marcar (etiquetar) un documento. HTML define e interpreta las etiquetas de acuerdo a SGML.

Las páginas HTML se pueden diseñar usando textos con distintos tipos de letras o colores, imágenes, listas de elementos, tablas, entre otros. Su modo de empleo es muy sencillo: se basa en el uso de etiquetas que indican que elementos contiene cada página, el formato que hay que aplicar a cada uno de ellos y como se tiene que distribuir por la página. (Luján, 2001)

2.2.3 Hojas de Estilo en Cascada (CSS)

CSS, *Cascading Style Sheets*, en español Hojas de Estilo en Cascada es una especificación desarrollada por el W3C (World Wide Web Consortium) para separar el estilo de una página web de su contenido. CSS usa reglas que especifican como se deben mostrar las distintas partes de un documento incluyendo elementos como colores, fondos, márgenes, bordes, tipos de letra, entre otros, permitiendo modificar la apariencia de una página web de una forma más sencilla. (Duckett, 2004)

Cada una de estas reglas definidas en lenguaje CSS consta de un selector y una declaración, esta última va entre corchetes y consiste en una propiedad o atributo, y un valor separados por dos puntos.

2.2.4 JavaScript

JavaScript es un lenguaje de programación interpretado, basado en objetos (no es orientado a objetos "puro") y multiplataforma, inventado por NETSCAPE COMMUNICATIONS CORPORATION. Los navegadores Netscape fueron los primeros que usaron JavaScript.

El núcleo de JavaScript se puede ampliar añadiendo objetos adicionales, como por ejemplo:

- JavaScript en el cliente amplia el lenguaje agregando objetos que permiten controlar un navegador y su DOM.
- JavaScript del lado del servidor amplia el lenguaje añadiendo objetos útiles cuando se ejecuta JavaScript en un servidor, como lectura de ficheros, acceso base de datos, entre otros.

Este lenguaje permite crear aplicaciones que se ejecuten a través de Internet, basadas en el paradigma cliente/servidor. La parte del cliente se ejecuta en un navegador web, mientras que la parte del servidor se ejecuta de forma remota en un servidor. (Luján, 2001)

2.2.5 Bootstrap

Es un conjunto de herramientas de software libre para la creación de páginas y aplicaciones web. Contiene plantillas de diseño basadas en HTML y CSS para tipografía, formularios, botones y otros componentes de interfaz, además posee extensiones de JavaScript. (Niska, 2014)

2.2.6 Preprocesador de Hipertexto (PHP)

Es un lenguaje de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían embeber directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página web resultante. (Wikipedia, s. f)

Las principales características de PHP (según la enciclopedia libre Wikipedia) son:

- Orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.
- El código fuente escrito en PHP es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado al navegador, lo que hace que la programación en PHP sea segura y confiable.
- Se puede conectar con una gran cantidad de motores de base de datos que se utilizan en la actualidad, destacando su conectividad con MySQL y PostgreSQL.
- Posee una amplia documentación en su sitio web oficial.
- Es software libre, por lo que es una alternativa de fácil acceso.
- Permite aplicar técnicas de programación orientada a objetos.
- No requiere definición de tipos de variables aunque sus variables pueden ser evaluadas también por el tipo que estén manejando en tiempo de ejecución. (Wikipedia, s. f)

2.2.7 Patrón de Arquitectura Modelo-Vista-Controlador (MVC)

El patrón Modelo-Vista-Controlador se para el modelado del dominio, la presentación, y las acciones basadas en la interacción del usuario en tres clases:

- **Modelo:** El modelo maneja el comportamiento y los datos del dominio de la aplicación, responde a consultas por información sobre su estado (usualmente desde la vista), y responde a instrucciones para cambiar su estado (usualmente desde el controlador).

- **Vista:** La vista maneja la visualización de la información.

- **Controlador:** El controlador interpreta la interacción del usuario, informando al modelo y/o la vista a cambiar según sea apropiado. (Microsoft, s. f)

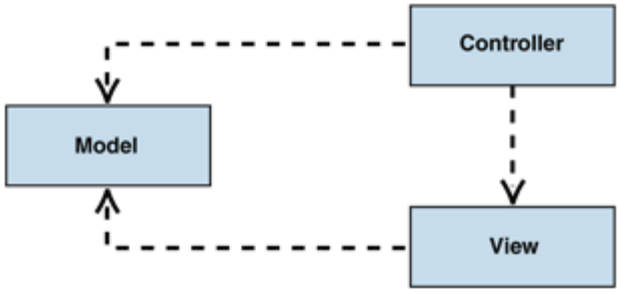

Figura 1: Diagrama del Patrón Modelo-Vista-Controlador
Fuente: (Microsoft, s. f)

2.2.8 Base de Datos

Una base de datos es una serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por los sistemas de información de una empresa o negocio en particular. (Valdés, 2007)

Según Valdés (2007) las bases de datos poseen las siguientes características principales:

- Independencia lógica y física de los datos.
- Redundancia mínima.
- Acceso concurrente por parte de múltiples usuarios.
- Integridad de los datos.
- Consultas complejas optimizadas.

- Seguridad de acceso y auditoría.
- Respaldo y recuperación.
- Acceso a través de lenguajes de programación estándar.

Pero las bases de datos dependen de un software el cual es el que soportará lógicamente esta estructura, a este software se le conoce como Sistema de Gestión de Bases de Datos (SGBD).

Sistemas de Gestión de Base de Datos

Los Sistemas de Gestión de Base de Datos son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. (Valdés, 2007)

El software utilizado como sistema de gestión de base de datos para el desarrollo de este sistema fue PostgreSQL.

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional que ha sido desarrollo en varias formas desde el año 1977. Inició como un proyecto llamado Ingres en la Universidad de California en Berkeley. Más tarde Ingres sería desarrollado comercialmente por la Corporación *Relational Technologies/Ingres*. (Worsley y Drake, 2002)

Este sistema manejador de bases de datos es distribuido bajo la licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales.

PostgreSQL utiliza un modelo cliente/servidor y usa *multiprocesos* en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. (PostgreSQL, 2015)

2.2.9 *Framework*

Según la Wikipedia un *framework* es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un

framework puede incluir soporte de programas, librerías y un lenguaje de *scripting* entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Además Johnson y Boote (1988) lo definen como "un diseño reusable de un sistema (o subsistema)". Está expresado por un conjunto de clases abstractas y el modo en que sus instancias colaboran para un tipo específico de software. Todos los frameworks son diseños orientados a objetos".

Aplicando estos conceptos podemos decir que los frameworks de desarrollo web son estructuras definidas cuyos componentes permiten facilitar el desarrollo de aplicaciones web. Además proveen una capa de abstracción sobre la arquitectura original adaptándola para acelerar los tiempos de desarrollo y mantenimiento.

Laravel

Es un *framework open source* de PHP diseñado para el desarrollo de aplicaciones y servicios web. Laravel es relativamente nuevo ya que su primera versión fue lanzada en abril del 2011, haciendo principal hincapié en la elegancia y simplicidad de su sintaxis.

Este innovador *framework* propone una forma de desarrollar aplicaciones web de un modo mucho más ágil, creando código de forma sencilla y permitiendo multitud de funcionalidades. Intenta aprovechar lo mejor de otros *frameworks* y las características de las últimas versiones de PHP. (Wikipedia, 2015)

Según su sitio web, Laravel posee las siguientes características:

- Es un proyecto liberado bajo la licencia MIT.
- Utilizada un motor de plantillas llamado Blade.
- Usa componentes del *framework* Symfony.
- Utiliza Eloquent, una herramienta que permite que la interacción con la base de datos sea totalmente orientada a objetos.
- Cuenta con una herramienta de línea de comandos llamada Artisan que permite ejecutar tareas programadas como migraciones, pruebas unitarias, entre otras.

- Permite el uso de controladores Restful, los cuales proveen una forma opcional de separar la lógica detrás de las peticiones HTTP POST y GET.

AngularJS

Es un *framework* de JavaScript que facilita el desarrollo de aplicaciones web del lado de cliente. Permite utilizar HTML y extender su sintaxis para expresar los componentes de su aplicación claramente. AngularJS sincroniza automáticamente los datos de la interfaz de usuario con sus objetos JavaScript a través de dos vías de enlace de datos para ayudar a estructurar la aplicación.

Los objetivos de su diseño son los siguientes:

- Disociar la manipulación del DOM de la lógica de la aplicación. Esto mejora la capacidad de prueba del código.

- Considerar a las pruebas de la aplicación como iguales en importancia a la escritura de la aplicación. La dificultad de las pruebas se ve reducida de gran manera por la forma en que el código está estructurado.

- Disociar el lado del cliente de una aplicación del lado del servidor. Esto permite que el trabajo de desarrollo avance en paralelo, y permite la reutilización de ambos lados.

- Guiar a los desarrolladores a través de todo el camino de la construcción de la aplicación, desde el diseño de la interfaz de usuario, a través de la escritura de la lógica del negocio, hasta las pruebas.

2.2.10 Proceso

Según Euskalit (2008) un proceso se define como "Cualquier secuencia repetitiva de actividades que una o varias personas desarrollan para hacer llegar una salida a un destinatario a partir de unos recursos que se utilizan o bien se consumen".

Por otra parte, la norma ISO 9000 (2006) define proceso como "Conjunto de actividades mutuamente relacionadas o que interactúan, las cuales transforman elementos de entrada en resultados".

Basándonos en estos conceptos se puede decir que un proceso es un conjunto de pasos o etapas necesarios para llevar a cabo una actividad o lograr un objetivo.

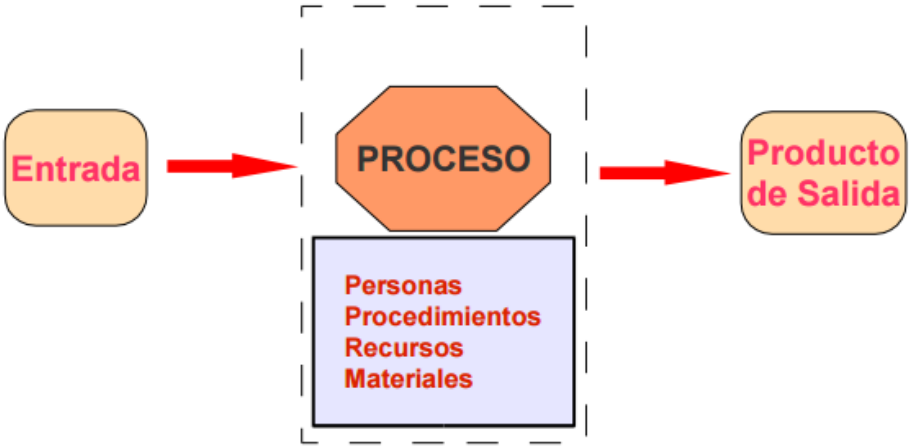

Figura 2: Flujo General de un proceso
Fuente: Flujo (adrformacion, 2008)

2.2.11 Gestión de Procesos de Negocio (BPM)

La Gestión de Procesos de Negocio (BPM por sus siglas en inglés) representa una estrategia para administrar y mejorar el desempeño de los negocios al optimizar continuamente los procesos de negocio en un ciclo cerrado de modelado, ejecución y evaluación. Combinando una metodología de mejores prácticas con una solución de tecnología integrada, BPM ha surgido gracias a la evolución de procesos de negocios y de la convergencia de una cantidad de tendencias de tecnología. El resultado es una categoría de soluciones de tecnología basadas en un conjunto de actividades relacionadas y estructuradas, que combina una variedad de funciones y características para satisfacer un ciclo de vida impulsado por los objetivos de la organización. Al fusionar estas tecnologías y funciones en un entorno de diseño integrado y sin defectos, BPM brinda a los especialistas en negocio y tecnología un lenguaje común para alcanzar sus objetivos individuales y compartidos, lo cual implica lograr que la organización sea más sólida y rentable. (Oracle, 2008)

2.2.12 Automatización de Procesos

La automatización busca que el funcionamiento de los procesos se haga de manera fácil y eficiente. La idea es implantar una tecnología para la realización de actividades en un orden específico y lograr tener un seguimiento de los procesos.

2.2.13 Arquitectura Cliente-Servidor

La arquitectura cliente-servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma. Una arquitectura es un conjunto de reglas, definiciones, términos y modelos que se emplean para producir un producto. (Wikipedia, s. f)

2.2.14 Control de versiones

Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Una versión, revisión o edición de un producto, es el estado en el que se encuentra el mismo en un momento dado de su desarrollo o modificación.

Aunque un sistema de control de versiones puede realizarse de forma manual, es muy aconsejable disponer de herramientas que faciliten esta gestión dando lugar a los llamados sistemas de control de versiones o VCS (del inglés *Version Control System*). Estos sistemas facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas. (Wikipedia, s. f)

Para el desarrollo de este proyecto se usó Git como sistema de control de versiones.

Git

Es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. (Wikipedia, s. f)

CAPÍTULO III. MARCO METODOLÓGICO

3.1 Metodologías de desarrollo de software

Una metodología de desarrollo de software se define como un proceso para producir software de forma organizada, empleando una colección de técnicas y convenciones de notación predefinidas.

Según Palvia y Nosek (1993) una metodología de desarrollo es una aproximación organizada y sistemática para el ciclo de vida del sistema o sus partes, especificando las tareas individuales y sus secuencias.

Por otro lado, Sommerville (2002) define que "un método de ingeniería de software es un enfoque estructurado para el desarrollo de software cuyo propósito es facilitar la producción de software de alta calidad de una forma costeable".

Particularmente, una metodología es un conjunto de componentes que especifican: Cómo dividir un proyecto en etapas; Qué tareas se llevarán a cabo en cada etapa; Qué salidas se producen y cuándo deben producirse; Qué restricciones se aplican; Qué herramientas van a ser utilizadas; Cómo se gestiona y se controla el proyecto; entre otras interrogantes. Adicionalmente, una metodología debería definir con precisión los artefactos, roles y actividades, junto con prácticas, técnicas recomendadas y guías de adaptación de la metodología al proyecto. Sin embargo, la complejidad del proceso de creación de software es netamente dependiente de la naturaleza del proyecto mismo, por lo que el escogimiento de la metodología estará acorde al nivel de aporte del proyecto, ya sea de pequeño, mediano o gran nivel.

3.1.1 Objetivos de una metodología

- Establecer los requisitos de un sistema software de una forma acertada.
- Proporcionar un método sistemático de desarrollo de forma que se pueda controlar su proceso.
- Construir un sistema software dentro de un tiempo apropiado y unos costes aceptables.
- Construir un sistema que esté bien documentado y que sea fácil de mantener.
- Ayudar a identificar, lo antes posible, cualquier cambio que sea necesario realizar dentro del proceso de desarrollo.

- Proporcionar un sistema que satisfaga a todas las personas afectadas por el mismo.

3.1.2 Características deseables en una metodología

Según Henderson-Sellers y Firesmith (1999) una metodología debe cubrir los siguientes aspectos:

- Un proceso de ciclo de vida completo, que comprenda aspectos tanto del negocio como técnicos.
- Un conjunto completo de conceptos y modelos que sean internamente consistentes.
- Una colección de reglas y guías.
- Una descripción completa de artefactos a desarrollar.
- Una notación con la que trabajar, idealmente soportada por diversas herramientas CASE y diseñada para una usabilidad óptima.
- Un conjunto de técnicas probadas.
- Un conjunto de métricas, junto con asesoramiento sobre calidad, estándares y estrategias de prueba.
- Identificación de los roles organizacionales.
- Guías para la gestión de proyectos y aseguramiento de la calidad.
- Asesoramiento para la gestión de bibliotecas y reutilización.

3.2 Ciclo de vida del desarrollo de software

Es la forma mediante la cual se describen los diferentes pasos que se deben seguir para el desarrollo de un software, partiendo desde una necesidad hasta llegar a la puesta en marcha de una solución y su apropiado mantenimiento. El ciclo de vida para un software comienza cuando se tiene la necesidad de resolver un problema, y termina cuando el programa que se desarrolló para cumplir con los requerimientos, deja de ser utilizado.

Entre las funciones que debe tener un ciclo de vida se pueden destacar las siguientes:

- Determinar el orden de las fases del proceso de software.
- Establecer los criterios de transición para pasar de una fase a la siguiente.
- Definir las entradas y salidas de cada fase.
- Describir los estados por los que pasa el producto.
- Describir las actividades a realizar para transformar el producto.
- Definir un esquema que sirve como base para planificar, organizar, coordinar, desarrollar, entre otros.

El ciclo de vida clásico del software tal como lo plantean diferentes autores contiene las siguientes etapas:

- **Análisis:** Es el proceso de investigar un problema que se quiere resolver. Sus dos principales objetivos son definir claramente el problema que se desea resolver o el sistema que se desea crear e identificar los componentes principales que integrarán el producto.
- **Diseño:** Esta etapa consiste en utilizar la información recolectada en la etapa de análisis al diseño del producto. La principal tarea en esta etapa es desarrollar un modelo o las especificaciones para el producto o componentes del sistema.
- **Desarrollo:** Consiste en utilizar los modelos creados durante la etapa de diseño para crear los componentes del sistema.
- **Prueba:** Consiste en asegurar que los componentes individuales que integran al sistema o producto, cumplen con los requerimientos de la especificación creada durante la etapa de diseño.
- **Implantación:** Consiste en poner a disposición del cliente el producto.
- **Mantenimiento y evolución:** El objetivo de esta etapa es corregir problemas del producto y liberarlo como una nueva versión o revisión del mismo.

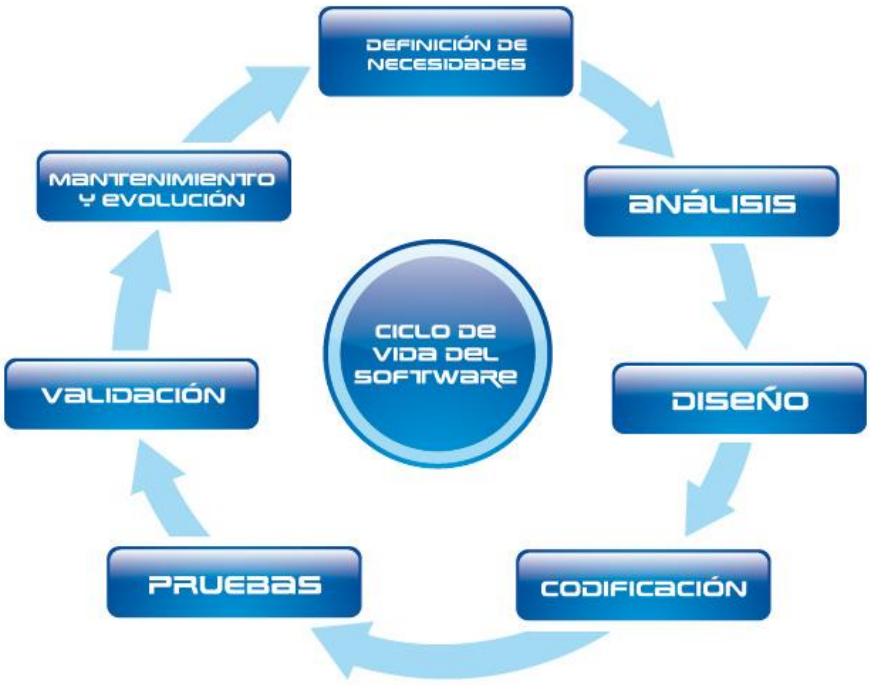

Figura 3: Ciclo de Vida de un Software
Fuente: (comusoft, 2011)

3.3 Metodologías Tradicionales

Las metodologías tradicionales son un conjunto de métodos de ingeniería del software que se enfocan en documentación, planificación y procesos.

Las metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un software más eficiente. Para ello, se hace énfasis en la planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del producto software. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada.

Entre las metodologías tradicionales más conocidas tenemos RUP y MSF, estas metodologías tradicionales centran su atención en llevar una documentación precisa y exhaustiva de todo el proyecto y enfocan su atención en realizar y cumplir un plan de proyecto.

3.3.1 RUP (*Rational Unified Process*)

Es un proceso de ingeniería de software que suministra un enfoque para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de software de alta calidad que satisfaga la necesidad del usuario final dentro de un tiempo y presupuesto previsible. Es una metodología de desarrollo iterativo enfocada hacia los casos de uso, manejo de riesgos y el manejo de la arquitectura. El RUP mejora la productividad del equipo ya que permite que cada miembro del equipo sin importar su responsabilidad específica acceda a la misma base de datos de conocimiento. Esto hace que todos compartan el mismo lenguaje, la misma visión y el mismo proceso acerca de cómo desarrollar software. (Academia, s. f)

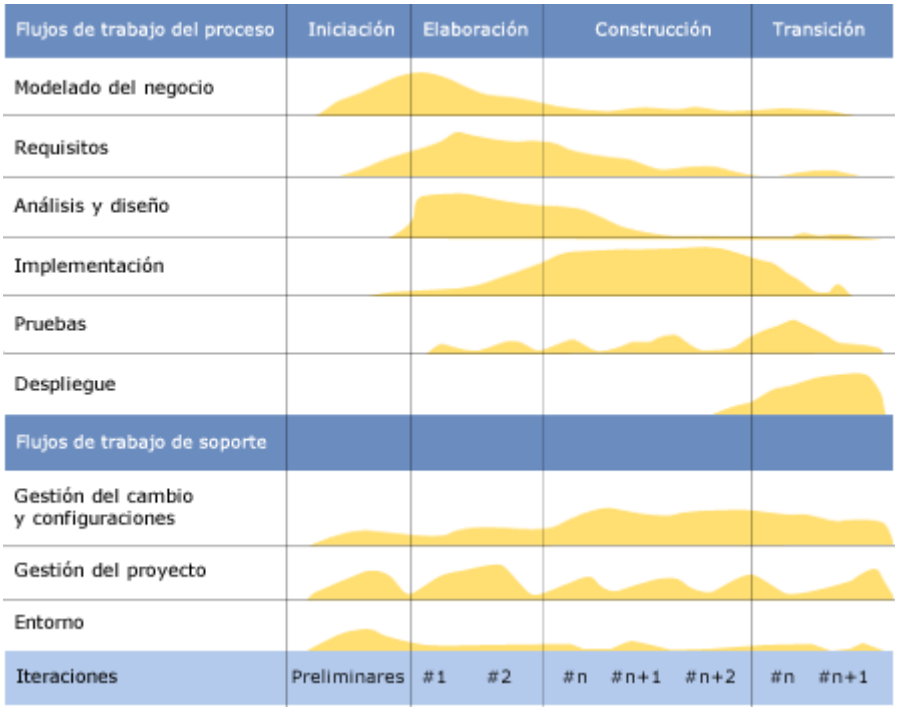

Figura 4: Proceso RUP
Fuente: (Wikipedia, 2008)

3.3.2 MSF (*Microsoft Solution Framework*)

Este modelo se basa principalmente en los modelos espiral y cascada. Fue desarrollado con el objetivo de crear un modelo estructurado basado en una estructura de trabajo en desarrollo de software.

Tiene como principios fundamentales la comunicación entre cliente/usuario y los desarrolladores, una capacitación de las personas (disciplina de disponibilidad) es decir, cumple con el proceso de formación de personal y compartir los roles entre todo el equipo de trabajo.

Figura 5: Proceso MSF
Fuente: (Microsoft, 2006)

3.4 Metodologías Ágiles

En febrero de 2001, tras una reunión celebrada en Utah-EEUU, nace el término "ágil" aplicado al desarrollo de software. En esta reunión participan un grupo de 17 expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales,

caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas. Tras esta reunión se creó *The Agile Alliance*, una organización, sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida fue el Manifiesto Ágil, un documento que resume la filosofía "ágil".

Manifiesto Ágil

Según el Manifiesto se valora:

- **Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas**. La gente es el principal factor de éxito de un proyecto software. Es más importante construir un buen equipo que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte automáticamente. Es mejor crear el equipo y que éste configure su propio entorno de desarrollo en base a sus necesidades.

- **Desarrollar software que funciona más que conseguir una buena documentación**. La regla a seguir es "no producir documentos a menos que sean necesarios de forma inmediata para tomar un decisión importante". Estos documentos deben ser cortos y centrarse en lo fundamental.

- **La colaboración con el cliente más que la negociación de un contrato**. Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.

- **Responder a los cambios más que seguir estrictamente un plan**. La habilidad de responder a los cambios que puedan surgir a los largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta.

Estos valores inspiran los doce principios del manifiesto. Son características que diferencian un proceso ágil de uno tradicional. Los dos primeros principios son generales y resumen gran parte del espíritu ágil. El resto tienen que ver con el proceso a seguir y con el equipo de desarrollo, en cuanto metas a seguir y organización del mismo. Los principios son:

1. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.
2. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.
3. Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.
4. La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.
5. Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.
6. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
7. El software que funciona es la medida principal de progreso.
8. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.
9. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.
10. La simplicidad es esencial.
11. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.
12. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento. (Canós, Letelier y Penadés, 2003)

Métodos ágiles	Métodos tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas.
Es bastante flexible.	Poco flexible.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos artefactos.	Más artefactos.
Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.

Tabla 1 – Comparación entre metodologías ágiles y metodologías tradicionales
Fuente: (Canós, Letelier y Penadés, 2003)

3.4.1 ¿Por qué usar Metodologías Ágiles?

Tomando las ideas de la Tabla Nº 1 podemos decir que las metodologías tradicionales presentan los siguientes problemas a la hora de abordar proyectos:

- Existen unas costosas fases previas de especificación de requisitos, análisis y diseño. La corrección durante el desarrollo de errores introducidos en estas fases será costosa, es decir, se pierde flexibilidad ante los cambios.
- El proceso de desarrollo está encorsetado por documentos firmados.
- El desarrollo es más lento. Es difícil para los desarrolladores entender un sistema complejo en su globalidad.

Las metodologías ágiles de desarrollo están especialmente indicadas en proyectos con requisitos poco definidos o cambiantes. Estas metodologías se aplican bien en equipos pequeños que resuelven problemas concretos, lo que no está reñido con su aplicación en el desarrollo de grandes sistemas, ya que una correcta modularidad de los mismos es fundamental para su exitosa implantación. Dividir el trabajo en módulos abordables minimiza los fallos y el coste.

Las metodologías ágiles presentan diversas ventajas, entre las que podemos destacar:

- Capacidad de respuesta a cambios de requisitos a lo largo del desarrollo.
- Entrega continua y en plazos breves de software funcional.
- Trabajo conjunto entre el cliente y el equipo de desarrollo.
- Importancia de la simplicidad, eliminado el trabajo innecesario.
- Atención continua a la excelencia técnica y al buen diseño.
- Mejora continua de los procesos y el equipo de desarrollo.

3.5 Metodología de desarrollo seleccionada

La metodología elegida como base para el desarrollo de este proyecto fue Scrum. Scrum es una metodología ágil de gestión de proyectos cuyo objetivo primordial es elevar al máximo la productividad de un equipo. Reduce al máximo la burocracia y actividades no orientadas a producir software que funcione y produce resultados en periodos muy breves de tiempo. Como método, Scrum enfatiza valores y prácticas de gestión, sin pronunciarse sobre requerimientos, prácticas de desarrollo, implementación y demás cuestiones técnicas. Más bien delega completamente en el equipo la responsabilidad de decidir la mejor manera de trabajar para ser lo más productivos posibles.

La palabra Scrum procede de la terminología del juego de rugby, donde designa al acto de preparar el avance del equipo en unidad pasando la pelota a uno y otro jugador. Igual que el juego, Scrum es adaptable, ágil, auto-organizante y con pocos tiempos muertos.

Scrum fue desarrollado por Jeff Sutherland y elaborado más formalmente por Ken Schwaber. Poco después Sutherland y Schwaber se unieron para refinar y extender Scrum. Se la ha llegado a conocer como una herramienta de hiperproductividad. Schwaber se dio cuenta entonces de que un proceso necesita aceptar el cambio, en lugar de esperar predictibilidad. Se enfoca en el hecho de que procesos definidos y repetibles sólo funcionan para atacar problemas definidos y repetibles con gente definida y repetible en ambientes definidos y repetibles. Toma el cambio como una forma de entregar al final del desarrollo algo más cercano a la verdadera necesidad del Cliente. Puede ser aplicado teóricamente a cualquier contexto en donde un grupo de gente necesita trabajar junta para lograr una meta común.

Se basa en los principios ágiles:

- Privilegiar el valor de la gente sobre el valor de los procesos.
- Entregar software funcional lo más pronto posible.
- Predisposición y respuesta al cambio.
- Fortalecer la comunicación y la colaboración.

- Comunicación verbal directa entre los implicados en el proyecto.
- Simplicidad; supresión de artefactos innecesarios en la gestión del proyecto. (Citón, 2006)

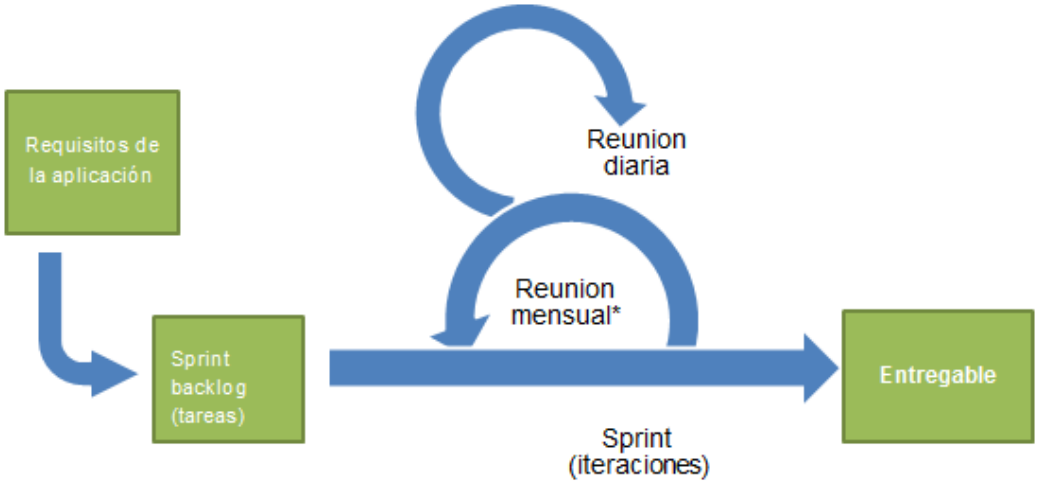

Figura 6: Metodología Scrum
Fuente: (Wikipedia, 2013)

3.5.1 Características de Scrum

Scrum es un modelo de referencia que define un conjunto de prácticas y roles, y que puede tomarse como punto de partida para definir el proceso de desarrollo que se ejecutará durante un proyecto. Los roles principales en Scrum son el *Scrum Master*, que mantiene los procesos y trabaja de forma similar al director de proyecto, el Dueño del Producto (*Product Owner)*, que representa a los clientes externos o internos, y el Equipo que incluye a los desarrolladores. Durante cada iteración (*sprint)*, un periodo entre 15 y 30 días (la magnitud es definida por el equipo), el equipo crea un incremento de software potencialmente entregable.

El conjunto de características que forma parte de cada *sprint* viene del *Product Backlog*, que es un conjunto de requisitos de alto nivel priorizados que definen el trabajo a realizar. Los elementos del *Product Backlog* que forman parte del sprint se determinan durante la reunión de *Sprint Planning*. Durante esta reunión, el *Product Owner* identifica los elementos del *Product Backlog* que quiere ver completados y los hace del conocimiento del equipo. Entonces, el equipo determina la cantidad de ese trabajo que puede comprometerse a completar durante la siguiente iteración. Durante el *sprint*, nadie puede cambiar el *Sprint Backlog*, lo que significa que los requisitos están congelados durante la iteración.

Scrum permite la creación de equipos auto-organizados impulsando la co-localización de todos los miembros del equipo, y la comunicación verbal entre todos los miembros y disciplinas involucrados en el proyecto. Un principio clave de Scrum es el reconocimiento de que durante un proyecto los clientes pueden cambiar de idea sobre lo que quieren y necesitan, y que los desafíos impredecibles no pueden ser fácilmente enfrentados de una forma predictiva y planificada.

Por lo tanto, Scrum adopta una aproximación pragmática, aceptando que el problema no puede ser completamente entendido o definido, y centrándose en maximizar la capacidad del equipo de entregar rápidamente y responder a requisitos emergentes. (Ecured, s. f)

3.5.2 Elementos de Scrum

- Roles
  - Product Owner
  - Scrum Master
  - Equipo
- Poda de Requerimientos
- Product Backlog
- Sprint
  - Planificación
  - Sprint Backlog
  - Scrum
  - Estimaciones
  - Builds continuos
  - Revision del Sprint
  - Reunion retrospectiva
- Valores
  - Foco, comunicación, respeto y coraje

Roles

La dimensión del equipo total de Scrum no debería ser superior a veinte personas. Si hay más, lo más recomendable es formar varios equipos. No hay una técnica oficial para coordinar equipos múltiples, pero se han documentado experiencias de hasta 800 miembros, divididos en Scrums de Scrum,

definiendo un equipo central que se encarga de la coordinación, las pruebas cruzadas y la rotación de los miembros. Scrum tiene una estructura muy simple. Todas las responsabilidades del proyecto se reparten en 3 roles:

- **Dueño del Producto (*Producto Owner*)**

Representa a todos los interesados en el producto final. Sus áreas de responsabilidad son:

- Financiación del proyecto
- Requisitos del sistema
- Retorno de la inversión del proyecto
- Lanzamiento del proyecto

Es el responsable oficial del proyecto, gestión, control y visibilidad de la lista de acumulación o lista de retraso del producto. Toma las decisiones finales de las tareas asignadas al registro y convierte sus elementos en rasgos a desarrollar.

- **Líder de Proyecto (*Scrum Master*)**

Responsable del proceso Scrum, de cumplir la meta y resolver los problemas. Así como también, de asegurarse que el proyecto se lleve a cabo de acuerdo con las prácticas, valores y reglas de Scrum y que progrese según lo previsto.

Interactúa con el cliente y el equipo. Coordina los encuentros diarios, y se encarga de eliminar eventuales obstáculos. Debe ser miembro del equipo y trabajar a la par.

- **Equipo**

Responsable de transformar el *Backlog* de la iteración en un incremento de la funcionalidad del software. Tiene autoridad para reorganizarse y definir las acciones necesarias o sugerir remoción de impedimentos.

- Auto-gestionado
- Auto-organizado
- Multi-funcional

La dimensión del equipo total de Scrum no debería ser superior a veinte. El número ideal es diez, más o menos dos. Si hay más, lo más recomendable es formar varios equipos. No hay una técnica oficial para

coordinar equipos múltiples, pero se han documentado experiencias de hasta 800 miembros, divididos en Scrums de Scrums, definiendo un equipo central que se encarga de la coordinación, las pruebas cruzadas y la rotación de los miembros.

Poda de Requerimientos

La primera actividad es armar una lista exhaustiva de los requerimientos originales del sistema. Luego se procede a ver qué requerimientos son realmente necesarios, cuáles pueden posponerse y cuáles eliminarse. Para ello debe identificarse un representante con capacidad de decisión, priorizar los requerimientos en base a su importancia y acordar cuáles son los prioritarios para la fecha de entrega.

La poda de requerimientos es una buena práctica implícita en modelos ágiles, se hace lo que el cliente realmente desea, no más.

Pila del Producto (*Product Backlog*)

Con los requerimientos priorizados y podados armamos la Pila del Producto. Este es una forma de registrar y organizar el trabajo pendiente para el producto (actividades y requerimientos).

Es un documento dinámico que incorpora constantemente las necesidades del sistema. Por lo tanto, nunca llega a ser una lista completa y definitiva. Se mantiene durante todo el ciclo de vida (hasta la retirada del sistema) y es responsabilidad del *Product Owner*.

Sprint

Scrum está basado en el control empírico de procesos. Se utiliza cuando la capacidad de predicción es vaga, la incertidumbre alta o el proceso es demasiado complejo para ser modelado y definido.

En el enfoque empírico de control de procesos se establecen reglas simples y se crea una disciplina de inspección frecuente para adaptarse rápidamente a situaciones imprevistas o problemas.

Un *Sprint* es el periodo de tiempo durante el que se desarrolla un incremento de funcionalidad. Constituye el núcleo de Scrum, que divide de esta forma el desarrollo de un proyecto en un conjunto de pequeñas “carreras”.

• Duración máxima del *Sprint*: 30 días.

• Durante el *Sprint* no se puede modificar el trabajo que se ha acordado en el *Backlog*.

• Sólo es posible cambiar el curso de un *Sprint*, abortándolo, y sólo lo puede hacer el *Scrum Master* si decide que no es viable por alguna de las razones siguientes:

- La tecnología acordada no funciona.
- Las circunstancias del negocio han cambiado.
- El equipo ha tenido interferencias.

- **Planificación**

Se planifica en detalle el trabajo al inicio de cada *Sprint* asumiendo que los objetivos no van a cambiar durante el mismo. De esta manera se atenúa el riesgo.

Aspectos a tener en cuenta sobre la planificación de un *Sprint*:

• Una determinación general de alcance, frecuentemente basada en una EDT (Estructura de División del Trabajo).

• Estimaciones de esfuerzo de alto nivel realizadas durante la etapa de concepción del proyecto.

• Esfuerzo dedicado a labores de soporte o de preparación de los ambientes requeridos por el proyecto.

• Esfuerzo asociados a las reuniones diarias, de planificación y de revisión.

• Requerimientos de recursos de infraestructura o logísticos (máquinas, redes, licencias, papel, pizarras, etc.).

• Habilidades presentes y necesarias en el equipo.

• Restricciones asociadas al conocimiento del negocio, la tecnología o externas (legales, reglamentarias, estándares, etc.).

Rol del *Scrum Master* durante la planificación:

• Dirige la planificación.

• Es vínculo entre el equipo y el *Product Owner* del proyecto.

• Registra problemas y riesgos detectados durante la planificación.

• Registra las tareas, asignaciones y estimaciones.

• Inicia el *Backlog* del *Sprint*.

- ***Sprint Backlog***

Trabajo o tareas determinadas por el equipo para realizar en un *Sprint*.

• Tareas a convertir en producto funcional.

• Las tareas se estiman en una duración entre 1 a 20 horas de trabajo. Las de mayor duración deben intentar descomponerse en sub-tareas de ese rango de tiempo.

• La estimación se actualiza día a día.

- **Scrum diario**

Scrum asume que el proceso es complejo y que es necesario inspeccionarlo frecuentemente, por eso se realiza una reunión diaria de seguimiento.

El foco de la reunión es determinar el avance en las tareas y detectar problemas o "bloqueos" que estén haciendo lento el progreso del equipo o que eventualmente impidan a un equipo cumplir con la meta del *Sprint*. La idea es que ningún problema quede sin resolver o, por lo menos, sin iniciar alguna acción de respuesta dentro de las 24 horas después de su detección.

La reunión es además un espacio definido para que cada miembro del equipo comunique a los demás el estado de su trabajo y por lo tanto reafirme el compromiso.

Tips para un buen *Scrum* diario:

• Todos los días en el mismo sitio y a la misma hora. Con una duración máxima de 15 minutos.

• Se recomienda que sea la primera actividad del día.

• Deben acudir todos los miembros del equipo.

• Hacer un círculo, permitir que todos vean a todos.

• Mantener el foco.

• Solo habla una persona a la vez y los demás escuchan. No se permiten interrupciones, entrar y salir, hablar por teléfono, etc.

• Sólo se habla de: ¿En que trabajé ayer? ¿En qué voy a trabajar hoy? ¿Qué problemas impiden mi progreso?

• No iniciar discusiones muy largas sobre temas técnicos o sobre los problemas, esto hay que registrarlo y manejarlo después de la reunión.

• Si están permitidas las preguntas para aclarar el alcance de un trabajo o de un problema.

• Cuando un miembro informa de algo de interés para otros, o necesita ayuda de otros, estos se reúnen al terminar la revisión diaria.

Rol del *Scrum Master* durante el *Scrum*:

• Dirige la reunión y mantiene el foco.

• Hace preguntas para aclarar dudas.

• Registra (escribe o documenta) los problemas para su resolución después de la reunión.

• Se asegura que los miembros cuenten con el ambiente adecuado para la reunión.

- **Estimaciones**

Las estimaciones se realizan por primera vez en la reunión de planificación al inicio del *Sprint*. Luego las tareas se re-estiman todos los días y se registran sus cambios en el *Backlog* del *Sprint*; esta actividad puede ser realizada inmediatamente antes o después del *Scrum* diario.

Algunas claves para la estimación:

• Siempre se realizan estimaciones de esfuerzo, no de duración.

• Siempre se estima el esfuerzo total pendiente para terminar la tarea, no se estima el esfuerzo consumido.

• Se buscan unidades manejables, lo usual es que estén en un mínimo de 2 horas y un máximo de 20. Si la tarea es muy corta se trata de juntarla con otras relacionadas. Si la tarea es muy grande se trata de descomponerla.

- **Builds continuos y pruebas básicas**

La estrategia que generalmente se utiliza es la de Builds continuos y "*smoke test*" (prueba básica para la funcionalidad del sistema).

Procedimiento de Builds continuos:

• Los programadores desarrollan según el Backlog del Sprint, y al finalizar, notifican al integrador.

• El integrador toma el código y lo integra con el resto del producto.

• Se compila el software y se prueba el producto, para verificar que no se haya roto.

• Si se encuentran problemas se devuelve al desarrollador.

• Se notifica al equipo que hay una nueva versión "estable" del código para usar como base.

- **Revisión del *Sprint***

El objetivo de la reunión de revisión es presentar el producto o porción del producto desarrollada por el equipo a los usuarios. La reunión se utiliza para detectar inconformidades mayores que se vuelven elementos del *Backlog* de Producto y que eventualmente se resuelven en el siguiente *Sprint*.

A la reunión asisten el equipo, el *Scrum Master*, el *Product Owner* y todas las personas implicadas en el proyecto.

Tips para una buena revisión:

• Duración máxima de la reunión: 4 horas.

• Preparar la presentación, teniendo en cuenta que lo único que se presenta es el producto.

• Mantener el foco durante la reunión de revisión, el foco es el producto solamente.

• Finalidad: presentar al propietario del producto y a las demás personas implicadas en el proyecto las nuevas funcionalidades implementadas.

• Las funcionalidades no implementadas no se presentan.

• Registrar todos los comentarios e inconformidades declaradas de los usuarios sobre el producto para determinar cuáles de ellas formarán parte del *Backlog*.

- **Reunión Retrospectiva**

Scrum involucra el concepto de mejora continua a través de las reuniones de retrospección. Las reuniones buscan detectar los puntos positivos y negativos del *Sprint* para generar propuestas de mejora para futuros *Sprints*.

Las reuniones de retrospección son el concentrador del aprendizaje organizacional sobre el *Scrum*. Los puntos positivos y negativos se registran y se definen ítems de acción para cada uno. Los ítems de acción definidos se toman en cuenta en los siguientes *Sprints*.

Acuden el equipo y el *Scrum Master*, y opcionalmente el Dueño del Producto.

Tips para una buena retrospección:

• Todos los miembros del equipo responden a dos preguntas:

- ¿Qué cosas fueron bien en el último *Sprint*?

- ¿Qué cosas se podrían mejorar?

• El *Scrum Master* anota todas las respuestas.

• El equipo prioriza las mejoras posibles.

• El *Scrum Master* no proporciona respuestas, sino que ayuda al equipo a encontrar la mejor forma de trabajar con Scrum.

• Las acciones de mejora localizadas que se puedan implementar en el próximo *Sprint* deben introducirse en el *Backlog* como elementos no funcionales.

Valores

- **Foco:** Los individuos y sus interacciones son más importantes que el proceso y las herramientas. La gente es el principal factor de éxito de un proyecto de software.

- **Comunicación:** Scrum pone en comunicación directa y continua a clientes y desarrolladores. El cliente se integra en el equipo para establecer prioridades y resolver dudas. De esta forma ve el avance día a día, y es posible ajustar la agenda y las funcionalidades de forma consecuente.

- **Respeto:** Scrum diferencia claramente entre dos grupos para garantizar que quienes tienen la responsabilidad tienen también la autoridad necesaria para poder lograr el éxito, y que quienes no tienen la responsabilidad, los observadores externos, no produzcan interferencias innecesarias.

- **Coraje:** El coraje implica saber tomar decisiones difíciles. Reparar un error cuando se detecta. Mejorar el código siempre que el *feedback* y las sucesivas iteraciones se manifiesten susceptibles de mejora. (Citón, 2006)

3.5.3 Ventajas del uso de Scrum

En general, el desarrollo de aplicaciones web se puede ver favorecido frecuentemente por la utilización de procesos ágiles. La necesidad del cliente que contrata un desarrollo web es que su producto esté disponible en la red lo más pronto posible. Si no se tiene en cuenta esta necesidad, la aplicación no resultará un producto provechoso para el cliente. Puesto que los procesos ágiles permiten tener versiones de producto previas a la versión final, si se aplican correctamente estos procesos el cliente podrá disponer de forma rápida de alguna versión intermedia. Además el ciclo de desarrollo de la mayoría de los sitios y aplicaciones web es extremadamente corto.

Por otra parte, los desarrollos web se perciben como sencillos y los desarrolladores son sometidos a una gran presión de trabajo para terminar lo más pronto posible. Esta forma de trabajar implica, sin duda alguna, modificaciones. Luego, sería conveniente garantizar un proceso de desarrollo adaptable a los cambios. Otra

cuestión fundamental a tener en cuenta es que las aplicaciones web se desarrollan sin conocer los perfiles de los usuarios finales de las mismas, es decir, sin conocer los requisitos de usuario del sistema. Esto provocará cambios en los requisitos inicialmente detectados, lo que lleva de nuevo a la elección de un proceso adaptativo.

Scrum es un proceso en el que se aplican de manera regular un conjunto de prácticas para trabajar en equipo y obtener el resultado eficiente de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

En esta metodología se realizan entregas parciales del resultado final del proyecto, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad y la productividad son fundamentales.

Aunque Scrum surgió como modelo para el desarrollo de productos tecnológicos, también se emplea en entornos que trabajan con requisitos inestables y que requieren rapidez y flexibilidad; situaciones frecuentes en el desarrollo de determinados sistemas de software, y donde al aplicarlo proporciona ventajas como:

- Entrega de un producto funcional al finalizar cada iteración.
- Posibilidad de ajustar la funcionalidad con base en la necesidad de negocio del cliente.
- Visualización del proyecto día a día.
- Alcance acotado y viable.
- Equipos integrados y comprometidos con el proyecto, dado que ellos definieron el proyecto.

Entre los principales beneficios que reporta utilizar Scrum como metodología de desarrollo de un sistema se encuentran además:

- Entrega mensual (o quincenal) de resultados (los requisitos más prioritarios en ese momento, ya completados).
- Mitigación de riesgos. Desde la primera iteración el equipo tiene que gestionar los problemas que pueden aparecer en una entrega del proyecto.

- Productividad y calidad. De manera regular el equipo va mejorando y simplificando su forma de trabajar.
- Alineamiento entre el Cliente y el Equipo de Desarrollo. Los resultados y esfuerzos del proyecto se miden en forma de objetivos y requisitos entregados al negocio.
- Equipo motivado. Las personas se sienten más satisfechas cuando pueden mostrar los logros que consiguen.

Scrum también se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, las entregas se alargan demasiado, los costes se disparan o la calidad no es aceptable, cuando es necesario identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto. (Acuña, 2006)

CAPÍTULO IV. DESARROLLO DE LA SOLUCIÓN

En este capítulo se presentan los objetivos de la aplicación, la lista de requerimientos, alcance, herramientas tecnológicas utilizadas en su desarrollo, principales funcionalidades del Módulo, implementación del método de desarrollo seleccionado, diagramas de secuencias y de casos de uso, y para culminar se muestran las pruebas funcionales realizadas y sus resultados.

4.1 Objetivo General

Implementar un Módulo que permita gestionar los procesos de planificación docente, a través del uso de herramientas tecnológicas que simplifiquen la realización de estas actividades.

4.2 Objetivos Específicos

1. Permitir a los docentes llenar un formulario con sus preferencias y enviarlo a su respectivo coordinador de centro o jefe de departamento en su defecto.
2. Proporcionar la funcionalidad de rechazar o aceptar las propuestas recibidas a los Coordinadores de Centro y Jefe de Departamento.
3. En caso de haber un rechazo de preferencias, permitir a los Coordinadores de Centro y Jefe de Departamento enviar un mensaje que explique el porqué del rechazo de las propuestas.
4. Otorgar a los docentes la facilidad de poder editar sus preferencias en caso de recibir una respuesta de rechazo por parte del Coordinador o Jefe de Departamento.
5. Permitir a los usuarios involucrados en el flujo del proceso verificar el estatus de las preferencias.
6. Ofrecer a los usuarios autorizados una interfaz con la bitácora de las actividades realizadas en el sistema.
7. Proveer al Jefe de Departamento una interfaz con una tabla dinámica editable que plasmará la información a exportar como documento con la planificación docente propuesta.
8. Proveer una interfaz para que los usuarios puedan editar sus datos personales.
9. Asignar roles a los distintos usuarios.
10. Enviar notificaciones masivas por correo.

4.3 Lista de requerimientos

Luego de haber realizado varias reuniones con el Profesor Carlos Acosta, la Profesora Yusneyi Carballo y el Personal Administrativo se llegó a un acuerdo sobre los requerimientos que la aplicación debería enmarcar dentro de su alcance. Cabe destacar que a medida que fue evolucionando la aplicación y debido a la naturaleza de la metodología aplicada surgieron nuevos requerimientos necesarios para el cumplimiento de los objetivos.

4.3.1 Requerimientos funcionales

Un requisito o requerimiento funcional es una función específica que cumple el sistema, es decir, es una de las partes del sistema completo. A continuación se listan los requerimientos funcionales de esta aplicación:

1. Permitir a los usuarios autenticarse con su correo electrónico y contraseña para ingresar a la aplicación.
2. Otorgar a los usuarios la funcionalidad de recuperar su contraseña en caso de haberla olvidado.
3. Permitir a los profesores llenar un formulario con sus preferencias para la programación docente.
4. Permitir a los profesores adjuntos a un centro enviar sus preferencias al coordinador, mientras que los profesores no adjuntos las envían al jefe de departamento.
5. Proporcionar a los coordinadores de centro y al jefe de departamento la funcionalidad de aprobar o rechazar las propuestas recibidas.
6. En caso de rechazo, enviar un mensaje en donde describa las razones del mismo.
7. Permitir a los profesores editar sus propuestas en caso de rechazo.
8. Otorgar al jefe de departamento la funcionalidad de acceder a la bitácora del sistema, la cual almacena las actividades realizadas en la aplicación hasta el momento.
9. El jefe de departamento también podrá modificar la tabla dinámica de programación docente de acuerdo a las propuestas recibidas.
10. Proporcionar a los usuarios autorizados la funcionalidad de enviar notificaciones masivas.
11. El administrador del sistema y el jefe de departamento tendrán la posibilidad de crear o modificar centros y materias.
12. Una vez generada la programación docente, permitir a las secretarias consultarla y descargarla.

4.3.2 Requerimientos no funcionales

Los requerimientos no funcionales se refieren a las propiedades o características emergentes que posee el sistema. A diferencia de los requerimientos funcionales estos se aplican frecuentemente al sistema en su totalidad y no a una parte del mismo. A continuación se listan los requisitos no funcionales de la aplicación:

1. Usabilidad
2. Eficiencia
3. Eficacia en los resultados
4. Integridad de la información
5. Tiempo de respuesta
6. Disponibilidad
7. Costo

4.4 Alcance de la aplicación

La magnitud del presente proyecto generó como consecuencia que la gestión de la planificación docente para períodos intensivos se haya excluido de esta solución, por ende quedaría para mejoras futuras sobre esta aplicación.

4.5 Tecnologías y herramientas utilizadas

Para el desarrollo de la aplicación se usó código HTML para las vistas, CSS para las hojas de estilo y Bootstrap como marco de trabajo para el diseño de las interfaces. Del lado del cliente se usó el *framework* AngularJS basado en JavaScript, y del lado del servidor el marco de trabajo Laravel basado en lenguaje PHP. Para facilitar el manejo de los cambios en el código, y el versionamiento del mismo, se usó el manejador de versiones Git. Por último se usó PostgreSQL como sistema de gestión de base de datos.

4.6 Metodología de desarrollo de software utilizada

El método de desarrollo utilizado fue una adaptación de la metodología Scrum estructurada para equipos de desarrollo de una persona, la cual fue llamada Scrum Personal. Además se utilizaron algunos artefactos para modelar la solución, como los diagramas de casos de uso y diagramas de secuencia. Finalmente el proceso fue dividido en 10 iteraciones siguiendo un proceso incremental y dividiendo el desarrollo en etapas de análisis, diseño, código y pruebas.

4.6.1 Iteraciones del Scrum Personal

A continuación se detallan las 10 iteraciones en las que se realizó el proceso de desarrollo:

- **Iteración 1. Levantamiento de Requerimientos:** se hicieron reuniones y entrevistas con el Profesor Carlos Acosta, las profesoras Concettina Di Vasta y Zenaida Castillo, y con el personal administrativo con el objetivo de analizar los procesos de negocio asociados y establecer cuáles serían las funcionalidades del sistema.

- **Iteración 2. Diseño de la interfaz:** se definieron los lineamientos básicos de usabilidad a usar en el diseño de las interfaces de la aplicación.

- **Iteración 3. Manejo de sesiones:** se llegó a un acuerdo sobre el diseño y la implementación a seguir para la gestión de las credenciales de los usuarios para su autenticación en la aplicación.

- **Iteración 4 - Formulario de preferencias:** se estableció el diseño e implementación del formulario de propuestas que deben llenar los profesores para la conformación de la programación docente.

- **Iteración 5. Flujo de preferencias docentes:** se estableció y se implementó el flujo que debía seguir el proceso de gestión de propuestas docentes.

- **Iteración 6. Envío de notificaciones masivas:** se creó una funcionalidad que permite enviar notificaciones a un grupo de usuarios al mismo tiempo. Se envían a través del sistema e igualmente vía correo electrónico.

- **Iteración 7. Tabla de Programación Docente:** se diseñó e implementó una tabla dinámica de programación docente, la cual permite modificar los distintos campos y finalmente generar un archivo con la programación docente tentativa.

- **Iteración 8. Gestión de roles:** creación de un módulo que permite agregarles o eliminarles roles a los usuarios.

- **Iteración 9. Gestión de usuarios:** se definió un módulo para crear, editar o eliminar usuarios.

- **Iteración 10. Pruebas y entrega:** se realizaron las pruebas correspondientes a todas las funcionalidades del sistema y se corrigieron los errores encontrados.

4.7 Descripción de la aplicación desarrollada para el Módulo

A continuación se describen las funcionalidades del Sistema de Gestión de la Programación Docente (en adelante SIGEPROD), también se especifican sus diagramas de casos de uso y de secuencia, el diseño de la base de datos, las principales interfaces de la aplicación y los resultados obtenidos una vez realizadas las pruebas.

4.7.1 Funcionalidades más importantes de la aplicación

Entre las funcionalidades de mayor importancia en el marco del proceso de conformación de la programación docente se muestran las siguientes:

Formulario de Preferencias: Los usuarios que posean el perfil de profesor deberán llenar un formulario que consta de tres preferencias que conforman la propuesta del usuario para el próximo semestre. Cada preferencia consta de: el nombre de la materia a dictar, su carga académica, horarios y la posibilidad de coordinar la materia. El usuario debe rellenar todos los campos y colocar tres materias distintas para que el envío de preferencias sea satisfactorio.

Estado de las Preferencias: El estado es el que determina la condición en la que se encuentran las preferencias en un determinado momento. A través de ellas los usuarios pueden saber quién debe dar el siguiente paso dentro del flujo del proceso en un momento dado. Por ejemplo, si el estado de las preferencias de un profesor es "Pendiente por aprobación del Coordinador de Centro" significa que hasta que el coordinador no apruebe o rechace las propuestas, el profesor no puede realizar ninguna actividad, pero si el estado es "Rechazado" significa que el profesor debe editar las propuestas y enviarlas nuevamente para su aprobación.

Bitácora de actividades: Otra de las funcionalidades de la aplicación es la bitácora de actividades, la cual permite al jefe de departamento o al administrador del sistema chequear las actividades que se han ido realizando a lo largo del proceso, así como su autor, fecha y horario, y el orden en que estas actividades se han ido ejecutando.

Envío de Notificaciones Masivas: El sistema permite al jefe de departamento, administrador del sistema, coordinadores y secretarias enviar notificaciones a un grupo de usuarios al mismo tiempo, con el objetivo de establecer a través de la

aplicación toda la comunicación necesaria para el cumplimiento del objetivo principal. Estas notificaciones también serán enviadas al correo electrónico de cada remitente.

Tabla dinámica de Programación Docente: El jefe de departamento tendrá a su disposición una tabla dinámica que permite la modificación de la planificación docente, una vez finalizada el usuario podrá generarla en formato pdf o csv. Es importante resaltar que solo será mostrada en la tabla la información de profesores cuyas preferencias hayan sido aprobadas previamente por el jefe de departamento.

4.7.2 Diseño de la base de datos

Luego de estudiar las diferentes funcionalidades que posee la aplicación se procede a diseñar la base de datos, cuya construcción se realizó con la ayuda de las funcionalidades que posee el *framework* Laravel.

Después de tener el conocimiento de todas las tablas que debe contener la aplicación se creó el diagrama entidad-relación con la ayuda de la herramienta MySQL Workbench como se muestra en la figura 7.

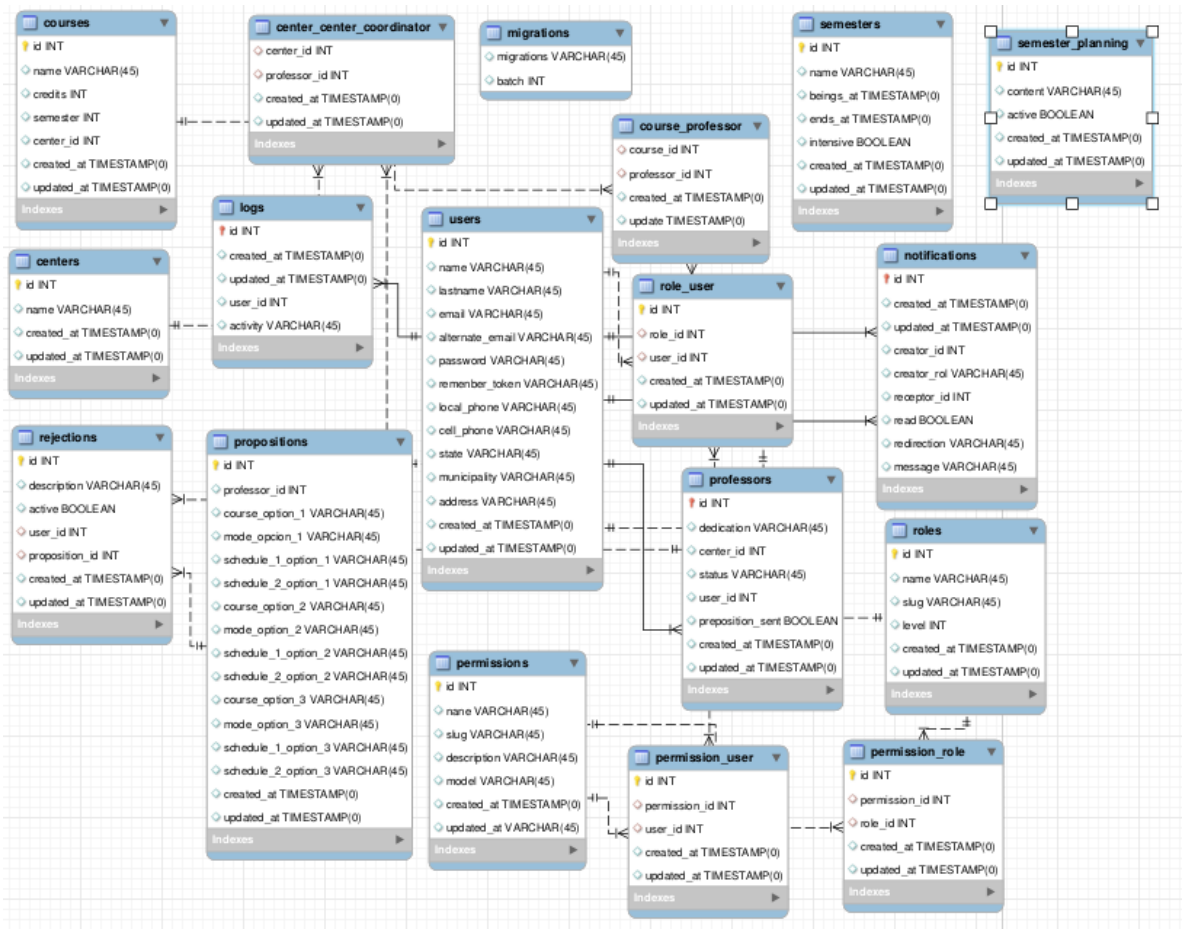

Figura 7 - Diagrama entidad-relación de la base de datos

4.7.3 Usuarios y roles de la aplicación

Los usuarios de la aplicación están conformados por el personal docente (profesores), personal administrativo, y autoridades de la Escuela de Computación.

La siguiente tabla muestra la relación entre los roles definidos sobre la aplicación y las funcionalidades que estos poseen:

Funciones	Creación	Consulta	Modificación	Creación	Consulta	Modificación	Notificaciones	Solicitudes de concurso	Llenar formulario de propuestas	Aceptar o rechazar propuestas	Consulta	Modificación	Exportación	Acceder a la bitácora del sistema	Documentos
Perfil	**Usuarios y perfiles**			**Materias y Centros**							**Planificación Docente Tentativa**				
Administrador del sistema	Si			Si			Enviar	n/a	n/a	n/a	n/a			n/a	n/a
Jefe de Departamento	n/a			Si			Enviar	Gestionar	n/a	Si	Si			Si	Exportar
Director de Escuela	n/a			Si			Enviar	Gestionar	n/a	Si	Si			Si	Exportar
Secretaria del Departamento	n/a			Si (Solo consulta)			n/a	Consultar	n/a	n/a	Si (Solo consulta)			n/a	Exportar
Secretaria del Dirección	n/a			Si (Solo consulta)			n/a	Consultar	n/a	n/a	Si (Solo consulta)			n/a	Exportar
Coordinador de Centro	n/a			Si (Solo consulta)			Enviar	Gestionar	n/a	Si	n/a			n/a	n/a
Coordinador de Materia	n/a			Si (Solo consulta)			Enviar	Gestionar	n/a	n/a	n/a			n/a	n/a
Profesor	n/a			Si (Solo consulta)			n/a	n/a	Si	n/a	n/a			n/a	n/a
Auxiliar Docente	n/a			n/a			n/a	n/a	n/a	n/a	n/a			n/a	Importar y exportar
Preparador	n/a			n/a			n/a	n/a	n/a	n/a	n/a			n/a	Importar y exportar

Tabla 2 - Tabla de roles y funcionalidades
Fuente: Creado por el autor (2015)

4.7.4 Principales diagramas de secuencia y casos de uso

A continuación se aprecia con mayor detalle el proceso de las diferentes funciones que formarán parte de la solución de la gestión de usuarios, materias y centros, envío de notificaciones y gestión de propuestas docentes; utilizando diagramas de secuencia.

Gestión de Usuarios: La solución para este proceso involucra al administrador del sistema que está en la posibilidad de crear, modificar o eliminar cualquier usuario registrado en el sistema.

Los pasos a aplicar serían tanto para creación, edición y eliminación de usuarios:

1. El administrador del sistema ingresa a la aplicación.

2. Selecciona la opción “Gestionar usuarios”.
    a. En caso de creación, el administrador selecciona la pestaña “Crear Usuario” e ingresa la información asociada, de esta forma el usuario es creado en el sistema.
    b. En caso de modificación, el administrador debe hacer clic en la pestaña “Editar Usuario”, seleccionar el usuario a modificar y posteriormente ingresar la información que se va a actualizar en el sistema.
    c. En caso de eliminación, el administrador debe hacer clic en la pestaña “Eliminar Usuario” y seleccionar el usuario que va a ser eliminado del sistema.

Gestión de Materias y Centros: Este proceso involucra al administrador del sistema y al jefe de departamento quienes podrán crear o editar parámetros asociados a las materias o centros.

Los pasos a seguir para la creación o modificación de centros o materias son los siguientes:

1. El usuario ingresa a la aplicación.
2. Selecciona la opción "Gestionar Centros" o "Gestionar Materias".
    a. En caso de creación, el usuario ingresa los parámetros asociados al elemento nuevo y este es creado en el sistema.
    b. En caso de seleccionar la opción de edición, el usuario ingresa los parámetros a modificar y este es actualizado en el sistema.

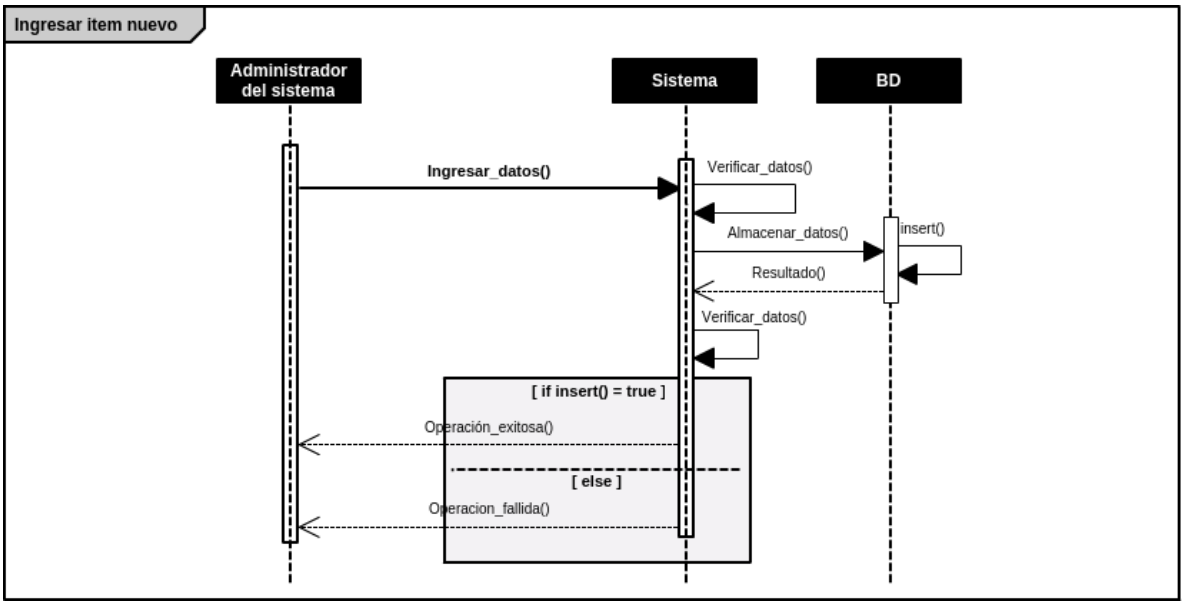

Figura 8: Diagrama de Secuencia - Ingresar ítem nuevo

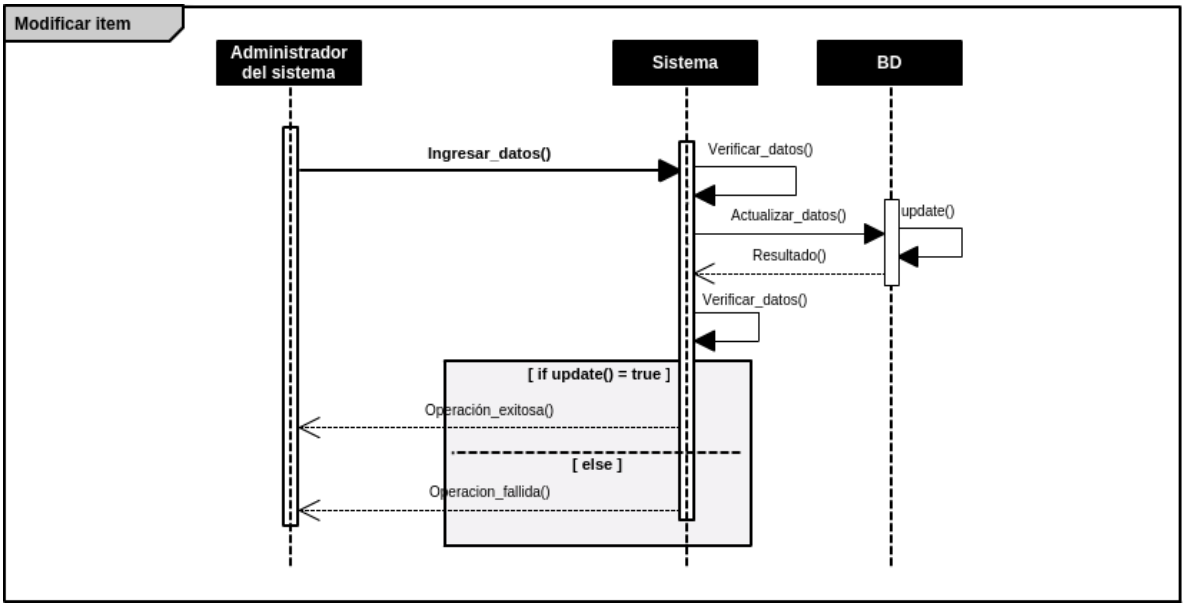

Figura 9: Diagrama de Secuencia - Modificar ítem

Estos diagramas indican los procesos para la creación o modificación de un usuario, materia o centro, es decir, se utiliza la palabra *ítem* para generalizar cualquiera de estos elementos.

Envío de Notificaciones: La solución de este proceso involucra al administrador del sistema, jefe de departamento y los coordinadores de centro, los cuales podrán enviar notificaciones masivas a uno o varios usuarios a través de la aplicación.

Para realizar este proceso se deben seguir los siguientes pasos:

1. El usuario ingresa a la aplicación.
2. Selecciona la opción “Envió de Notificaciones”.
3. Selecciona los usuarios destinatarios, título, contenido y el enlace de destino del mensaje.

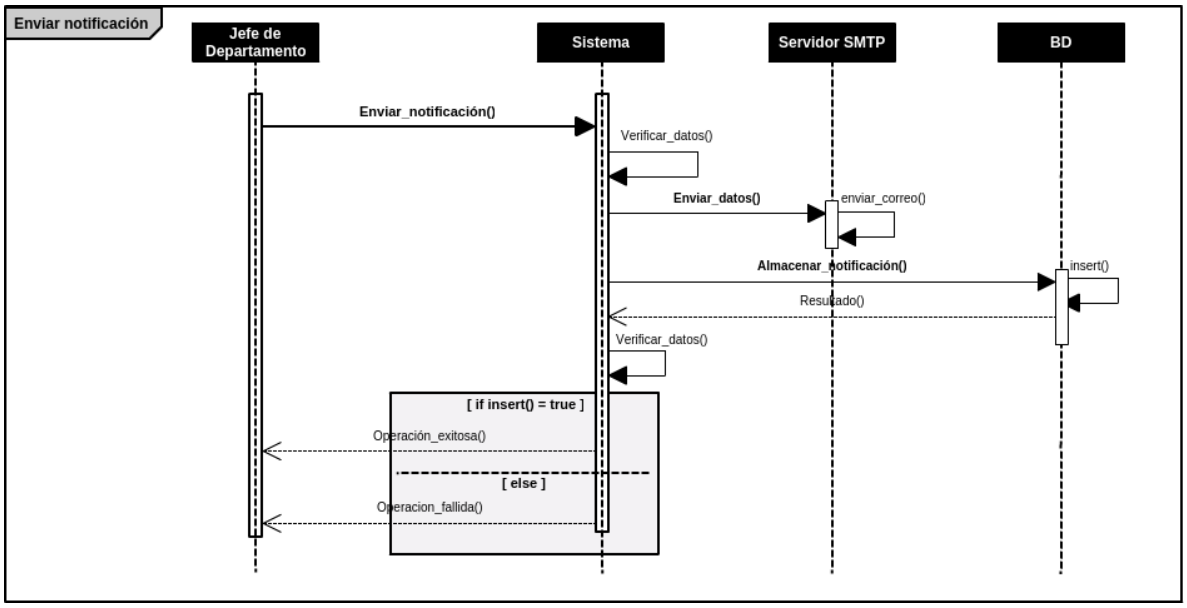

Figura 10: Diagrama de Secuencia - Enviar notificación

Gestión de Propuestas Docentes: La solución de este proceso involucra a los profesores, coordinadores de centro y al jefe de departamento. Estos roles tendrán la posibilidad de crear, editar, aprobar o rechazar las propuestas para la programación docente.

Para la ejecución de este proceso se deben realizar una serie de actividades descritas a continuación:

1. El usuario ingresa a la aplicación.
2. Selecciona la opción "Gestión de Propuestas".
3. Los profesores envían sus preferencias para la programación docente.
4. Los coordinadores de centro chequean las preferencias enviadas por los profesores y las aprueban o las rechazan. En caso de rechazo los profesores editan sus preferencias y las reenvían para su aprobación.
5. El jefe de departamento revisa las propuestas enviadas por los coordinadores de centro y profesores no adjuntos a ningún centro y procede a aprobarlas o rechazarlas. En caso de rechazo el profesor debe editarlas y enviarlas de nuevo para su posterior aprobación.
6. El jefe de departamento genera la programación docente, partiendo de las propuestas aprobadas previamente.

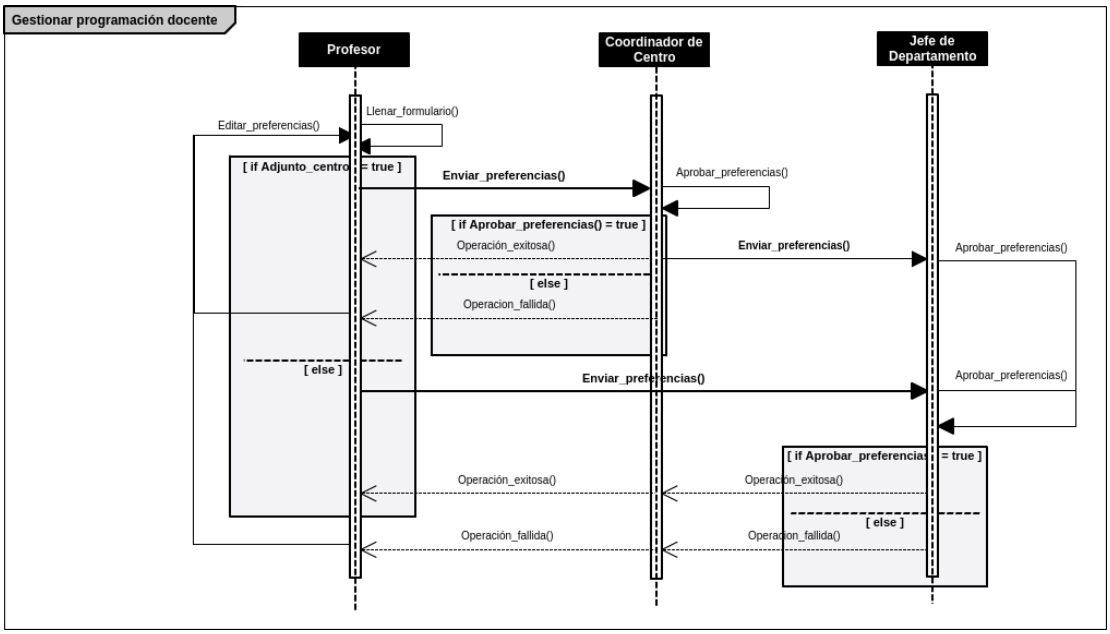

Figura 11. Diagrama de Secuencia - Gestionar programación docente

A continuación se muestran los diferentes niveles de los diagramas de casos de uso y las especificaciones de los casos de uso más importantes.

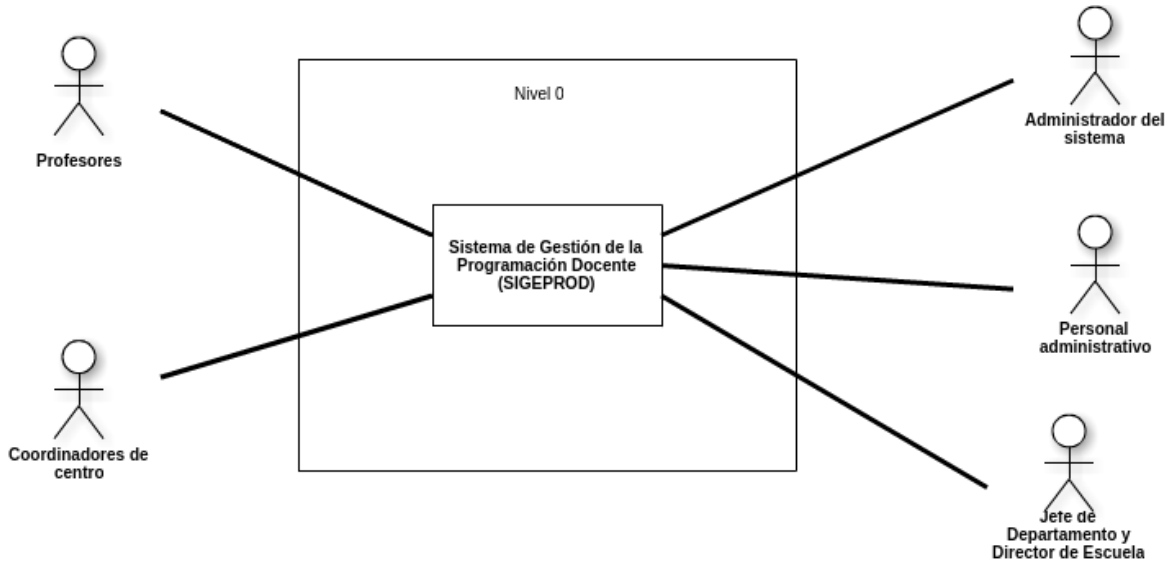

Figura 12 - Diagrama de casos de uso - Nivel 0

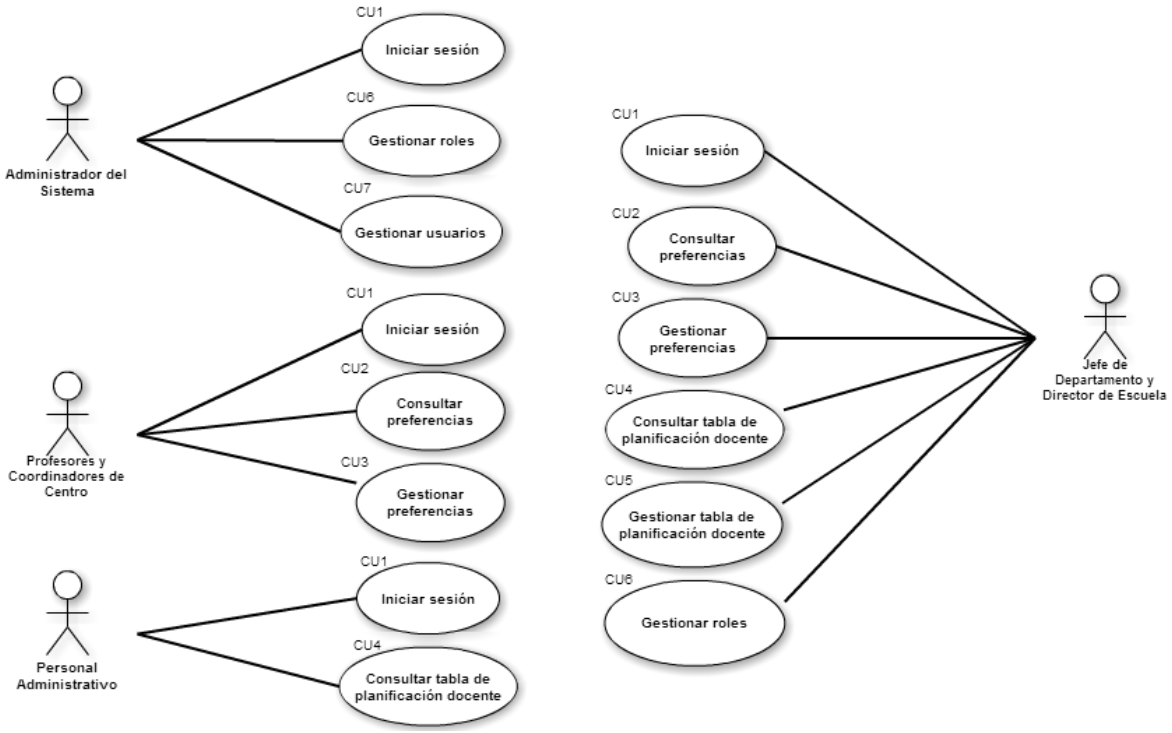

Figura 13 - Diagrama de casos de uso - Nivel 1

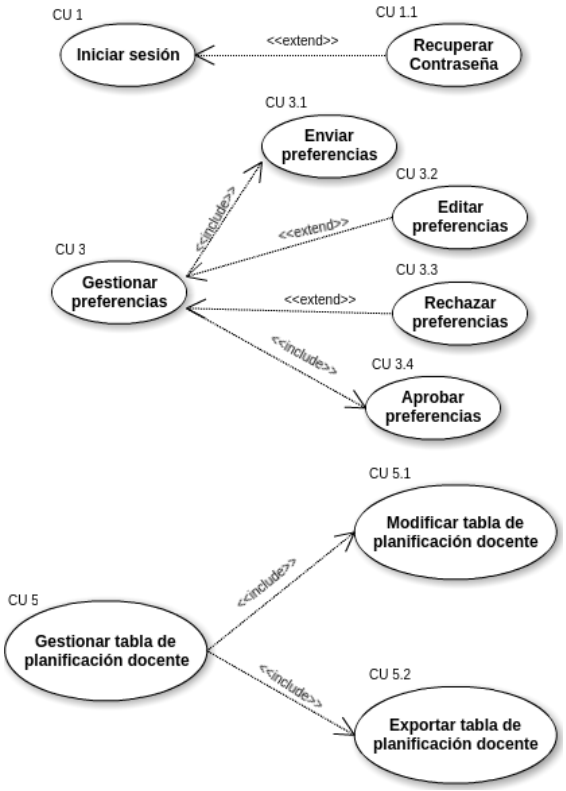

Figura 14 - Diagrama de casos de uso - Nivel 2

Especificación de los casos de uso más importantes:

Caso de uso: Recuperar contraseña
Actores: Todos los usuarios

Flujo normal	**Flujo alternativo**
1) En la pantalla de inicio de sesión, el usuario selecciona la opción "Recuperar contraseña".	
2) Se introduce el correo electrónico asociado a la cuenta de la cual se quiere recuperar la contraseña.	
3) Es enviado un correo a la dirección de correo electrónico especificada, con una nueva clave generada por el sistema.	Si el correo electrónico no existe en el sistema, se muestra un mensaje de error notificando al usuario.

Tabla 3 - Especificación de casos de uso - Recuperar contraseña

Caso de uso: Enviar preferencias
Actores: Profesores

Flujo normal	Flujo alternativo
1) Una vez autenticado el usuario debe hacer clic en la pestaña "Gestión de propuestas" en el menú lateral izquierdo.	
2) Se muestra un formulario en el cual el usuario debe seleccionar tres materias con su respectivo horario y carga académica. Finalmente debe hacer clic en el botón "Enviar".	Si el usuario no introduce toda la información solicitada, se le notifica a través de un mensaje de error.
3) Se muestran las opciones seleccionadas por el usuario para su consulta y verificación. Además las preferencias tienen el estado "Pendiente por aprobación del Coordinador de Centro" o "Pendiente por aprobación del Jefe de Departamento" según sea el caso.	

Tabla 4 - Especificación de casos de uso - Enviar preferencias

Caso de uso: Editar preferencias
Actores: Profesores

Flujo normal	Flujo alternativo
1) El Profesor selecciona la opción "Gestión de propuestas" en el menú lateral izquierdo.	
2) El Profesor expande las preferencias enviadas que deben tener el estado "Rechazado" y hace clic sobre el botón "Editar".	
3) Se despliega una ventana modal de edición de las preferencias, que permite al usuario modificar las preferencias enviadas anteriormente, y donde se muestran las razones del rechazo en cuestión.	Es posible cerrar la ventana modal de edición. Además, si alguna de las propuestas está incompleta o falta algún campo por llenar se muestra un mensaje de error notificándole al usuario.
4) Se actualiza el estado de las preferencias a "Pendiente por aprobación del Coordinador de	

Centro" o "Pendiente por aprobación del Jefe de Departamento" según sea el caso. El Coordinador o Jefe de Departamento remitente recibe una notificación.	

Tabla 5 - Especificación de casos de uso - Editar preferencias

Caso de uso: Rechazar preferencias
Actores: Coordinadores de Centro y Jefe de Departamento

Flujo normal	**Flujo alternativo**
1) El usuario selecciona la opción "Gestión de Propuestas" en el menú lateral.	
2) El usuario expande una de las preferencias con estado "Pendiente" y hace clic sobre el botón "Rechazar".	
3) Se abre una ventana modal de confirmación, permitiéndole al usuario expresar las razones del rechazo de las preferencias.	Es posible cerrar la ventana modal de confirmación, cancelando así el rechazo de las preferencias.
4) Se actualiza el estado de las preferencias, cambiándolo a "Rechazado". En caso de que se hayan agregado las observaciones del rechazo, estas son agregadas a la información de las preferencias, y el Profesor correspondiente recibe una notificación.	

Tabla 6 - Especificación de casos de uso - Rechazar preferencias

Caso de uso: Aprobar preferencias
Actores: Coordinadores de Centro y Jefe de Departamento

Flujo normal	**Flujo alternativo**
1) El usuario selecciona la opción "Gestión de Propuestas" en el menú lateral.	
2) El usuario expande una de las preferencias con estado "Pendiente" y hace clic sobre el botón "Aprobar".	

3) Se actualiza el estado de las preferencias, cambiándolo a "Pendiente por aprobación del Jefe de Departamento" en el caso de los Coordinadores o "Aprobado" en el caso del Jefe de Departamento. Además se le envía una notificación al Profesor emisor de las preferencias y en el caso de aprobación por parte del Jefe de Departamento se le notifica también al Coordinador de Centro correspondiente, en el caso de los Coordinadores se le envía la notificación al Jefe de Departamento.	

Tabla 7 - Especificación de casos de uso - Aprobar preferencias

Caso de uso: Modificar tabla de planificación docente
Actores: Jefe de Departamento

Flujo normal	**Flujo alternativo**
1) El Jefe de Departamento selecciona la opción "Programación Docente" en el menú lateral izquierdo.	
2) Se debe hacer clic sobre la pestaña "Gestión de la Programación Docente".	
3) El Jefe de Departamento selecciona los nombres, dedicación, secciones, centros y horarios de los profesores para cada materia. Los profesores disponibles solo serán aquellos cuyas preferencias hayan sido aprobadas por el Jefe de Departamento previamente.	El Jefe de Departamento puede exportar la tabla sin realizar ninguna modificación.

Tabla 8 - Especificación de casos de uso - Modificar tabla de planificación docente

Caso de uso: Exportar tabla de planificación docente
Actores: Jefe de Departamento, Personal Administrativo

Flujo normal	**Flujo alternativo**
1) El usuario selecciona la opción	

"Programación Docente" en el menú lateral izquierdo.	
2) Luego debe hacer clic sobre la pestaña "Gestión de la Programación Docente.	
3) El usuario hace clic sobre el botón de la esquina superior derecha de la tabla y se despliegan una serie de opciones para la exportación.	
4) Se escoge el formato en el que se va a realizar la exportación (pdf o csv) y se hace clic sobre el elemento.	La tabla permite al usuario eliminar columnas de la exportación, así como también permite seleccionar y exportar solo un grupo determinado de filas en caso de ser necesario.

Tabla 9 - Especificación de casos de uso - Exportar tabla de planificación docente

4.7.5 Pruebas y resultados

Para validar el correcto funcionamiento de las distintas funcionalidades que provee el módulo de gestión docente del sistema web, se realizaron pruebas funcionales para determinar los resultados que se obtenían después de aplicar diferentes comportamientos sobre cada funcionalidad y ver si dichos resultados concuerdan con los resultados esperados.

Prueba N° 1 – Inicio de sesión

Primera entrada	Segunda entrada	Tercera entrada
Entrada: vacío	**Entrada:** credenciales incorrectas	**Entrada:** credenciales válidas
Resultado esperado: se muestra un mensaje de error notificándole al usuario que sus credenciales son incorrectas	**Resultado esperado**: se muestra un mensaje de error notificándole al usuario que sus credenciales son incorrectas	**Resultado esperado**: se muestra la página de bienvenida de la aplicación y se muestra un mensaje notificando que la autenticación fue satisfactoria.
Resultado obtenido: se muestra un mensaje de error notificándole al usuario que sus credenciales son incorrectas.	**Resultado obtenido**: se muestra un mensaje de error notificándole al usuario que sus credenciales son incorrectas.	**Resultado obtenido**: se muestra la página de bienvenida de la aplicación y se muestra un mensaje notificando que la autenticación fue satisfactoria.

Tabla 10 – Pruebas y resultados – Inicio de sesión

Prueba N° 2 – Recuperación de contraseña

Primera entrada	Segunda entrada	Tercera entrada
Entrada: vacío	**Entrada:** correo electrónico incorrecto	**Entrada:** correo electrónico válido
Resultado esperado: se muestra un mensaje de error notificándole al usuario que sus credenciales son incorrectas	**Resultado esperado**: se muestra un mensaje de error notificándole al usuario que sus credenciales son incorrectas	**Resultado esperado**: se envía el correo con la contraseña de recuperación y se redirecciona la página al inicio de sesión.
Resultado obtenido: se muestra un mensaje de error notificándole al usuario que sus credenciales son incorrectas.	**Resultado obtenido**: se muestra un mensaje de error notificándole al usuario que sus credenciales son incorrectas.	**Resultado obtenido**: se envía el correo con la contraseña de recuperación y se redirecciona la página al inicio de sesión.

Tabla 11 – Pruebas y resultados – Recuperación de contraseña

Prueba N° 3 – Envío de preferencias

Primera entrada	Segunda entrada	Tercera entrada
Entrada: vacío	**Entrada:** seleccionando una misma materia más de una vez	**Entrada:** todos los campos válidos y materias distintas
Resultado esperado: se muestran varios mensajes notificando al usuario los campos que le faltó por rellenar.	**Resultado esperado**: se muestra un mensaje notificando al usuario que debe seleccionar tres materias diferentes.	**Resultado esperado**: se muestra un mensaje notificando que el envío de preferencia fue satisfactorio y se recarga la página mostrando las propuestas enviadas para verificar que son las correctas.
Resultado obtenido: se muestran varios mensajes notificando al usuario los campos que le faltó por rellenar.	**Resultado obtenido**: se muestra un mensaje notificando al usuario que debe seleccionar tres materias diferentes.	**Resultado obtenido**: se muestra un mensaje notificando que el envío de preferencia fue satisfactorio y se recarga la página mostrando las propuestas enviadas para verificar que son las correctas.

Tabla 12 – Pruebas y resultados – Envío de preferencias

Los resultados de la tabla 13 aplican tanto para el envío de preferencias la primera vez como para la modificación de preferencias después de un rechazo.

Prueba N° 4 – Aprobación de preferencias

Coordinador de Centro	Jefe de Departamento
Resultado esperado: Al hacer clic en el botón "Aprobar" cambia el estado de "Pendiente" a "Pendiente por aprobación del Jefe de Departamento" y se muestra un mensaje notificando que se aprobaron satisfactoriamente.	**Resultado esperado**: Al hacer clic en el botón "Aprobar" cambia el estado de "Pendiente" a "Aprobado" y se muestra un mensaje notificando que se aprobaron satisfactoriamente. Además las preferencias aprobadas deben estar disponibles inmediatamente en la tabla de planificación docente.
Resultado obtenido: Al hacer clic en el botón "Aprobar" cambia el estado de "Pendiente" a "Pendiente por aprobación del Jefe de Departamento" y se muestra un mensaje notificando que se aprobaron satisfactoriamente.	**Resultado obtenido**: Al hacer clic en el botón "Aprobar" cambia el estado de "Pendiente" a "Aprobado" y se muestra un mensaje notificando que se aprobaron satisfactoriamente. Además las preferencias aprobadas deben estar disponibles inmediatamente en la tabla de planificación docente.

Tabla 13 – Pruebas y resultados – Aprobación de preferencias

Prueba N° 5 Rechazo de preferencias

Coordinador de Centro	Jefe de Departamento
Resultado esperado: Al hacer clic en el botón "Rechazar" se abre una ventana modal para colocar un mensaje exponiendo las razones del rechazo. Además se muestra un checkbox con opción de notificarle al profesor vía correo electrónico. Luego después de hacer clic en el botón "Enviar" se envía el mensaje, cambia el estado de las preferencias a "Rechazado" y se le notifica al usuario a través del sistema, si se marcó el checkbox también se le notifica vía correo electrónico.	**Resultado esperado**: Al hacer clic en el botón "Rechazar" se abre una ventana modal para colocar un mensaje exponiendo las razones del rechazo. Además se muestra un checkbox con opción de notificarle al profesor vía correo electrónico. Luego después de hacer clic en el botón "Enviar" se envía el mensaje, cambia el estado de las preferencias a "Rechazado" y se le notifica tanto al coordinador de su centro como al profesor a través del sistema, si se marcó el checkbox también se le notifica al profesor vía correo electrónico.
Resultado obtenido: Al hacer clic en el botón "Rechazar" se abre una ventana modal para colocar un mensaje exponiendo las razones del rechazo. Además se muestra un checkbox con opción de notificarle al profesor vía correo electrónico. Luego después de hacer clic en el botón "Enviar" se envía el mensaje, cambia el estado de las preferencias a "Rechazado" y se le notifica al usuario a través del sistema, si se marcó el checkbox también se le notifica vía correo electrónico.	**Resultado obtenido**: Al hacer clic en el botón "Rechazar" se abre una ventana modal para colocar un mensaje exponiendo las razones del rechazo. Además se muestra un checkbox con opción de notificarle al profesor vía correo electrónico. Luego después de hacer clic en el botón "Enviar" se envía el mensaje, cambia el estado de las preferencias a "Rechazado" y se le notifica tanto al coordinador de su centro como al profesor a través del sistema, si se marcó el checkbox también se le notifica al profesor vía correo electrónico.

Tabla 14 – Pruebas y resultados – Rechazo de preferencias

Prueba N° 6 - Tabla de Planificación Docente

Modificación de la tabla	Exportación de la tabla
Resultado esperado: El usuario puede editar los campos modificables de la tabla como profesores, horarios, secciones, centros, entre otros a través de selección de listas múltiples con las preferencias recibidas previamente y en otros casos campos de texto.	**Resultado esperado**: Al hacer clic en el botón de la esquina superior derecha de la tabla se despliegan una serie de opciones para la exportación como descargar en formato pdf o csv. Luego de seleccionar la opción correspondiente se realiza la descarga y se muestra un mensaje de generación exitosa de la programación docente tentativa.
Resultado obtenido: El usuario puede editar los campos modificables de la tabla como profesores, horarios, secciones, centros, entre otros a través de selección de listas múltiples con las preferencias recibidas previamente y en otros casos campos de texto.	**Resultado obtenido**: Al hacer clic en el botón de la esquina superior derecha de la tabla se despliegan una serie de opciones para la exportación como descargar en formato pdf o csv. Luego de seleccionar la opción correspondiente se realiza la descarga y se muestra un mensaje de generación exitosa de la programación docente tentativa.

Tabla 15 – Pruebas y resultados – Tabla de Planificación Docente

Las pruebas realizadas confirmaron el correcto funcionamiento de las distintas partes del Módulo. A cada funcionalidad se le aplicaron distintas entradas para verificar que el comportamiento era el esperado. De la misma forma se espera que una vez que el sistema esté funcionando en los servidores del Departamento de Computación los usuarios puedan ser testigos de las ventajas que ofrece el Módulo de Gestión Docente de SIGEPROD (Sistema de Gestión de la Programación Docente).

Por otro lado, este desarrollo intentó establecer parámetros de seguridad básicos sobre este sistema. Por lo que se plantea para mejoras futuras implementar una seguridad más avanzada y posteriormente realizar las pruebas de seguridad y de vulnerabilidad correspondientes.

RESULTADOS DEL TRABAJO ESPECIAL DE GRADO

Después de la realización de este proyecto, se pudo notar la mejora que significa la integración de SIGEPROD (Sistema de Gestión de la Programación Docente) en los procesos del Departamento de Computación de la Facultad de Ciencias. Esta aplicación cambia el enfoque de los actores sobre el proceso, debido a que se realiza de manera más sencilla, se ahorran recursos, y se organiza el flujo de las actividades a realizar.

Por otra parte, este sistema mejora la comunicación necesaria para el cumplimiento del objetivo, y esta se realiza netamente a través de la aplicación con el fin de que los actores se involucren en la gestión del proceso. Otro punto importante y de mejora en el desempeño del proceso es que se mantiene informado a los usuarios de las actividades que se van realizando para que estos tengan conocimiento de las etapas por las que va pasando el proceso.

Además es importante tomar en cuenta que con la implementación de este sistema aumentan las probabilidades de calcular el tiempo de duración del proceso debido a que se disminuyen los factores que generan retrasos en la realización del mismo.

Finalmente se puede decir que se logró cumplir el objetivo principal de este proyecto, que era desarrollar una aplicación web que ayude a mejorar y simplificar los procesos de gestión de la programación docente en la Escuela de Computación de la Universidad Central de Venezuela. Esperamos que una vez esté en producción y sea utilizado, se pueda conocer realmente los beneficios en los procesos que gestiona e incorporar algunas mejoras adicionales.

CONCLUSIONES Y RECOMENDACIONES

Como conclusiones podemos decir que el estudio realizado sobre la metodología, herramientas y tecnologías a usar en la realización de este proyecto fue acertado. Todos estos elementos permitieron facilitar el diseño y la implementación de este sistema gracias a las ventajas que ofrecen.

En cuanto a la metodología usada en este proyecto, Scrum garantizó un proceso de desarrollo adaptable a los cambios. Esta metodología permitió que la aplicación incrementara la cantidad de funcionalidades a ofrecer a medida que se iban probando y al mismo tiempo generando nuevos requerimientos que eran cubiertos posteriormente.

Entre las tecnologías que significaron un mayor acierto al momento de ser elegidas para este proyecto se encuentran los *frameworks* Laravel y AngularJS. A pesar de que anteriormente ambos *frameworks* no habían sido utilizados, se dedicó gran parte del tiempo a conocer todas las bondades que ofrecían cada uno de ellos. La curva de aprendizaje del *framework* AngularJS no fue muy rápida debido a que anteriormente no se habían usado en su totalidad *frameworks* para el desarrollo del lado del cliente, sin embargo fue un acierto sin duda alguna. Laravel por su parte tuvo una curva de aprendizaje bastante rápida ya que anteriormente se habían usado *frameworks* de ese tipo como CakePHP o CodeIgniter.

Por otra parte el *framework* Bootstrap permitió facilitar el diseño de las interfaces y mejorar la usabilidad de la aplicación. A pesar de no haber sido usado anteriormente fue bastante rápido su aprendizaje.

Finalmente podemos decir que los objetivos planteados en un principio se cumplieron satisfactoriamente.

Como recomendaciones para trabajos futuros de este Trabajo Especial de Grado se tienen las siguientes:

- Que el sistema pueda ser accedido desde la página de la Escuela de Computación.
- Realizar pruebas de seguridad tanto del lado del cliente como del lado del servidor.
- Integrar el Sistema de Gestión de la Programación Docente con otros módulos desarrollados en la escuela a fin de promover una retroalimentación entre los distintos sistemas.

REFERENCIAS BIBLIOGRÁFICAS Y DIGITALES

Academia. (s. f). *RUP*. Recuperado el 28 de Abril de 2015. Disponible en: http://www.academia.edu/8261231/RUP

Acuña, K. (2009). *Metodologías de Desarrollo para Aplicaciones Web*. Recuperado el 03 de Mayo de 2015. Disponible en: http://www.eumed.net/libros-gratis/2009c/584/index.htm

Adrformacion. (2008). *Curso de Calidad ISO 9001:2008.* Recuperado el 22 de Abril de 2015. Disponible en: http://www.adrformacion.com/cursos/calidad08/leccion2/tutorial4.html

AngularJS. (2015). *AngularJS*. Recuperado el 16 de Abril de 2015. Disponible en: https://docs.angularjs.org/guide/introduction

Canós, J., Letelier, P. & Penadés, M. (2003). *Metodologías Ágiles en el Desarrollo de Software.* Recuperado el 30 de Abril de 2015. Disponible en: http://issi.dsic.upv.es/archives/f-1069167248521/actas.pdf

Citón, M. (2006). *Método Ágil Scrum aplicado al desarrollo de un software de trazabilidad.* Recuperado el 03 de mayo de 2015. Disponible en: http://www.um.edu.ar/catedras/claroline/backends/download.php?url=L01ldG9kb3NfQWdpbGVzL01ldG9kb19BZ2lsX1NjcnVtLnBkZg%3D%3D&cidReset=true&cidReq=II0162004

Duckett, J. (2004). *Beginning Web Programming with HTML, XHTML and CSS.* Recuperado el 4 de Abril de 2015. Disponible en: https://books.google.co.ve/books?id=mwrf1EMOcskC&printsec=frontcover&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

Euskalit. (2008). *Calidad Total: Principios y Modelos de Gestión. Certificación ISO. Satisfacción del cliente interno y externo.* Recuperado el 22 de Abril de 2015. Disponible en: https://ope2011.osakidetza.net/procesoselectivo/d26501/docinteres6.pdf

Ecured. (s. f). *Metodología Scrum.* Recuperado el 03 de Mayo de 2015. Disponible en: http://www.ecured.cu/index.php/Metodolog%C3%ADa_Scrum

Fondonorma-ISO 9000. (2006). *Sistemas de Gestión de la Calidad. Fundamentos y Vocabulario.* Recuperado el 22 de Abril de 2015. Disponible en: http://gebolivar.e-bolivar.gov.ve/gebolivar/archivos_iso/ISO9000_2006.pdf

García, F. (2011). *Modelado conceptual de aplicaciones web*. Recuperado el 30 de Abril de 2015. Disponible en: http://repositorio.grial.eu/bitstream/123456789/70/2/20110505%20-%20Modelado%20conceptual%20de%20aplicaciones%20web.pdf

Johnson & Boote. (1988). *Framework*. Recuperado el 16 de Abril de 2015. Disponible en: http://encyclopedia.thefreedictionary.com/Software+framework

Luján, S. (2001). *Programación en Internet: Clientes Web.* Recuperado el 2 de Abril de 2015. Disponible en: http://rua.ua.es/dspace/bitstream/10045/16994/1/sergio_lujan-programacion_en_internet_clientes_web.pdf

Microsoft. (s. f). *Modelo-Vista-Controlador.* Recuperado el 10 de Abril de 2015. Disponible en: https://msdn.microsoft.com/en-us/library/ff649643.aspx

Microsoft. (2005). *Introduction to the Microsoft Solution Framework.* Recuperado el 28 de Abril de 2015. Disponible en: https://technet.microsoft.com/en-us/library/bb497060.aspx

Moreira, V. (2009). *Aplicación Web.* Recuperado el 2 de Abril de 2015. Disponible en: http://www.scribd.com/doc/75239310/Aplicaciones-Web#scribd

Niska, C. (2014). *Extending Bootstrap: Preview.* Recuperado el 4 de Abril de 2015. Disponible en: http://www.it-ebooks.org/book/packt/extending_bootstrap

Oracle. (2008). *Gestión de Procesos de Negocio.* Recuperado el 22 de Abril de 2015. Disponible en: http://www.oracle.com/technetwork/es/middleware/fusion-middleware/documentation/gestion-proceso-negocio-soa-web-450487-esa.pdf

Página Oficial de Laravel. (2015). *Laravel.* Recuperado el 18 de Abril de 2015. Disponible en: http://laravel.com/

Palvia & Nosek. (1993). *Metodologías de Desarrollo de Software.* Recuperado el 30 de Abirl de 2015. Disponible en: https://uvirtual.unet.edu.ve/pluginfile.php/189174/mod_resource/content/0/Metodologias_de_desarrollo_de_Software_IS.pdf

Sebastián, J. (2011). *Ciclos de Vida – Proceso de Desarrollo del Software.* Recuperado el 30 de Abril de 2015. Disponible en: http://www.comusoft.com/ciclos-de-vida-proceso-de-desarrollo-del-software

Sommerville, I. (2002). *Ingeniería del Software: 7ma Edición.* Recuperado el 30 de Abril de 2015. Disponible en: https://books.google.co.ve/books?id=gQWd49zSut4C&printsec=frontcover#v=onepage&q&f=false

Tutorialspoint. (2015). *What is AngularJS?.* Recuperado el 16 de Abril de 2015. Disponible en: http://www.tutorialspoint.com/angularjs/angularjs_overview.htm

Valdés, D. (2007). *¿Qué son las Bases de Datos?.* Recuperado el 12 de Abril de 2015. Disponible en: http://www.maestrosdelweb.com/que-son-las-bases-de-datos/

Wikipedia. (s. f). *PHP.* Recuperado el 10 de Abril de 2015. Disponible en: https://es.wikipedia.org/wiki/PHP

Wikipedia. (s. f). *Framework.* Recuperado el 16 de Abril de 2015. Disponible en: https://es.wikipedia.org/wiki/Framework

Wikipedia. (s. f). *Laravel.* Recuperado el 18 de Abril de 2015. Disponible en: https://es.wikipedia.org/wiki/Laravel

Wikipedia. (s. f). *Arquitectura Cliente-Servidor*. Recuperado el 20 de Abril de 2015. Disponible en: https://es.wikipedia.org/wiki/Cliente-servidor

Wikipedia. (s. f). *Control de versiones.* Recuperado el 24 de Abril de 2015. Disponible en: https://es.wikipedia.org/wiki/Control_de_versiones

Wikipedia. (s. f). *Git.* Recuperado el 24 de Abril de 2015. Disponible en: https://es.wikipedia.org/wiki/Git

Wikipedia. (s. f). *Proceso Racional Unificado.* Recuperado el 28 de Abril de 2015. Disponible en: https://es.wikipedia.org/wiki/Proceso_Unificado_de_Rational

Wikipedia. (s. f). *Scrum*. Recuperado el 03 de mayo de 2015. Disponible en: https://es.wikipedia.org/wiki/Scrum

Worsley, J. & Drake, J. (2002). *PostgreSQL.* Recuperado el 12 de Abril de 2015. Disponible en: https://books.google.es/books?id=G8dh95j5NgcC&printsec=frontcover&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

ANEXO

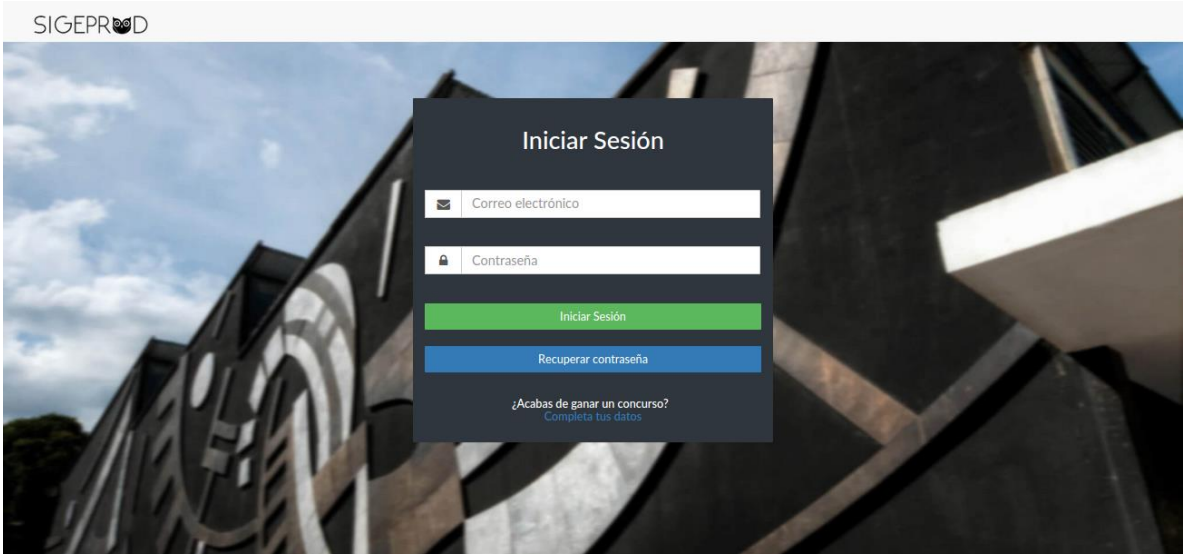

Figura 15: Interfaz de inicio de sesión

En la figura 15 podemos ver la interfaz que visualizan los usuarios una vez accedan a la url de SIGEPROD. El usuario debe ingresar su correo electrónico y su contraseña para ingresar a la aplicación.

Figura 16: Interfaz inicial después de autenticación

La figura 16 muestra la página inicial una vez el usuario es autenticado, la cual muestra el logo de la aplicación y un mensaje de bienvenida al Sistema. Además en la parte superior derecha de la vista se muestra el nombre del usuario y un ícono para acceder a las notificaciones.

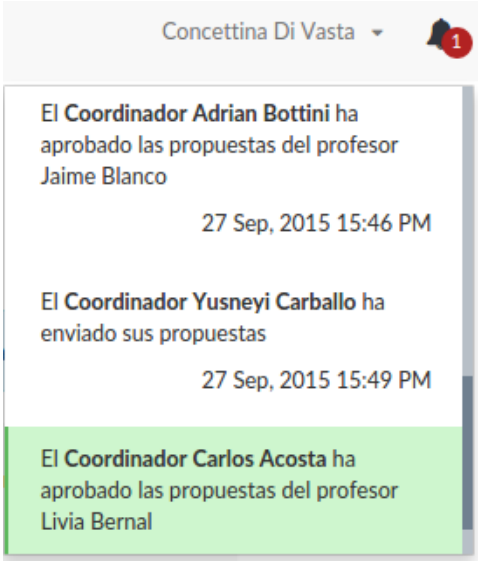

Figura 17: Interfaz de ventana de notificaciones

Una vez el usuario hace clic sobre el ícono se muestran las notificaciones que le han llegado al usuario (como vemos en la figura 17), las cuales contienen un mensaje y la fecha en la que fue recibida dicha notificación.

Figura 18: Interfaz de formulario de preferencias

En la figura 18 se muestra el formulario de preferencias que deberán llenar los Profesores y posteriormente enviar dicha información a sus respectivos Coordinadores de Centro o Jefe de Departamento.

Figura 19: Interfaz de gestión de preferencias (Coordinador de Centro)

La figura 19 muestra la interfaz que podrán visualizar los Coordinadores de Centro al hacer clic en la pestaña "Gestión de Propuestas". Se muestran los nombres de los profesores que han enviado sus preferencias y el estado que poseen. Al hacer clic sobre uno de ellos se expanden y se puede ver el detalle de las propuestas enviadas.

Figura 20: Interfaz de gestión de preferencias (Jefe de Departamento)

La figura 20 muestra la interfaz de gestión de preferencias del Jefe de Departamento, la cual muestra los diferentes centros de investigación y la cantidad de preferencias que ha recibido de cada uno de ellos. Si las preferencias están en estado “Pendiente” tienen el color amarillo, con el estado “Rechazado” en rojo y “Aprobado” en verde.

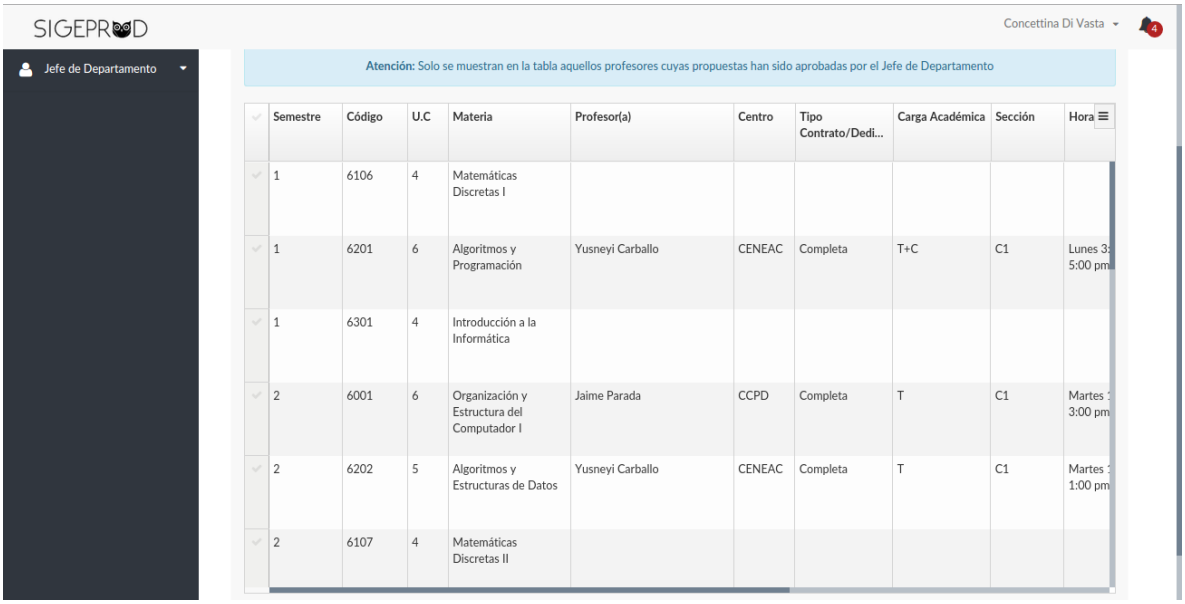

Figura 21: Interfaz de tabla de programación docente

En la figura 21 podemos visualizar la tabla de planificación docente a la cual accederá el Jefe de Departamento haciendo clic en la pestaña “Gestión de la Programación Docente”. El objetivo de esta tabla es mostrar al Jefe de Departamento las preferencias que han sido aprobadas y que el usuario pueda modificarla según sus requerimientos.

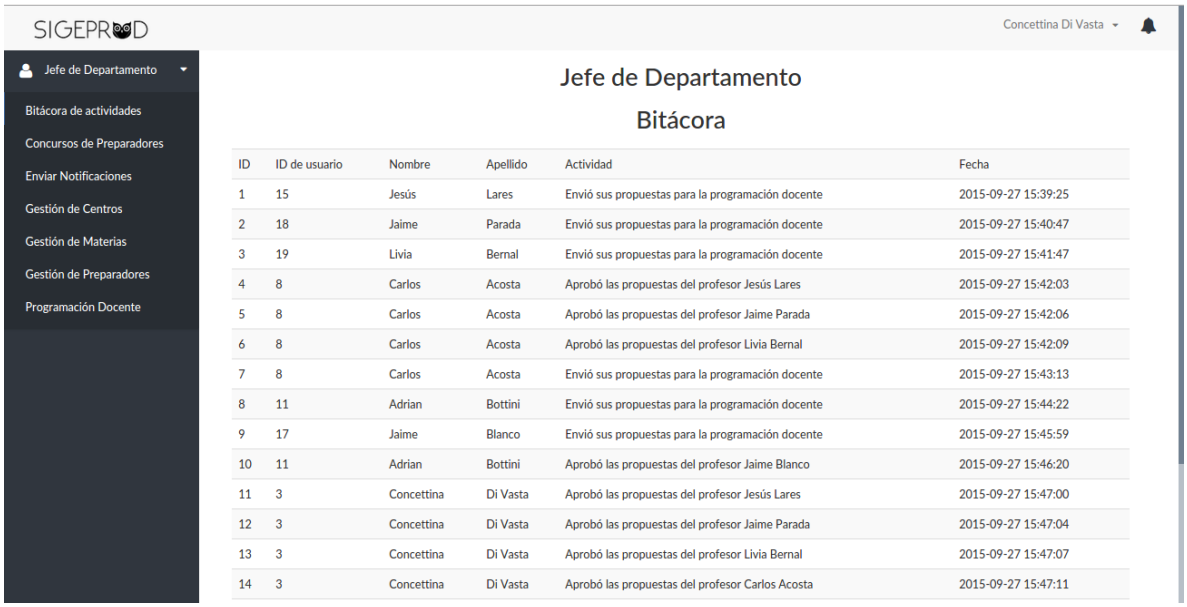

Figura 22: Interfaz de bitácora de actividades

La figura 22 muestra la interfaz de la Bitácora del sistema, en la cual se pueden ver las diferentes actividades que se han realizado, el usuario y la fecha en que se ejecutó.

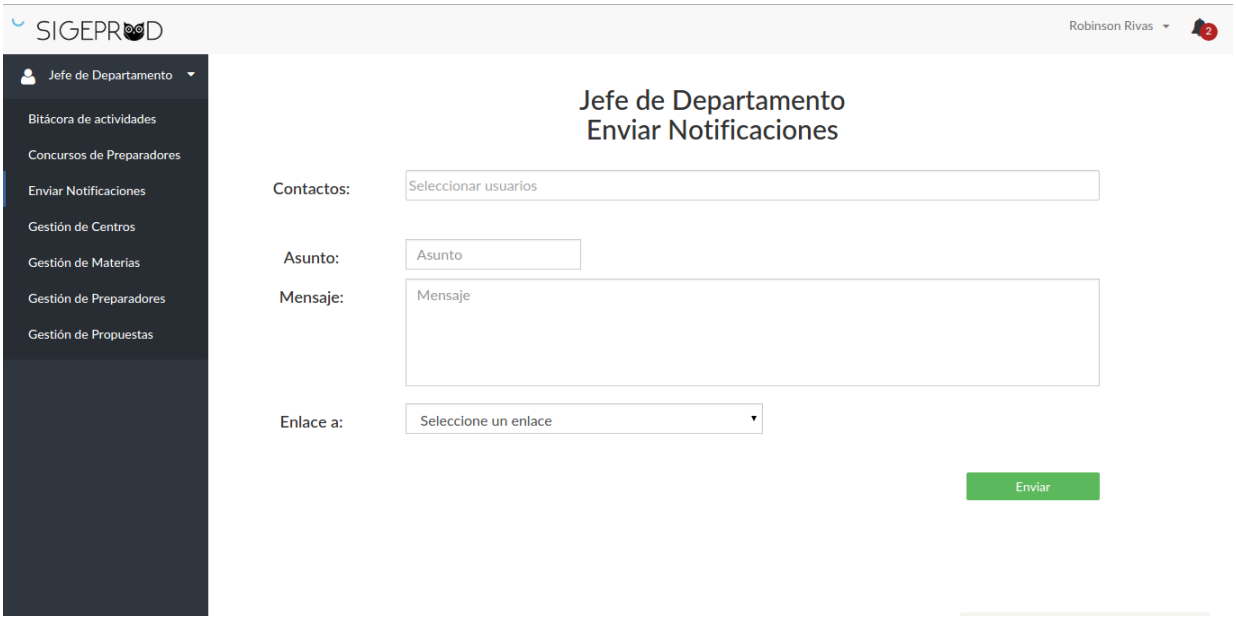

Figura 23: Interfaz de envío de notificaciones masivas

Esta figura (#13) muestra el formulario que deben llenar los usuarios para el envío de notificaciones masivas. Deben colocar los contactos a quien desean que se envíe el mensaje, el asunto, el contenido del mensaje y el enlace al que deben acceder los usuarios.

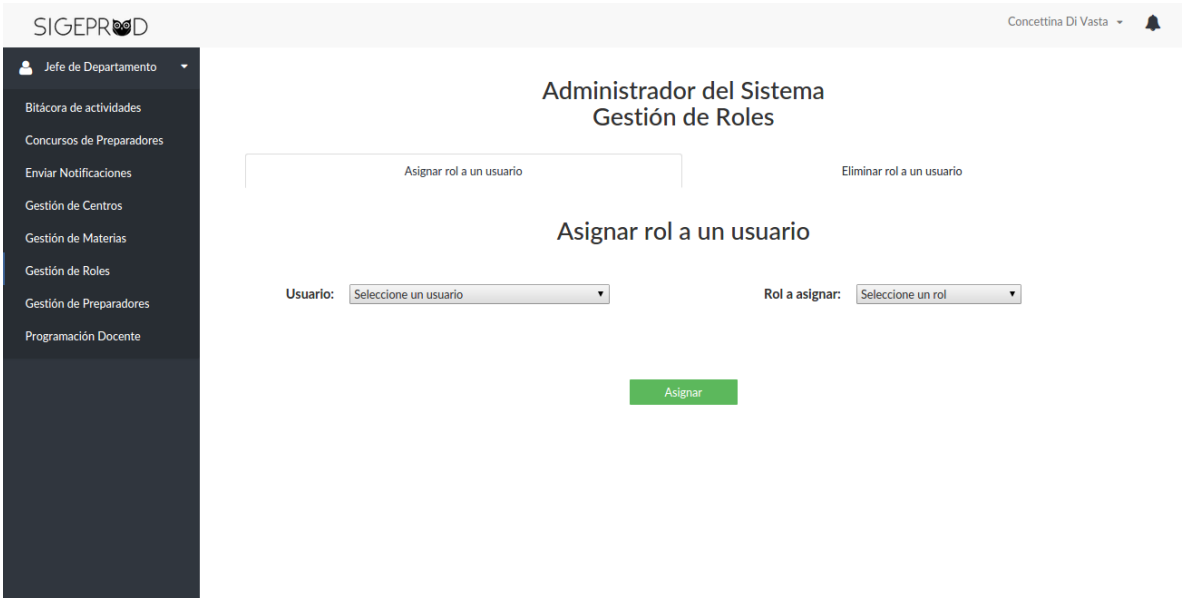

Figura 24: Interfaz de asignación de rol a un usuario

En la figura 24 podemos visualizar la funcionalidad de asignación de roles, donde se debe seleccionar el usuario y el rol a asignar y posteriormente hacer clic en el botón "Asignar". Si el rol es Coordinador de Centro se debe seleccionar además el Centro que va a coordinar, si es Coordinador de Materia se debe seleccionar la materia que va a Coordinar y si el rol es Profesor se debe seleccionar además el centro al que pertenece y la dedicación.

Figura 25: Interfaz de eliminación de rol a un usuario

La figura 25 muestra la funcionalidad de eliminación de roles, donde se debe seleccionar el usuario, el rol que se desea eliminar y finalmente hacer clic en el botón “Eliminar”.

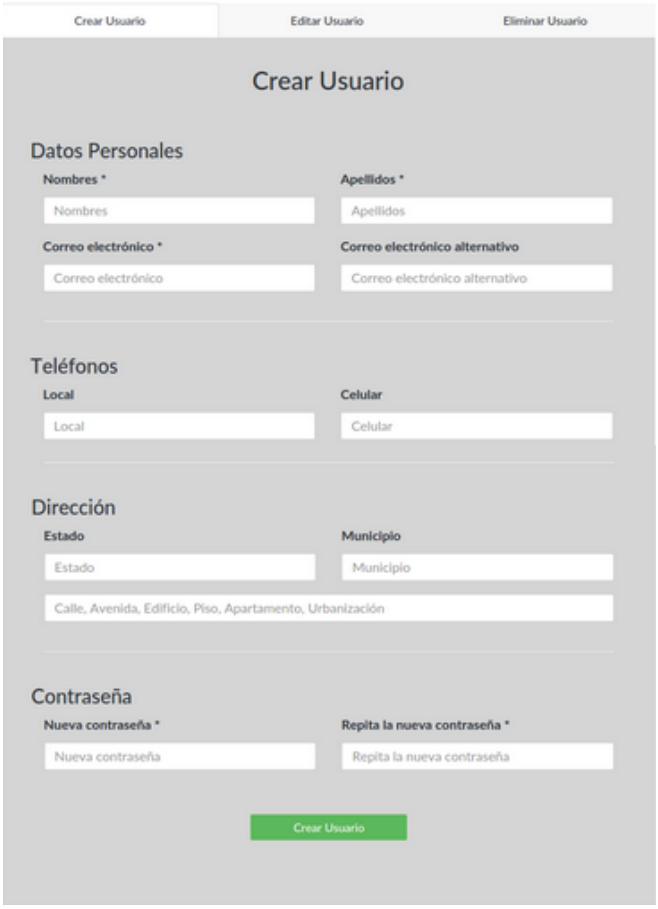

Figura 26: Interfaz de creación de usuario

En la figura 26 podemos ver la interfaz para la creación de un usuario nuevo en el sistema. La información mínima requerida para la creación de un usuario es nombre, apellido, correo electrónico, la contraseña y su confirmación, el resto es información no menos importante pero que puede ser agregada posteriormente.

Figura 27: Interfaz de edición de usuario

Como podemos ver la figura 27 muestra el formulario para la modificación de los datos referentes a los usuarios como nombre, apellido, teléfonos, dirección, entre otros. Inicialmente el Administrador del Sistema selecciona el usuario, luego modifica sus datos y finalmente hace clic sobre el botón “Actualizar datos”.

Figura 28: Interfaz de eliminación de usuario

La figura 28 muestra la funcionalidad para eliminar usuarios del sistema, correspondiente al rol de Administrador del Sistema, donde solo se debe seleccionar el usuario y hacer clic sobre el botón “Eliminar Usuario”.

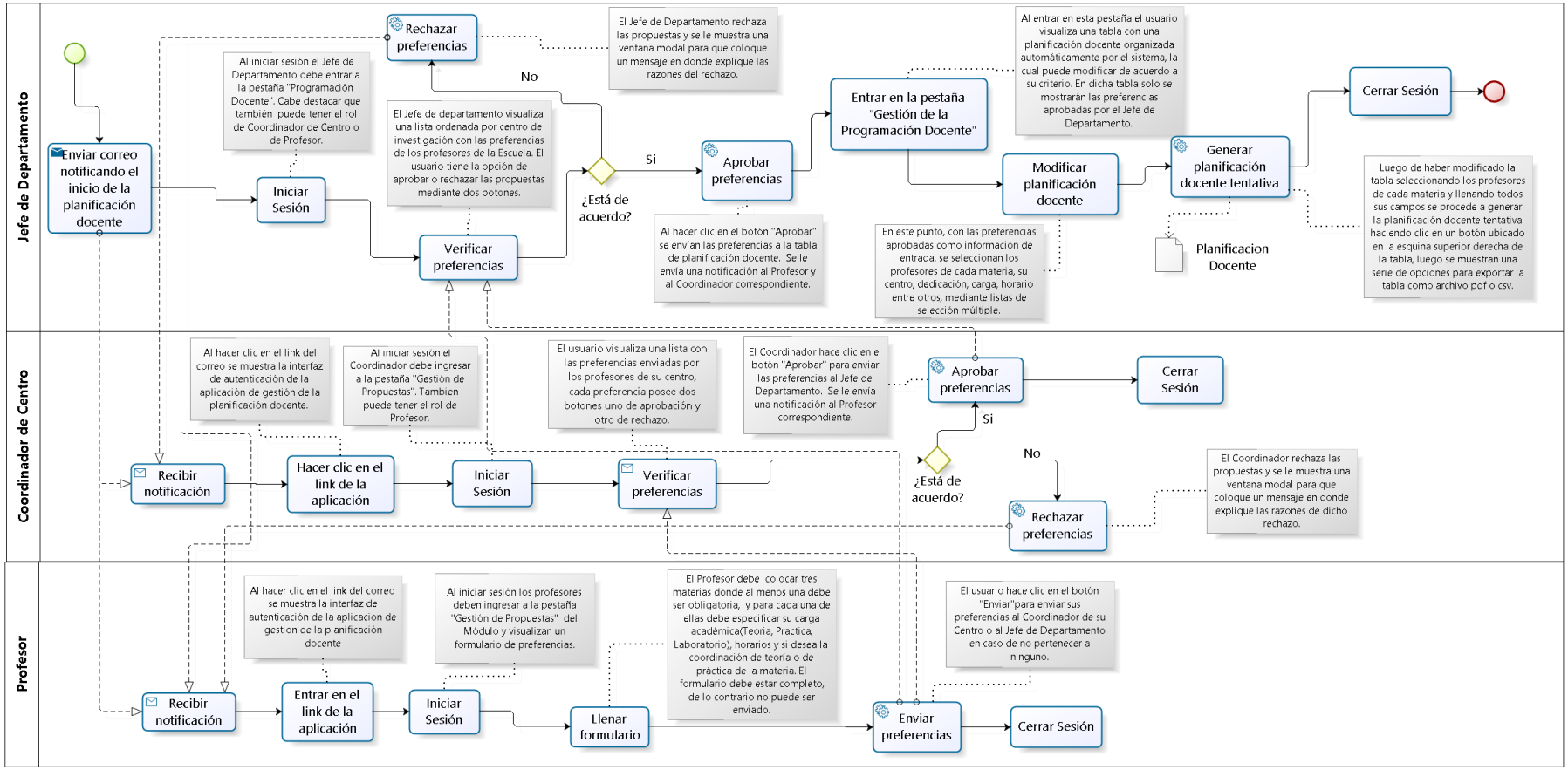

Figura 29 - Diagrama BPM - Gestión de la Programación Docente

La figura 29 muestra el diagrama BPM (Gestión de Procesos de Negocio), el cual describe el flujo de actividades que sigue el proceso de gestión de la Programación Docente.