Universidad de las Ciencias Informáticas

Facultad 15

Título: Sistema de Gestión de Cuidados de Examen

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autores: Alfredo Matamoros Ruiz

Alberto Ramírez Torres

Tutor:

MSc. Yenin Calderín Abad

Ciudad de La Habana, Junio 2010

Es justamente la posibilidad de realizar un sueño lo que hace que la vida sea interesante.

Paulo Coelho

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los _ días del mes de junio del año 2010.

_____ _____

Alfredo Matamoros Ruiz

Firma del Autor

Alberto Ramírez Torres

Firma del Autor

_____

MSc Yenin Calderín Abad

Firma del Tutor

DEDICATORIA

Alberto:

En especial a mis queridos padres, por su incondicional e infinito amor, por la confianza que siempre depositaron en mí, por inspirarme en todo momento y guiarme a ser todo lo que soy, por no defraudarme nunca, por respetarme siempre, por quererlos tanto. A mi hermano por estar ahí siempre para mí y ser un guía, en si ,el mejor hermano mayor. A mis sobrinitos por los momentos tan felices que nos han dado en la casa. A mis abuelos tanto paternos como maternos que tanto han hecho por mí y por los míos. A mi novia linda. A mi familia en general. A todos los que siempre han creído en mí.

Alfredo:

Quiero dedicársela en especial a mi abuelo que se que estaría orgulloso de mi, a mis padres por su fiel apoyo y amor, a mi hermano que espero que le sirva de guía, a toda mi familia en general, a mis amigos y a mi querida novia.

AGRADECIMIENTOS

Alberto:

A todos los amigos que tuve en todos estos largos cinco años en esta Universidad, por estar "en las verdes y en las maduras" a mi lado. A mi compañero de tesis Alfredo por creer en mi cuando todo se tornaba incierto. A Javier por enseñarnos y ayudarnos cuando más lo necesitábamos. A nuestra tutora Yenin y a nuestra amiga más que profesora Yaniselis por ayudarnos mucho. A mis amigos del "Barrio Fino", todos son mis hermanos y lo saben por creer siempre en mi. A todos los que de una forma u otra creen y me tienen como un verdadero amigo como yo a ellos. A todos los que se lo merecen en general: Gracias de todo corazón.

Alfredo:

Le agradezco a toda mi familia por apoyarme siempre, a mis padres, en especial a Titi, sin ti no se lo que hubiera sido de mi; a mi mamá gracias por creer siempre en mi, le agradezco siempre a mi hermano, eres imprescindible en mi vida. Quiero agradecerle con todo mi corazón a mi novia Massiel, eres lo mejor que me a pasado, así como a sus padres que me han apoyado todo este tiempo, a mis amigos del alma. También le agradezco a mi compañero de tesis Alberto por soportarme todo este tiempo, te quiero mi hermano, gracias a todos.

RESUMEN

El Sistema Informático para la Gestión de Cuidados de Examen en la Facultad 15 de la Universidad de las Ciencias Informáticas (UCI), surge debido a la necesidad de gestionar de manera más rápida, segura y eficaz los procesos que se desarrollan actualmente relacionados con el cuidado de exámenes y la planificación de los profesores para dicho cuidado. Este sistema informático desarrollado soluciona los problemas existentes en la gestión de cuidados de examen, que actualmente se desarrolla de forma manual imposibilitando la realización de reportes y actualización de la información con la calidad y rapidez que se necesita ya que se deben revisar varios documentos, lo que hace el proceso muy engorroso. Además existen muchos problemas de comunicación entre el encargado de la planificación y los responsables de llevar a cabo el cuidado de examen, etc. Se esperan como resultados más relevantes la implantación del sistema de forma satisfactoria y las mejoras en la gestión de cuidados de examen en la Facultad 15 de la UCI.

TABLA DE CONTENIDO

INTRODUCCIÓN 9
CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA 12
1.1 Introducción 12
1.2 Análisis de otras soluciones existentes 12
1.3 Tendencias y tecnologías actuales 13
1.3.1 Metodologías para el desarrollo de software 14
1.3.2 Lenguaje y herramientas para el modelado de software 17
1.3.3 Sistemas Gestores de Bases de Datos 19
1.3.4 Servidores Web 20
1.3.5 Lenguajes de Programación del lado del Cliente 21
1.3.6 Lenguajes de Programación del lado del Servidor 21
1.3.7 Entorno Integrado de Desarrollo 22
1.3.8 Framework 24
1.3.9 Otras herramientas utilizadas 25
1.4 Fundamentación metodológica 26
1.5 Conclusiones 27
CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA 29
2.1 Introducción 29
2.2 Modelo del negocio actual 29
2.2.1 Planificación del cuidado de exámenes 29
2.3 Análisis crítico de ejecución de los procesos actuales 30
2.4 Objeto de automatización 30
2.5 Actores del negocio 31
2.6 Trabajadores del negocio 31
2.7 Diagrama de casos de uso del negocio 32
2.8 Descripción de los casos de uso del negocio 32
2.9 Diagrama de actividades 33
2.10 Modelo de objetos del negocio. 34
2.11 Especificación de los requisitos de software 35
2.11.1 Definición de los requisitos funcionales 35
2.11.2 Definición de los requisitos no funcionales 36
2.12 Actores del sistema 38
2.12.1 Descripción de los actores del sistema 38
2.13 Diagrama de casos de uso del sistema 39
2.14 Descripción de los casos de usos del sistema 39
2.15 Conclusiones 42
CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA 43
3.1 Introducción 43
3.2 Modelo de análisis 43
3.2.1 Diagramas de Clases de Análisis 43

3.2.2 Diagramas de interacción 44
3.3 Modelo de Diseño 45
3.3.1. Diagramas de Clases de Diseño 45
3.4 Descripción de las tablas de Modelo de Datos 46
3.5 Diseño de la Base de Datos 47
3.5.1 Modelo Lógico de datos 48
3.5.2 Modelo Físico de datos 49
3.6 Seguridad 49
3.7 Conclusiones 50
CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA 51
4.1 Introducción 51
4.2 Implementación 51
4.2.1 Descripción de la funcionalidad Generar Distribución 52
4.2.2 Modelo de Despliegue 53
4.3 Modelo de Prueba 54
4.3.1 Métodos de Prueba 54
4.4 Casos de pruebas de la aplicación 55
4.4.1 Caso de prueba para la funcionalidad Adicionar Profesor 55
4.4.2 Caso de prueba para la funcionalidad Adicionar Planificación 56
4.4.3 Caso de prueba para la funcionalidad Modificar Planificación 58
4.5 Conclusiones 60
CONCLUSIONES GENERALES 61
RECOMENDACIONES 62
REFERENCIAS BIBLIOGRÁFICAS 63
BIBLIOGRAFÍA 64
GLOSARIO 65
ANEXOS 67
Anexo 1. Descripción de los principales casos de uso del sistema 67
Anexo 2. Diagramas de Clases de Análisis para Casos de Usos 73
Anexo 3. Diagramas de interacción (Diagramas de Secuencia) 74
Anexo 4. Diagramas de Clases de Diseño 76
Anexo 5. Descripción de las tablas de Modelo de Datos 78

INTRODUCCIÓN

Actualmente, una parte esencial para el desarrollo de una nación es lograr un avance en las tecnologías de la informática y las comunicaciones, un país encaminado a obtener dichos resultados puede alcanzar altos niveles de eficiencia y productividad en cuanto a su economía.

Nuestro país también necesita de estos avances, pero primero se debe lograr un incremento en el desarrollo del software que le permita obtener un elevado porciento de informatización, así como la exportación de estos, todo lo cual favorecerá su inclusión con un mayor impacto en el mercado internacional. La Universidad de las Ciencias Informáticas (UCI) es puntera en esta labor, donde estudiantes y profesores juegan un papel fundamental, al propiciar un continuo aporte a los diferentes proyectos productivos desarrollados en la misma; para mantener estos resultados es necesario un estricto cumplimiento de los deberes de todos los que en ella laboran.

Una parte esencial del plan de estudio de los estudiantes son los exámenes, estos se realizan a lo largo de todo el curso donde los profesores juegan un papel fundamental en sus desarrollos, al ser los encargados de velar por su correcto cumplimiento. En la medida en que se han aplicado dichos exámenes han tenido lugar las siguientes dificultades:

- Problemas de comunicación entre los responsables de llevar a cabo el cuidado del examen y el encargado de la planificación.
- Muchas veces a los profesores se les informa por vías no formales que les corresponde cuidar el examen, o les llega tarde la información.
- El proceso de asignación de profesores para el cuidado de exámenes se realiza de forma manual , donde se deben revisar varios documentos que hacen el proceso muy engorroso.
- En reiteradas ocasiones no se equilibra la asignación de cuidados de exámenes, lo que provoca que algunos profesores se sobrecarguen más que otros.
- En ocasiones, un solo profesor se encarga del cuidado de los exámenes finales de disciplina que tiene una duración de 4 horas.

A raíz de esta problemática se plantea el siguiente **problema a resolver** ¿Cómo viabilizar el proceso de gestión de los cuidados de exámenes en la Facultad 15 de la Universidad de Las Ciencias Informáticas?

Donde el **objeto de investigación** son los procesos para el control de las tareas asignadas a los profesores en la Facultad 15 de la Universidad de Las Ciencias Informáticas y el **campo de acción** es el proceso de gestión de cuidado de exámenes en la Facultad 15 de la Universidad de Las Ciencias Informáticas.

El **objetivo general** de esta investigación es "Desarrollar un sistema informático que viabilice el proceso de gestión de cuidados de exámenes en la Facultad 15 de la Universidad de Las Ciencias Informáticas*".*

Como **hipótesis** tenemos que "*si se desarrolla un sistema informático para la gestión de cuidado de exámenes en la Facultad 15 de la Universidad de Las Ciencias Informáticas entonces se viabilizará la ejecución de dicho proceso".*

De la hipótesis anterior se derivan las siguientes variables.

Variable Independiente: Desarrollo de un Sistema Informático, la gestión de cuidado de exámenes.

Variable Dependiente: Viabilizar el proceso de gestión de cuidado de exámenes.

Las **tareas de la investigación** trazadas para darle cumplimiento al objetivo general son:

- Valoración de las tendencias actuales de aplicaciones o soluciones informáticas similares en el mundo, Cuba y la universidad para el desarrollo del software.
- Identificación de los requisitos funcionales y no funcionales que debe cumplir el sistema informático de gestión de cuidado de exámenes para describir las necesidades y funcionalidades del sistema a desarrollar.
- Obtención de los modelos de análisis y diseño del sistema informático para lograr una mejor implementación de dicho sistema.
- Implementación del sistema informático a partir del modelo de análisis y diseño para obtener el software final.
- Selección de las herramientas para la implementación del sistema informático.
- Evaluación de la mejora del proceso de gestión de cuidado de exámenes de la Facultad 15, a partir de la puesta en funcionamiento del sistema informático, para verificar su correspondencia con los objetivos planteados.

Resultados esperados: Con la elaboración de este trabajo se pretende dotar a la Facultad 15 de una herramienta informática gratuita y de libre distribución, que se adapte a las necesidades de la misma.

El presente trabajo está estructurado por cuatro capítulos, tal y como se muestra a continuación:

Capítulo 1. Fundamentación teórica: Se realiza un estudio del estado del arte. Se describen las herramientas, tecnologías y metodología escogidas para la solución del problema.

Capítulo 2. Características del sistema: Se realiza la modelación del negocio. Se describe formalmente el proceso de gestión de cuidados de exámenes, especificando para cada actividad el responsable. Se enumeran las funcionalidades y los atributos del sistema informático. Por último se definen los casos de uso del sistema.

Capítulo 3. Análisis y diseño del sistema: Se presenta el modelo de análisis, que incluye los diagramas de clases del análisis, los diagramas de colaboración para cada caso de uso y diagrama de clases de diseño web por casos de uso. Por último, se presenta el modelo de diseño de la base de datos con el diagrama entidad de la base de datos y los diagramas de despliegue e implementación, concluyendo la programación del sistema informático.

Capítulo 4. Implementación y prueba: Contiene el diagrama de despliegue, así como una breve descripción de estos conceptos. Se reflejan las pruebas de caja blanca y caja negra que certifican la calidad del sistema informático propuesto.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se aborda el estudio de sistemas similares al que se pretende desarrollar en el ámbito nacional e internacional. Se realiza además un estudio de las metodologías de desarrollo de software y las características de las principales tecnologías y herramientas existentes utilizadas para el desarrollo de este tipo de aplicaciones.

1.2 Análisis de otras soluciones existentes

En la actualidad son múltiples las soluciones que de una forma u otra se relacionan con el tema del presente trabajo, a continuación se reflejan las más similares en este aspecto:

SGD WEB, del centro de demostración TECNAUSA, el cual permite acceder a los horarios, faltas y notas, además de permitir la comunicación entre padres, profesores y alumnos desde casa, sin necesidad de visitar el centro [1]. Consiste en un paquete completo de Software que se instala en un ordenador del centro y se integra con el programa de gestión del mismo. Almacena y procesa los datos captados desde la Unidad Central, la PDA (Personal Digital Assistant) o PC (Personal Computer) por aula simultáneamente, permitiendo al profesorado realizar cualquier tipo de consulta o modificación. Este software se distribuye bajo licencia propietaria.

La empresa española Peñalara mantiene el software **Generador de Horarios para Centros de Enseñanza**, actualmente en la versión 2009. Este software provee mecanismos de intercambio de datos con las principales aplicaciones de gestión académica, posee un potente motor de generación de horarios que obtiene resultados optimizados a costa de un consumo de tiempo elevado durante el proceso de generación de los horarios. Se distribuye bajo licencia propietaria y en su versión gratuita sólo permite modelar centros académicos con menos de 10 profesores, únicamente puede ser ejecutado sobre plataforma Windows y es poco flexible para modelar organizaciones con una estructura diferente a la concebida por sus autores [2].

Spiral Universe es un sistema de administración de escuelas basado en web, creado especialmente para escuelas primarias y secundarias experimentales, públicas y privadas y distritos alrededor del mundo. Como Spiral es una aplicación enteramente basada en web, se puede acceder a la información en cualquier momento y desde cualquier lugar. Además, no depende de una plataforma, es decir, que las

escuelas no necesitan comprar ningún hardware ni instalar ningún software nuevo. Spiral funciona sin defectos en cualquier computadora que esté conectada a Internet, ya sea una MAC (Media Access Control) o una PC. Posee un registro y alcance de características sólidas sin precedente, una interface fácil de usar y el compromiso de brindar capacitación y mantenimiento personalizados se combinan para garantizarle a las escuelas el acceso a las últimas tecnologías web.

En la UCI se han realizado varias investigaciones en este campo. En los años 2006 y 2007 se desarrollaron importantes trabajos de diploma titulados **"Sistema para la gestión de horarios docentes"** y **"Asistente de ayuda para la confección de horarios docentes"** respectivamente. Estos trabajos no concluyeron con la realización de un software funcional limitándose al análisis y el diseño.

Del estudio realizado únicamente se detectaron los sistemas y trabajos mencionados, los cuales presentan cierta relación con el objeto de estudio de este trabajo, sin embargo, ninguna de estas soluciones incluye funcionalidades inherentes a la gestión de la planificación de cuidado de exámenes, ni la información de los profesores relacionada con este proceso.

1.3 Tendencias y tecnologías actuales

El entorno donde se utilizará la presente propuesta de solución es un factor importante a tener en cuenta a la hora de seleccionar las mejores opciones para su desarrollo y puesta en funcionamiento. A continuación se muestra el estudio realizado para seleccionar las tecnologías y herramientas que serán utilizadas las cuales serán de software libre, excepto Visual Paradigm 6.1 Enterprise Edition que la universidad posee la licencia de la misma. Se decide emplear dichas herramientas y tecnologías porque el país y la Universidad de las Ciencias Informáticas tiene como política la tendencia de emigrar del sofware propietario al sofware libre debido a las ventajas que este proporciona como por ejemplo es mucho más económico, la libertad de uso y redistribución porque las licencias de software libre existentes permiten la instalación del software tantas veces y en tantas máquinas como el usuario desee, la independencia tecnológica porque el acceso al código fuente permite el desarrollo de nuevos productos sin la necesidad de desarrollar todo el proceso partiendo de cero, el soporte y compatibilidad a largo plazo, los sistemas no tienen puertas traseras y son más seguros, poseen una corrección más rápida y eficiente de fallos porque el funcionamiento e interés conjunto de la comunidad ha demostrado solucionar más rápidamente los fallos de seguridad en el software libre, algo que desgraciadamente en el software propietario es más difícil y costoso, el sistema en expansión porque el software libre ya no es una promesa, es una realidad y se utiliza en sistemas de

producción por algunas de las empresas tecnológicas más importantes como IBM, SUN Microsystems, Google, Hewlett-Packard, etc.

1.3.1 Metodologías para el desarrollo de software

Se entiende por metodología de desarrollo una colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software. La finalidad de una metodología de desarrollo es garantizar la eficacia y la eficiencia en el proceso de generación de software. Los riesgos a afrontar y los controles a establecer, varían en función de las diferentes etapas del ciclo de vida de desarrollo. En la actualidad se han desarrollado dos tipos de métodos en el desarrollo del software que son los llamados métodos ligeros o ágiles y métodos pesados. La diferencia fundamental entre ambos es que mientras los métodos pesados intentan lograr la calidad del software mediante el orden y la documentación, los métodos ágiles lo hacen por medio de una comunicación directa entre las personas que intervienen en el proceso. Dentro de estas metodologías las más populares y mundialmente utilizadas son: Las metodologías pesadas: MÉTRICA 3 y Rational Unified Process (RUP) y la metodología ágil: Extreme Programming (XP).

1.3.1.1 Metodologías pesadas o altamente predictivas

Métrica V.3

La metodología MÉTRICA Versión 3 ofrece a las Organizaciones un instrumento útil para la sistematización de las actividades que dan soporte al ciclo de vida del software dentro del marco que permite alcanzar los siguientes objetivos:

- Proporcionar o definir Sistemas de Información que ayuden a conseguir los fines de la Organización mediante la definición de un marco estratégico para el desarrollo de los mismos.
- Dotar a la Organización de productos de software que satisfagan las necesidades de los usuarios dando una mayor importancia al análisis de requisitos.
- Mejorar la productividad de los departamentos de Sistemas y Tecnologías de la Información y las Comunicaciones, permitiendo una mayor capacidad de adaptación a los cambios y teniendo en cuenta la reutilización en la medida de lo posible.

- Facilitar la comunicación y entendimiento entre los distintos participantes en la producción de software a lo largo del ciclo de vida del proyecto, teniendo en cuenta su papel y responsabilidad, así como las necesidades de todos y cada uno de ellos.
- Facilitar la operación, mantenimiento y uso de los productos de software obtenidos.

MÉTRICA V.3 contempla el desarrollo de Sistemas de Información para las distintas tecnologías que actualmente están conviviendo y los aspectos de gestión que aseguran que un Proyecto cumple sus objetivos en términos de calidad, costo y plazos.

Proceso Unificado de Desarrollo

El Proceso Unificado de Desarrollo (Rational Unified Process en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado (UML, conocido por sus siglas en inglés), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

El RUP está basado en 5 principios clave que son: adaptar el proceso, balancear prioridades, demostrar valor iterativamente, elevar el nivel de abstracción y enfocarse en la calidad. El ciclo de vida RUP es una implementación del desarrollo en espiral. Fue creado ensamblando los elementos en secuencias semi-ordenadas. El ciclo de vida organiza las tareas en fases e iteraciones. RUP divide el proceso en cuatro fases: Inicio, Elaboración, Construcción y Transición, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en los distintas actividades. Las primeras iteraciones (en las fases de Inicio y Elaboración) se enfocan hacia la comprensión del problema y la tecnología, la delimitación del ámbito del proyecto, la eliminación de los riesgos críticos, y al establecimiento de una línea base de la arquitectura. En la fase de construcción, se lleva a cabo la construcción del producto por medio de una serie de iteraciones. En la fase de transición se pretende garantizar que se tiene un producto preparado para su entrega a la comunidad de usuarios.

Principales características de RUP:

- Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo)
- Pretende implementar las mejores prácticas en Ingeniería de Software
- Desarrollo iterativo

- Administración de requisitos
- Uso de arquitectura basada en componentes
- Control de cambios
- Modelado visual del software
- Verificación de la calidad del software

El RUP se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos (que son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso).

1.3.1.2 Metodologías ligeras

Scrum

Es un proceso ágil y liviano que sirve para administrar y controlar el desarrollo de software. El desarrollo se realiza en forma iterativa e incremental (una iteración es un ciclo corto de construcción repetitivo). Cada ciclo o iteración termina con una pieza de software ejecutable que incorpora nuevas funcionalidades. Las iteraciones en general tienen una duración entre 2 y 4 semanas. Scrum se utiliza como marco para otras prácticas de ingeniería de software como RUP o Extreme Programming.

Scrum se enfoca en priorizar el trabajo en función del valor que tenga para el negocio, maximizando la utilidad de lo que se construye y el retorno de inversión. Está diseñado especialmente para adaptarse a los cambios en los requerimientos, por ejemplo en un mercado de alta competitividad. Los requerimientos y las prioridades se revisan y ajustan durante el proyecto en intervalos muy cortos y regulares. De esta forma se puede adaptar en tiempo real el producto que se está construyendo a las necesidades del cliente. Se busca entregar un software que realmente resuelva las necesidades, aumentando la satisfacción del cliente [3].

Extreme Programming

Extreme Programming (XP) es la más destacada entre los procesos ágiles de desarrollo de software formulada por Kent Beck. La programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad.

Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto, es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos [3].

1.3.2 Lenguaje y herramientas para el modelado de software

En todas las disciplinas de la Ingeniería se hace evidente la importancia de los modelos ya que describen el aspecto y la conducta de "algo". Ese "algo" puede existir, estar en un estado de desarrollo o estar, todavía, en un estado de planeación. Es en este momento cuando los diseñadores del modelo deben investigar los requerimientos del producto terminado y dichos requerimientos pueden incluir áreas tales como funcionalidad, performance y confiabilidad. Además, a menudo, el modelo es dividido en un número de vistas, cada una de las cuales describe un aspecto específico del producto o sistema en construcción.

El modelado sirve no solamente para los grandes sistemas, aún en aplicaciones de pequeño tamaño se obtienen beneficios de modelado, sin embargo es un hecho que entre más grande y más complejo es el sistema informático, más importante es el papel que juega el modelado.

1.3.2.1 Lenguaje Unificado de Modelado

El Lenguaje Unificado de Modelado (UML en sus siglas en inglés) es una técnica para la especificación de sistemas informáticos en todas sus fases. La versión 1.0 de UML fue liberada en enero de 1997 y ha sido utilizado con éxito en sistemas construidos para toda clase de industrias alrededor del mundo.

Los principales beneficios de UML son:

- Mejores tiempos totales de desarrollo (de 50 % o más).
- Modelar sistemas (y no sólo de software) utilizando conceptos orientados a objetos.
- Establecer conceptos y artefactos ejecutables.

- Encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica.
- Crear un lenguaje de modelado utilizado tanto por humanos como por máquinas.
- Mejor soporte a la planeación y al control de proyectos.
- Alta reutilización y minimización de costos.

UML ofrece una amplia variedad de diagramas que permiten visualizar el sistema desde varias perspectivas, estos son: Diagrama de Casos de Uso, Diagrama de Clases, Diagrama de Objetos, Diagrama de Secuencia, Diagrama de Colaboración, Diagrama de Estados, Diagrama de Actividad, Diagrama de Componentes, Diagrama de Despliegue. Para modelar el comportamiento dinámico del sistema son utilizados los diagramas de actividad, secuencia y colaboración.

1.3.2.2 Herramientas CASE

Cuando se habla de herramientas de modelado en la disciplina de Ingeniería de Software, es importante mencionar las herramientas CASE (Computer Aided Software Engineering). Estas están tomando cada vez más relevancia en la planeación y ejecución de proyectos que involucren sistemas de información, pues suelen inducir a sus usuarios a la correcta utilización de metodologías que le ayuden a llegar con facilidad a los productos de software construidos. CASE proporciona al ingeniero la posibilidad de automatizar actividades manuales y de mejorar su visión general de la ingeniería. Las herramientas CASE ayudan a garantizar que la calidad se diseñe antes de llegar a construir el producto [4].

Rational Rose

Esta herramienta CASE da soporte al modelado visual con UML cubriendo todo el ciclo de vida de un proyecto. Se enmarca dentro del desarrollo de modelado para fines académicos, investigativos y comerciales. Permite la autogeneración de código a partir de modelos y viceversa para lenguajes como Java y realizar ingeniería inversa en: Visual Basic, C++, etc.

Está dotado de un navegador que muestra información de forma jerárquica de todos los elementos de los modelos de un proyecto. Presenta varias opciones para el manejo de los diagramas en el momento de su visualización, exportación o impresión.

Visual Paradigm para UML

Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML [3].

1.3.3 Sistemas Gestores de Bases de Datos

Un Sistema Gestor de Base de Datos es un conjunto de programas que permiten crear y mantener una Base de datos, asegurando su integridad, confidencialidad y seguridad. Por tanto debe permitir:

1. Definir una Base de Datos: especificar tipos, estructuras y restricciones de datos.
2. Construir la Base de Datos: guardar los datos en algún medio controlado por el mismo *SGBD*.
3. Manipular la Base de Datos: realizar consultas, actualizarla, y generar informes.
4. Redundancia mínima: un diseño correcto de una Base de Datos debe evitar la duplicación o redundancia de la información.
5. Independencia: debe ser posible modificar la Base de Datos sin necesidad de realizar cambios en las aplicaciones que la accedan.

MySQL

MySQL es un sistema gestor de bases de datos relacionales multihilo y multiusuario, cuyo principal objetivo de diseño fue la velocidad (ofrece alto rendimiento debido a su rapidez de respuesta). Sus creadores sacrificaron algunas características, como verificación de integridad y uso de disparadores, a cambio de ganar en velocidad. Provee múltiples motores de almacenamiento. Posibilita conexiones entre diferentes computadoras con distintos sistemas operativos y una integración perfecta con PHP. Este gestor de bases de datos es muy usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Su gran aceptación es debida, en parte, a que existe un gran número de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración.

PostgreSQL

PostgreSQL es un sistema gestor de bases de datos relacional basada en objetos de código abierto. Actualmente es totalmente compatible con el lenguaje estándar de consultas (ANSI SQL 92), incluyendo selección anidada (subselects) y llaves foráneas (foreign keys), es un potente sistema gestor que posee una gran escalabilidad, pues es capaz de ajustarse al número de procesadores y a la cantidad de memoria que posee el sistema de forma óptima, soportando una mayor cantidad de peticiones simultáneas de manera correcta. Implementa el uso de deshacer los cambios (rollbacks), subconsultas y transacciones, haciendo su funcionamiento mucho más eficiente y tiene la capacidad de comprobar la integridad referencial, además de almacenar procedimientos en la propia base de datos. Es también, un sistema multiusuario, multiprogramado, con arquitectura cliente-servidor y control de privilegios de acceso [5].

Oracle

Oracle es un sistema de base de datos relacional extremadamente potente y flexible. Esta potencia y flexibilidad, sin embargo, implican también una cierta complejidad. Para poder diseñar aplicaciones útiles basadas en Oracle es necesario entender cómo manipula los datos almacenados en el sistema. PL/SQL es una herramienta de gran importancia diseñada para la manipulación de datos, tanto internamente dentro de Oracle como externamente, en las propias aplicaciones. PL/SQL está disponible en diversos entornos, cada uno de los cuales tiene diferentes ventajas. Es una aplicación propietaria y sus precios son muy altos en el mercado [6].

1.3.4 Servidores Web

Un servidor Web no es más que un programa que se ejecuta continuamente en una computadora (también llamada por lo general servidor) que interpreta las peticiones HTTP (HyperText Transfer Protocol) que recibe por parte de un cliente (un navegador Web) y las satisface. Dependiendo del tipo de la petición, el servidor Web buscará una página Web o bien ejecutará un programa en el servidor, devolviendo algún tipo de resultado HTML (HyperText Markup Language) al cliente o navegador que realizó la petición.

Internet Information Server

Es la solución de Microsoft a las necesidades de las empresas y usuarios de enviar y recibir la información no sólo de sus clientes si no también en el entorno de la empresa. Este servidor web engloba una serie de herramientas administrativas que le permitirán controlar sitios web, FTP (File Transfer Protocol), SMTP (Simple Mail Transfer Protocol) y servicio de noticias. Dispone también del soporte necesario para crear páginas dinámicas (ASP y ASP.NET), lenguaje de aplicaciones para Internet bastante extendido permitiendo la conexión y acceso a bases de datos, consiguiendo realizar aplicaciones web dinámicas y escalables [7].

Apache

Apache es un servidor web que corre sobre los sistemas operativos más utilizados (Unix, Linux, Windows), lo que lo hace prácticamente universal. Es gratuito y su código es abierto, esto lo hace transparente de manera que puede ser estudiado, sin ningún secreto o puerta trasera que nos sorprenda. Es un servidor altamente configurable de diseño modular. Actualmente existe gran cantidad de módulos para Apache que son adaptables a este, y pueden ser instalados cuando los necesitemos. Con una cierta experiencia en programación en Perl o C se pueden construir módulos personalizados para realizar funciones específicas. Apache permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurarlo para que ejecute un determinado script cuando ocurra un error en concreto. Es altamente configurable en la creación y gestión de logs. Apache permite la creación de ficheros de log a medida del administrador, de este modo se puede tener un mayor control sobre lo que sucede en el servidor.

1.3.5 Lenguajes de Programación del lado del Cliente

JavaScript

JavaScript es un lenguaje de scripting basado en objetos, utilizado para acceder a objetos en aplicaciones. Principalmente, se utiliza integrado en un navegador web permitiendo el desarrollo de interfaces de usuario mejoradas y páginas web dinámicas. Todos los navegadores modernos interpretan el código JavaScript integrado dentro de las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM.

1.3.6 Lenguajes de Programación del lado del Servidor

ASP.NET

ASP.NET es un marco de trabajo de programación generado en Common Language Runtime que puede utilizarse en un servidor para generar eficaces aplicaciones Web. ASP.NET ofrece varias ventajas importantes acerca de los modelos de programación Web anteriores:

Mejor rendimiento. ASP.NET es un código compilado que se ejecuta en el servidor. A diferencia de sus predecesores, puede aprovechar las ventajas del enlace anticipado, la compilación en tiempo de ejecución

(just-in-time), la optimización nativa y los servicios de caché desde el primer momento. Esto supone un incremento espectacular del rendimiento antes de siquiera escribir una línea de código.

Compatibilidad con herramientas de primer nivel. Se complementa con un diseñador y un paquete de herramientas excelentemente diseñado en el entorno integrado de programación. Una de las características que proporciona esta herramienta es que presenta controles de servidor de arrastrar y colocar, además de la implementación automática.

PHP

Es un lenguaje de programación utilizado para la creación de sitio web, es un acrónimo recursivo que significa PHP Hypertext Pre-processor, (inicialmente se llamó Personal Home Page). Surgió en 1995, desarrollado por PHP Group. Es un lenguaje de script interpretado en el lado del servidor utilizado para la generación de páginas web dinámicas, embebidas en páginas HTML y ejecutadas en el servidor. PHP no necesita ser compilado para ejecutarse, es multiplataforma y funciona tanto para Unix como para Microsoft Windows, de manera que el código creado para Unix no sufre modificaciones al ser ejecutado en Windows y viceversa. Ofrece soluciones simples y universales para las paginaciones dinámicas de la Web mediante una fácil programación y su diseño elegante lo hace más fácil de mantener y actualizar, a diferencia del código en otros lenguajes, posee capacidad de conexión con la mayoría de los manejadores de base de datos. PHP es libre, por lo que es una alternativa de fácil acceso para todos.

1.3.7 Entorno Integrado de Desarrollo

A continuación se expondrán, las características fundamentales de los entornos de desarrollo integrado (IDE en sus siglas en inglés) Zend Studio y Eclipse, quienes presentan mayor potencialidad para el trabajo con la Web que otros entornos.

De manera general un entorno de desarrollo integrado o en inglés Integrated Development Environment, es un programa compuesto por un conjunto de herramientas para un programador [9]. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, puede utilizarse para varios.

Zend Estudio

Zend Estudio es un programa de la compañía Zend, impulsores de la tecnología de servidor PHP, orientada a desarrollar aplicaciones web, en lenguaje PHP. El programa, además de servir de editor de texto para páginas PHP, proporciona una serie de ayudas que pasan desde la creación y gestión de proyectos hasta la depuración de código.

El programa entero está escrito en Java, lo que a veces supone que no funcione tan rápido como otras aplicaciones de uso diario. Sin embargo, esto ha permitido a Zend lanzar con relativa facilidad y rapidez versiones del producto para Windows y Linux.

Si se desea aumentar la productividad en los desarrollos PHP no cabe duda que este programa puede resultar útil y prestar gran ayuda en la realización de los mismos ofreciendo un entorno agradable y funcionalidades que simplifican el trabajo y optimizan el resultado. Zend Studio incorpora suficientes ayudas como para que su utilización sea confiable y resulte idóneo para desarrollar aplicaciones web.

Eclipse

En la web oficial de eclipse, se define como "An IDE for everything and nothing in particular" (un IDE para todo y para nada en particular). Eclipse es en el fondo, únicamente un armazón sobre el que se puede montar herramientas de desarrollo para cualquier lenguaje, mediante la implementación de los plugins adecuados.

La arquitectura de plugins de Eclipse permite además de integrar diversos lenguajes sobre un mismo IDE, introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo como: herramientas UML, editores visuales de interfaces, ayuda en línea para librerías, entre otros. Existen versiones de Eclipse instalables para cualquier plataforma que incluyen el código fuente y los plugins más habituales.

Una de las características más curiosas del IDE Eclipse es el modo en que se compilan los proyectos. No existe en Eclipse ningún botón que permita compilar individualmente un fichero concreto. La compilación es una tarea que se lanza automáticamente al guardar los cambios realizados en el código.

La principal diferencia entre un simple editor y un buen entorno de desarrollo es que este se integre con una buena herramienta visual para depurar los programas escritos. Eclipse incluye un depurador potente, sencillo y muy cómodo de utilizar. Permite de una forma muy simple, generar automáticamente la documentación del propio proyecto.

Eclipse ofrece servicios que permiten el trabajo entre desarrolladores de forma más eficiente y sincronizada, pues cuenta con las Taks (tareas), que posibilitan una comunicación directa entre los miembros del equipo, permitiendo dar órdenes de trabajo de manera online en forma de correo electrónico. Esta opción hace de Eclipse un IDE ideal para el trabajo en equipo.

1.3.8 Framework

Un framework simplifica el desarrollo de una aplicación web, esto se logra a través de la automatización de algunos patrones utilizados para resolver tareas comunes, además proporciona estructura al código fuente y facilita la programación de aplicaciones. Se estudiarán dos de estos framework con el fin de escoger uno de ellos para utilizar en la construcción de la solución propuesta.

Symfony

Symfony Project es una plataforma de trabajo para construir aplicaciones web con PHP. En otras palabras, es un enorme conjunto de herramientas y utilidades que simplifican el desarrollo de las aplicaciones web. Symfony emplea el tradicional patrón de diseño MVC (modelo-vista-controlador) para separar las distintas partes que forman una aplicación web. El modelo representa la información con la que trabaja la aplicación y se encarga de acceder a los datos. La vista transforma la información obtenida por el modelo en las páginas web a las que acceden los usuarios. El controlador es el encargado de coordinar todos los demás elementos y transformar las peticiones del usuario en operaciones sobre el modelo y la vista.

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y Microsoft SQL Server. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows.

CodeIgniter

CodeIgniter es un marco de trabajo abierto que permite crear webs dinámicas con PHP. Su principal objetivo es ayudar a que los desarrolladores, puedan realizar proyectos mucho más rápido que creando toda la estructura desde cero. También hay que destacar que CodeIgniter es más rápido que muchos otros frameworks. Es basado en el Modelo-Vista-Controlador, es compatible con PHP 4, posee características completamente equipadas de la base de datos con la ayuda para varias plataformas. Cuenta con una biblioteca de manipulación de imágenes.

1.3.9 Otras herramientas utilizadas

ExtJS

En el mercado actualmente existen múltiples librerías de JavaScript que permiten realizar todo tipo de maravillas en el navegador web. Una de estas librerías, se llama ExtJS y tiene muchas cualidades que la hacen interesante. ExtJS es una librería JavaScript que permite construir aplicaciones complejas en Internet. Esta librería incluye:

- Componentes de interfaz de usuario de alto performance y personalizables.
- Modelo de componentes extensibles.
- Licencia de código abierto y comercial.

Principales características:

- Permite crear aplicaciones ricas en internet, mediante JavaScript, utilizando componentes predefinidos así como un manejador de layouts similar al que provee Java Swing.
- La ventana flotante que provee ExtJS es excelente por la forma en la que funciona.
- Existe un balance entre Cliente – Servidor.
- Comunicación asíncrona.
- Eficiencia de la red [10].

1.4 Fundamentación metodológica

Luego del estudio de las principales metodologías, herramientas y tecnologías que se usan a nivel global para el desarrollo de sistemas informáticos, específicamente en aplicaciones web, se concluye cuáles se adaptan a la solución del sistema informático, concluyendo que:

- Para guiar el proceso de desarrollo del software, se decide utilizar la metodología RUP porque constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso.
- Para el modelado del negocio, modelado del sistema y análisis y diseño del sistema se utiliza el lenguaje de modelado UML, por ser un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software, basado en la modelación de sistemas con tecnología orientada a objetos.
- Como herramienta CASE para realizar el modelado se decide utilizar Visual Paradigm 6.1 Enterprise Edition, la cual es la única herramienta propietaria a utilizar y se debe precisamente a los beneficios que propicia la misma. Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue .Su mayor éxito consiste en la capacidad de ejecutarse sobre diferentes sistemas operativos. Visual Paradigm utiliza UML como lenguaje de modelado ofreciendo soluciones de software que permiten a las organizaciones desarrollar las aplicaciones de calidad más rápido, bien y más barato. Es muy fácil de usar y presenta un ambiente gráfico agradable para el usuario. Su notación es muy parecida a la estándar, permite configurar las líneas de redacción, el modelado de base de datos, el modelado de requerimientos, el modelado del proceso de negocio, la interoperabilidad, la generación de documentación y la generación de código base para diferentes lenguajes de programación como Java, C# y PHP además de permitir la integración con herramientas de desarrollo (IDE´s). La Universidad de Las Ciencias Informáticas que es donde radica la población de estudio, cuenta con su licencia.

- Para la implementación del sistema se propone la utilización del marco de trabajo Codelgniter, seleccionado porque los desarrolladores pueden realizar proyectos mucho más rápido que creando toda la estructura desde cero, además de ser un marco de trabajo de código abierto.
- PHP 5.2 para la implementación del lado del servidor ya que además de ser rápido, es multiplataforma, posee capacidad de conexión con la mayoría de los manejadores de base de datos entre otros.
- Apache como servidor Web porque es soportado en múltiples sistemas operativos, lo que lo hace prácticamente universal, es una tecnología gratuita de código fuente abierto.
- Como sistema gestor de base de datos PostgreSQL ya que posee una instalación ilimitada, no se puede demandar por violar acuerdos de licencia, puesto que no hay costo asociado a la licencia del software.
- Como IDE Zend Estudio por las bondades que brinda.
- Para la construcción de la interfaz de la aplicación se utilizará la librería de JavaScript ExtJS debido a que posee componentes de interfaz de usuario de alto performance y personalizables, un modelo de componentes extensible y licencia de código abierto y comercial.

1.5 Conclusiones

Luego del estudio realizado a las soluciones existentes que abordan problemáticas similares a la planteada en este trabajo, se arrojan las siguientes conclusiones:

- Ninguno de los sistemas de gestión analizados sigue los mismos objetivos que el proceso de planificación de profesores para el cuidado de exámenes en la Facultad 15. Similares fueron los resultados arrojados del estudio realizado en el país sobre la existencia de algún sistema de gestión que se adecuara a estos requerimientos, donde no se señaló ningún caso que siguiera la estrategia marcada. En la UCI este proceso se realiza mediante una sucesión de requisitos propios de la universidad, ello propicia la necesidad de realizar un sistema de gestión para lograr una mejor planificación del cuidado de exámenes en la Facultad 15 de esta universidad.
- Ninguno de los sistemas de gestión similares al propuesto a desarrollar se adaptan a los principios del sistema educacional de nuestro país y en particular de nuestra universidad, además de poseer

altos costos para la obtención de sus licencias ya que están desarrollados sobre tecnologías propietarias lo que va en contra de las políticas de migración hacia software libre del país y la universidad.

Del análisis detallado de las tendencias actuales y las tecnologías y herramientas utilizadas en el desarrollo de software y específicamente en el desarrollo web, se han logrado definir las más adecuadas para darle solución a la problemática planteada, partiendo del concepto de migración a software libre que sigue el país actualmente.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

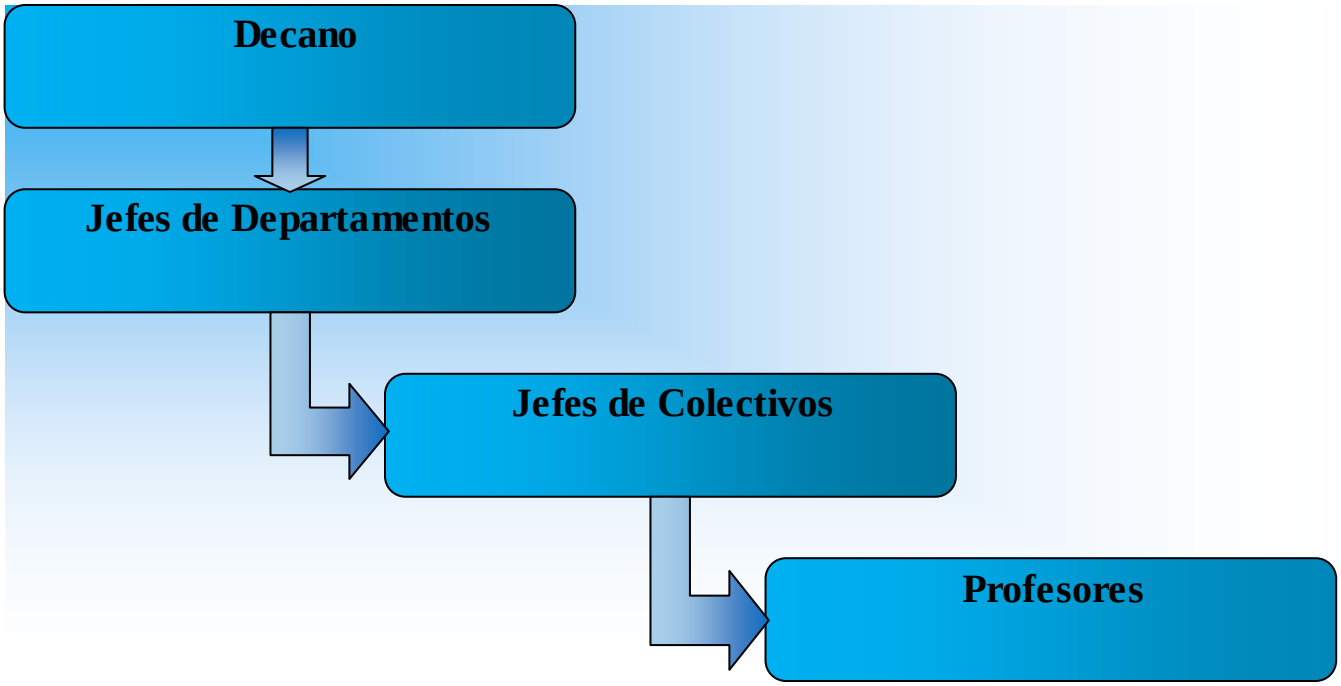
En el presente capítulo se realizará la descripción de la propuesta para dar solución a la problemática existente, primeramente se modela el negocio propuesto, se identifican los actores, trabajadores, los casos de uso correspondientes y la descripción de los mismos. También se enumeran los requisitos; tanto funcionales como no funcionales que debe cumplir el sistema que se propone, se identificará mediante un diagrama de casos de uso las relaciones de los actores que interactúan con el sistema.

2.2 Modelo del negocio actual

El negocio que requiere la solución informática a desarrollar en este trabajo comprende el desempeño de un proceso relacionado con la planificación docente del área de formación de la Facultad 15. En este sentido los jefes de departamento de la facultad se encargan de la ejecución de un conjunto de actividades relacionadas con la planificación de los profesores para ejecutar el cuidado de los exámenes previstos en cada una de las asignaturas. A continuación se especifica con más detalles la secuencia de actividades que se realiza en este proceso.

2.2.1 Planificación del cuidado de exámenes

Estructura de la Facultad 15 según las responsabilidades



Para llevar a cabo la planificación de cuidado de exámenes el Jefe de Departamento crea una planificación, donde dada una asignatura, se le asigna el día en que se va realizar su examen, la cantidad de profesores que intervienen por grupos y una breve descripción del examen; luego de creada la misma se dirige a generar la distribución de los profesores por grupos, culminando así el proceso.

2.3 Análisis crítico de ejecución de los procesos actuales

En estos momentos toda la gestión de la información asociada a la planificación de los cuidados de exámenes en la Facultad 15 se lleva de forma manual. Esta situación dificulta la asignación de profesores para el cuidado de exámenes, ya que el proceso manual es muy engorroso. También se hace muy difícil brindar determinada información estadística puesto que no hay forma de llevarla manualmente.

2.4 Objeto de automatización

Con el fin de solucionar los problemas existentes en la Facultad 15 en el proceso de gestión de profesores para el cuidado de exámenes, se propone realizar un sistema informático que tiene como objeto de automatización inicial la planificación de profesores para el cuidado de exámenes, dando la posibilidad de la generación automática de los listados de cuidados de exámenes, posibilitando así una mayor

confidencialidad, seguridad y control de la información que se manipula, así como la disminución del tiempo de planificación de cuidados de exámenes.

2.5 Actores del negocio

Un actor del negocio es cualquier individuo, grupo, entidad, organización, máquina o sistema de información externos; con los que el negocio interactúa [11].

Actor del negocio	Descripción
Jefe de Colectivo	Persona que obtiene o se beneficia con los resultados de una eficiente planificación de los cuidados de exámenes.

Tabla 1: Actores del Negocio

2.6 Trabajadores del negocio

Un trabajador del negocio define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos [11].

Trabajador del negocio	Descripción
Jefe de Departamento	Encargado de gestionar a los profesores de su departamento, además gestiona la planificación de cuidados de examen, también asigna evaluación a los profesores.
Profesor	Ejecuta el cuidado de examen que le ha sido asignado.

Tabla 2: Trabajadores del Negocio

2.7 Diagrama de casos de uso del negocio

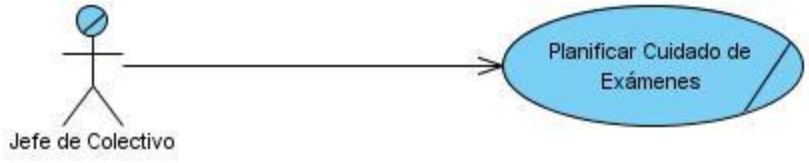


Figura 1: Diagrama de casos de uso del negocio.

2.8 Descripción de los casos de uso del negocio

Un caso de uso del negocio representa a un proceso de negocio, por lo que se corresponde con una secuencia de acciones que producen un resultado observable para ciertos actores del negocio [11]. Luego de la descripción del negocio a tratar, realizada anteriormente, se definen el caso de uso Planificar Cuidado de Exámenes. Seguidamente se especifica la descripción textual que lo acompaña.

Nombre del Caso de Uso	Planificar Cuidado de Exámenes
Actores	Jefe de Colectivo
Propósito	Planificar el cuidado de exámenes para lograr una eficiente asignación de profesores.
Resumen	El caso de uso inicia una vez que el Jefe de Colectivo solicita la planificación del cuidado de examen. De esta manera los jefes de departamento a partir de los datos de la planificación adicionada generan la distribución de los profesores, informándole finalmente al profesor implicado.
Curso Normal de los Eventos	
Acción del Actor	**Respuesta del proceso de negocio**
1 El Jefe de Colectivo solicita la	2 El Jefe de Departamento adiciona una planificación

planificación del cuidado de examen.	con sus datos asociados.
	3 El Jefe de Departamento selecciona la opción generar distribución de profesores de la planificación adicionada.
	4 El Jefe de Departamento le informa al profesor la planificación correspondiente.
	5 El Jefe de Departamento chequea el cumplimiento de la planificación y actualiza si el profesor cumplió o no con la planificación asignada, para mantener un control sobre el desempeño de cada profesor.
Curso Alternativo de los Eventos	
	4.1 El profesor le informa al Jefe de Departamento su inconveniente con la planificación. El Jefe de Departamento le sugiere que cambie con otro profesor.
	5.1 El Jefe de Departamento detecta que algún profesor no está cumpliendo con la planificación y le asigna una evaluación de mal.
Prioridad	Crítico

Tabla 3: Descripción de los casos de uso del negocio.

2.9 Diagrama de actividades

Un diagrama de actividades representa los flujos de trabajo paso a paso del negocio y operacionales de los componentes en un sistema. Un diagrama de actividades muestra el flujo de control general [12].

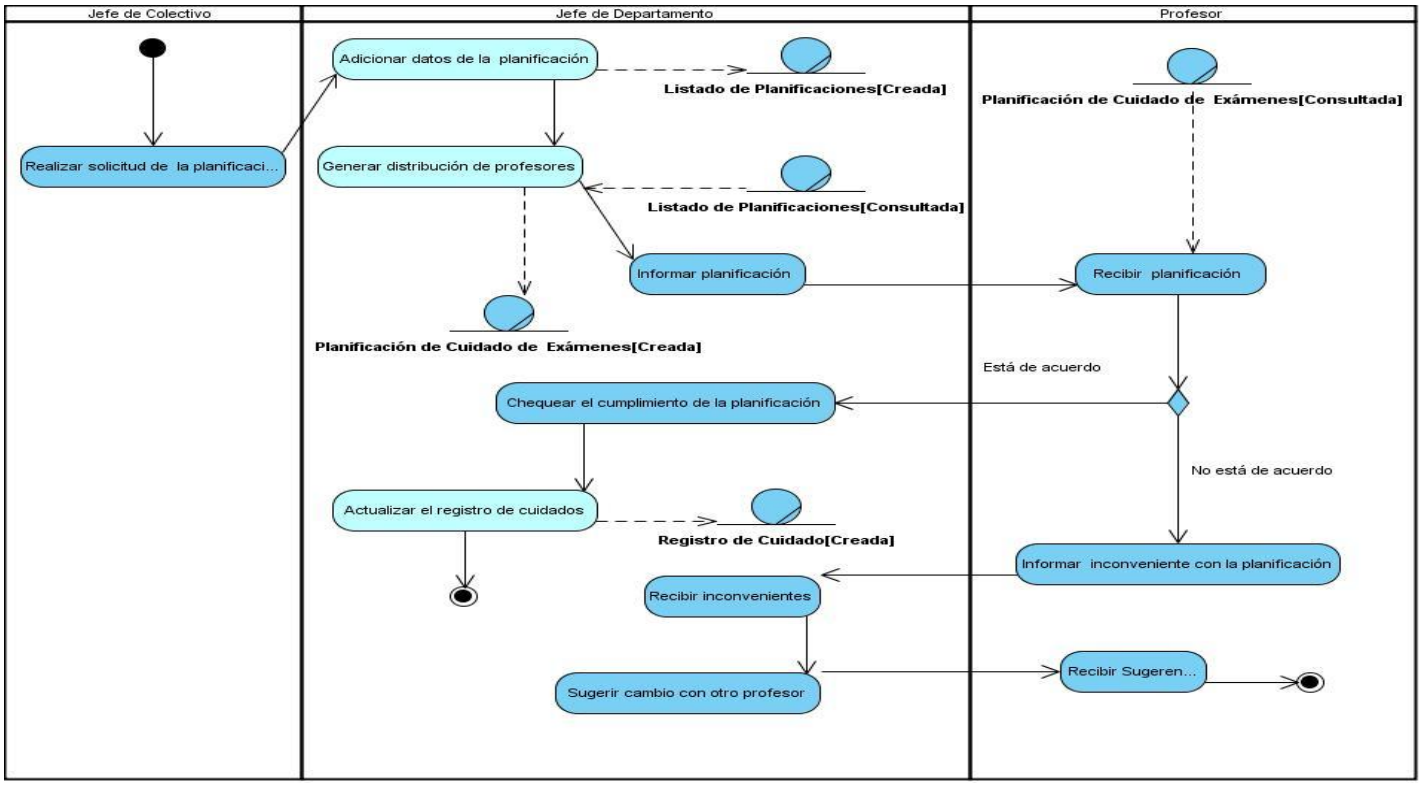


Figura 2: Diagrama de actividades del caso de uso Planificar Cuidado de Exámenes.

2.10 Modelo de objetos del negocio.

El modelo de objetos del negocio describe cómo colaboran los trabajadores y las entidades del negocio dentro del flujo de trabajo de cada uno de los procesos del negocio [12].

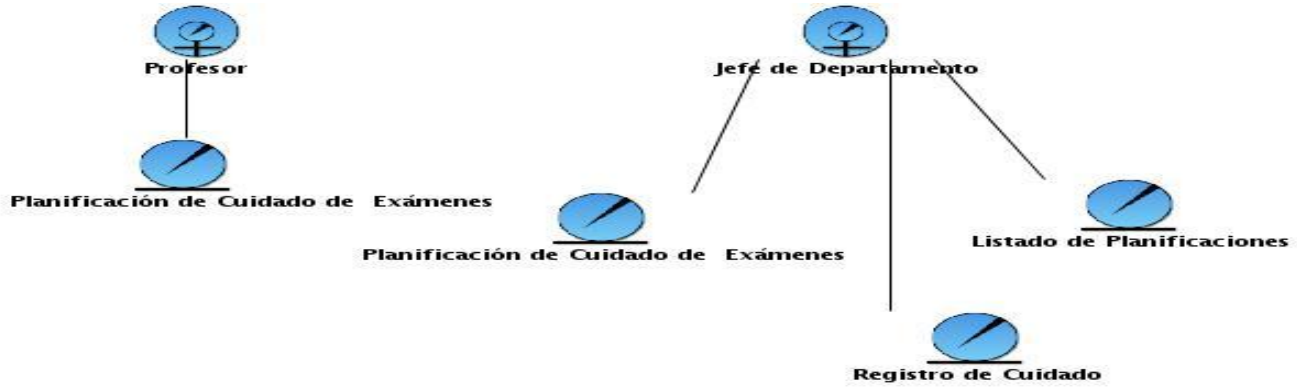


Figura 3: Modelo de objetos del negocio.

2.11 Especificación de los requisitos de software

Los requerimientos de software son condiciones o capacidades que debe tener el sistema para satisfacer las necesidades de un cliente, estos deben ser especificados por escrito. A continuación se muestra un listado de los requerimientos tanto funcionales como no funcionales de la aplicación.

2.11.1 Definición de los requisitos funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir, a continuación se presenta un conjunto de ellos:

RF1: Autenticar usuario.

RF2: Gestionar profesores.

- 2.1: Adicionar profesores.
- 2.2: Eliminar profesores.
- 2.3: Modificar profesores.

RF3: Gestionar planificación.

- 3.1: Adicionar planificación.
- 3.2: Modificar planificación.
- 3.3: Eliminar planificación.

RF4: Generar distribución.

RF5: Asignar evaluación.

RF6: Generar reporte.

RF7: Gestionar asignatura.

- 7.1: Adicionar asignatura.
- 7.2: Modificar asignatura.
- 7.3: Eliminar asignatura.

RF8: Gestionar departamento.

8.1: Adicionar departamento.

8.2: Modificar departamento.

8.3: Eliminar departamento.

RF9: Gestionar jefe de departamento.

9.1: Adicionar jefe de departamento.

9.2: Modificar jefe de departamento.

9.3: Eliminar jefe de departamento.

RF10: Gestionar grupo.

10.1: Adicionar grupo.

10.2: Modificar grupo.

10.3: Eliminar grupo.

RF11: Gestionar local.

10.1: Adicionar local.

10.2: Modificar local.

10.3: Eliminar local.

RF12: Mostrar planificación.

RF13: Cerrar sesión.

RF14: Consultar ayuda.

2.11.2 Definición de los requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Representan las características que hacen al producto atractivo, usable, rápido o confiable. Permitiendo que puedan

marcar la diferencia entre un producto bien aceptado y uno con poca aceptación [11]. A continuación se expondrán los requisitos no funcionales que el sistema propuesto debe cumplir:

- Usabilidad: El sistema debe estar disponible las 24 horas del día, se debe acceder al sistema desde cualquier PC que esté conectada a la red. El sistema debe diferenciar las interfaces gráficas y opciones para usuarios que accedan con diferentes roles.
- Rendimiento: El sistema debe ser capaz de gestionar toda la información referente al cuidado de exámenes y cargar las páginas dinámicas lo más rápido posible para dar respuesta a las solicitudes de los usuarios. Debe soportar varios usuarios conectados simultáneamente a la base de datos en cualquier momento dado. Debe ser eficiente a la hora de gestionar las solicitudes, alcanzando los resultados deseados sin hacer un uso extensivo de la navegación por el sitio
- Soporte: Fácil para el mantenimiento, de configuración sencilla y factible para los clientes.
- Portabilidad: La implementación del sistema sobre PHP propicia que sea una aplicación multiplataforma, debe correr no solo sobre Windows sino también sobre Linux o Unix para que se pueda llevar a cabo esta acción sin necesidad de realizar cambios significativos.
- Seguridad: La seguridad quedará bien definida, se establecerá un sistema de roles para diferentes usuarios como el administrador del sistema, los jefes de departamentos y los profesores, cada uno de ellos solo podrá ejecutar las acciones que le corresponde.
- Apariencia o interfaz externa: El sistema propuesto será usado también por personas que tengan un conocimiento medio de informática, por lo que la interfaz será amigable y fácil de usar.
- Software: Tener como sistema operativo Windows 98 o superior, se necesita tener instalado un servidor de aplicaciones web Apache, y haber instalado PostgreSQL.
- Hardware:
  - ✓ Servidor: Se requiere un servidor de 512 MB de RAM como mínimo y 100 MB como espacio mínimo en el disco duro.
  - ✓ Cliente: Las computadoras clientes deben de estar conectadas a la red y tener al menos 128 MB de RAM.

- Políticos y culturales: No debe contener palabras en otros idiomas, utilizar un vocabulario acorde a su entendimiento.
- Ayuda y documentación: El sistema debe brindar un manual de usuario para el correcto uso de sus funcionalidades y brindarle una mayor experiencia a los usuarios.
- Legales: Es un deber proteger la información por parte de las personas que tienen derecho de administración u otros que pongan en peligro la integridad y seguridad del sistema.

2.12 Actores del sistema

Los actores del sistema representan el rol que desempeña una o varias personas, un equipo o un sistema automatizado. Son parte del sistema y pueden intercambiar información con él o ser recipientes pasivos de información. Estos fueron trabajadores del negocio que interactúan con el sistema [11], en este caso los actores que interactúan con el sistema se definen a continuación:

2.12.1 Descripción de los actores del sistema

Actores del Sistema	Descripción
Jefe de Departamento	Es el responsable de realizar la planificación de cuidado de exámenes y generar el reporte de la misma, así como gestionar los profesores de su departamento.
Profesor	Encargado de realizar el cuidado de examen que le ha sido planificado.
Administrador	Encargado de gestionar los departamentos, los jefes de los mismos, los grupos y locales de la facultad y las asignaturas.
Usuario del Sistema	Es quien intenta ingresar al sistema. Si la autenticación es correcta recibe permisos en dependencia del rol que desempeña.
Servidor de Dominio	Encargado de realizar la validación de la contraseña de los usuarios.

Tabla 4: Descripción de los actores del sistema.

2.13 Diagrama de casos de uso del sistema

Un diagrama de casos de uso del sistema representa gráficamente a los procesos y su interacción con los actores, permite comprender qué actor interviene en qué proceso.

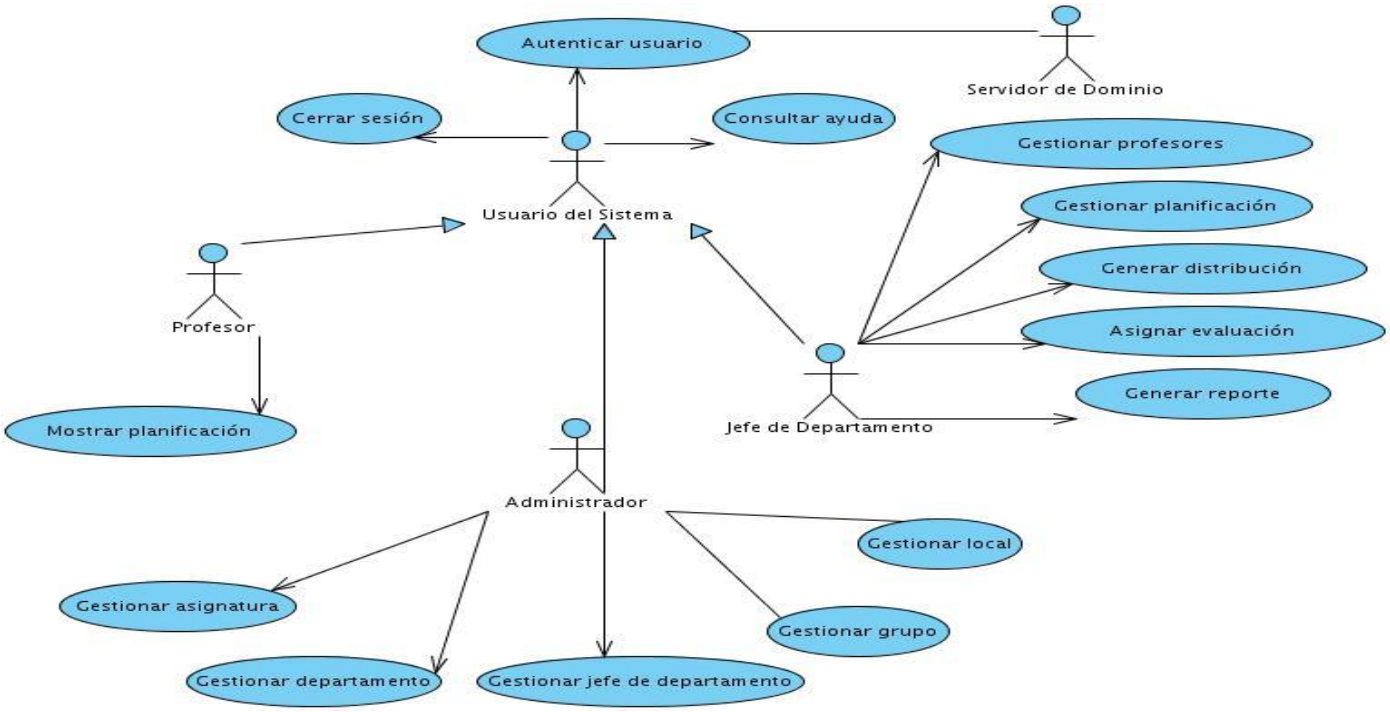


Figura 4: Diagrama de casos de uso del sistema.

2.14 Descripción de los casos de usos del sistema

Los casos de uso del sistema a tratar son:

- Autenticar usuario.
- Gestionar profesores.
- Gestionar planificación.

- Generar distribución.
- Asignar evaluación.
- Generar reporte.
- Gestionar asignatura.
- Gestionar departamento.
- Gestionar jefe de departamento.
- Gestionar grupo.
- Gestionar local.
- Mostrar planificación.
- Cerrar sesión.
- Consultar ayuda.

Para el desarrollo del informe de este trabajo se presentará uno de los casos de uso fundamentales del sistema, el resto puede ser consultado en el Anexo 1.

CU-3	Gestionar planificación.
Actores	Jefe de departamento (inicia).
Propósito	Permitir al jefe de departamento gestionar las planificaciones.
Resumen	El caso de uso se inicia cuando el Jefe de Departamento selecciona la opción gestionar planificación. Posteriormente el sistema muestra las opciones de gestión para que el usuario seleccione la deseada. Cuando se adiciona una planificación se manipulan datos tales como: el nombre de la asignatura que se le realizará el examen, la cantidad de profesores que cuidan por grupos, y una descripción del tipo de examen. Permitiendo el sistema modificar y eliminar estos datos.
Referencias	RF3
Precondiciones	El jefe de departamento se debe haber autenticado con anterioridad.

Poscondiciones	Información gestionada en la base de datos.
Escenario 1: Adicionar planificación	
Curso Normal de los Eventos	
Acción del Actor	**Respuesta del Sistema**
1 El jefe de departamento selecciona la opción Adicionar planificación.	2 El sistema muestra una interfaz con los datos a llenar.
3 El jefe de departamento ingresa los datos solicitados por el sistema y oprime el botón “Adicionar”.	4 El sistema verifica que los campos del formulario no estén vacíos.
	5 El sistema verifica la planificación que no esté registrada en la base de datos.
	6 El sistema registra la planificación en la base de datos correspondiente.
Curso Alternativo de los Eventos	
	4.1 Muestra un mensaje de error indicando que debe llenar todos los campos (“llene todos los campos”) y da la posibilidad nuevamente de llenar los campos requeridos.
	5.1 Muestra un mensaje de error indicando que la planificación ya existe.
Escenario 2: Modificar planificación	
Curso Normal de los Eventos	
1 El jefe de departamento selecciona la opción Modificar planificación.	2 El sistema muestra una interfaz con los datos de la planificación a modificar.
3 El Administrador modifica los campos y oprime el botón “Aceptar”.	4 El sistema verifica que los campos modificados estén correctos.
5 El Administrador modifica los datos	6 El sistema guarda la planificación modificada.

necesarios y pincha el botón “Modificar”.	
Curso Alternativo de los Eventos	
	4.1 El sistema muestra un mensaje indicando el error.
Escenario 3: Eliminar planificación	
Curso Normal de los Eventos	
1 El jefe de departamento selecciona la opción Eliminar planificación.	2 El sistema muestra una interfaz con todas las planificaciones.
3 El jefe de departamento selecciona una planificación y oprime el botón eliminar.	4 El sistema actualiza los datos eliminados.
	5 Se notifica que el usuario se eliminó satisfactoriamente.
Curso Alternativo de los Eventos	
	3.1 Se muestra un mensaje de confirmación antes de eliminar definitivamente.
Prioridad	Crítico

Tabla 5: Descripción del caso de uso del sistema Gestionar planificación.

2.15 Conclusiones

En este capítulo se describió el negocio, especificando su proceso fundamental: Planificar Cuidado de Exámenes, se identificó además los actores y trabajadores involucrados. Se modeló el diagrama de casos de uso del negocio, acompañado de la descripción textual del caso de uso del negocio, se modeló el diagrama de actividades del caso de uso del negocio, así como el modelo de objetos del negocio. Se definieron los requisitos no funcionales y los funcionales que sirvieron como entrada para la definición de los casos de uso del sistema involucrados en el modelo de sistema mostrado y acompañado además de las descripciones textuales de cada uno de ellos y del diagrama de casos de uso del sistema.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

3.1 Introducción

En el presente capítulo, después de haber definido las funcionalidades requeridas, se procederá a la construcción de la solución propuesta. El análisis de un sistema se centra en la investigación del problema y no en la manera de definir una solución, mientras que el diseño pone de relieve una solución lógica: cómo el sistema satisface los requerimientos funcionales, requerimientos de calidad y las restricciones, es decir, en esta parte del proceso de desarrollo del software se decide como se va a llevar a cabo el mismo. Se especificarán elementos relacionados con el análisis y el diseño del sistema a construir, formalizando en este sentido los diagramas de clases del análisis y los diagramas de clases del diseño, haciendo uso de los estereotipos Web, para cada uno de los casos de uso de la aplicación propuesta.

3.2 Modelo de análisis

El objetivo fundamental del análisis está dado en convertir los requisitos funcionales en un diseño de clases, observando las relaciones e interacciones que existen entre ellos, obteniéndose una visión del sistema. Es representado por las relaciones entre los actores del sistema y las clases de análisis, las que se clasifican en clase de interfaz, controladora y de entidad.

Entre las principales ventajas que aporta al proceso de desarrollo de un software es que suaviza la transición al diseño, sirve para tener una visión general de la propuesta de sistema, y entre otras cosas apoya el cambio a otra plataforma de programación puesto que no está ligado a un lenguaje en particular. El objetivo principal de este flujo son los diagramas de clases de análisis, los cuales muestran qué clases participan en las realizaciones de los distintos casos de usos, constituyendo la base sobre la cual debe realizarse el Modelo de Diseño.

3.2.1 Diagramas de Clases de Análisis

A continuación se muestra el diagrama de clase de análisis del caso de uso Gestionar planificación, encontrándose los restantes en el Anexo 2.

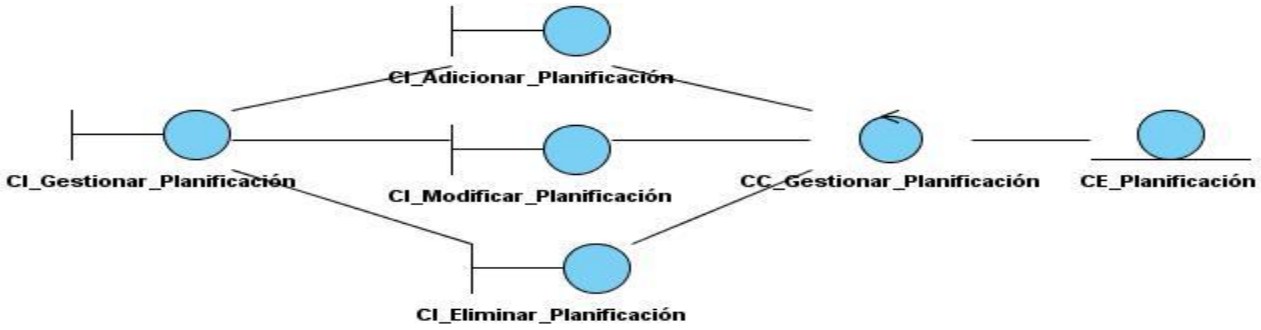


Figura 5: Diagrama de clase de análisis del caso de uso Gestionar planificación.

3.2.2 Diagramas de interacción

Los diagramas de interacción muestran cómo los objetos se comunican entre ellos mediante la transferencia de mensajes. Los diagramas de interacción están constituidos por dos tipos: los diagramas de Secuencia y los diagramas de Colaboración. Ambos expresan información similar, pero en una forma diferente. Los de secuencia destacan la ordenación temporal de los mensajes y los de colaboración, permiten representar enlaces y mensajes entre objetos. A continuación se muestra el diagrama de secuencia del caso de uso Gestionar planificación específicamente de la funcionalidad Adicionar planificación, encontrándose los restantes en el Anexo 3.

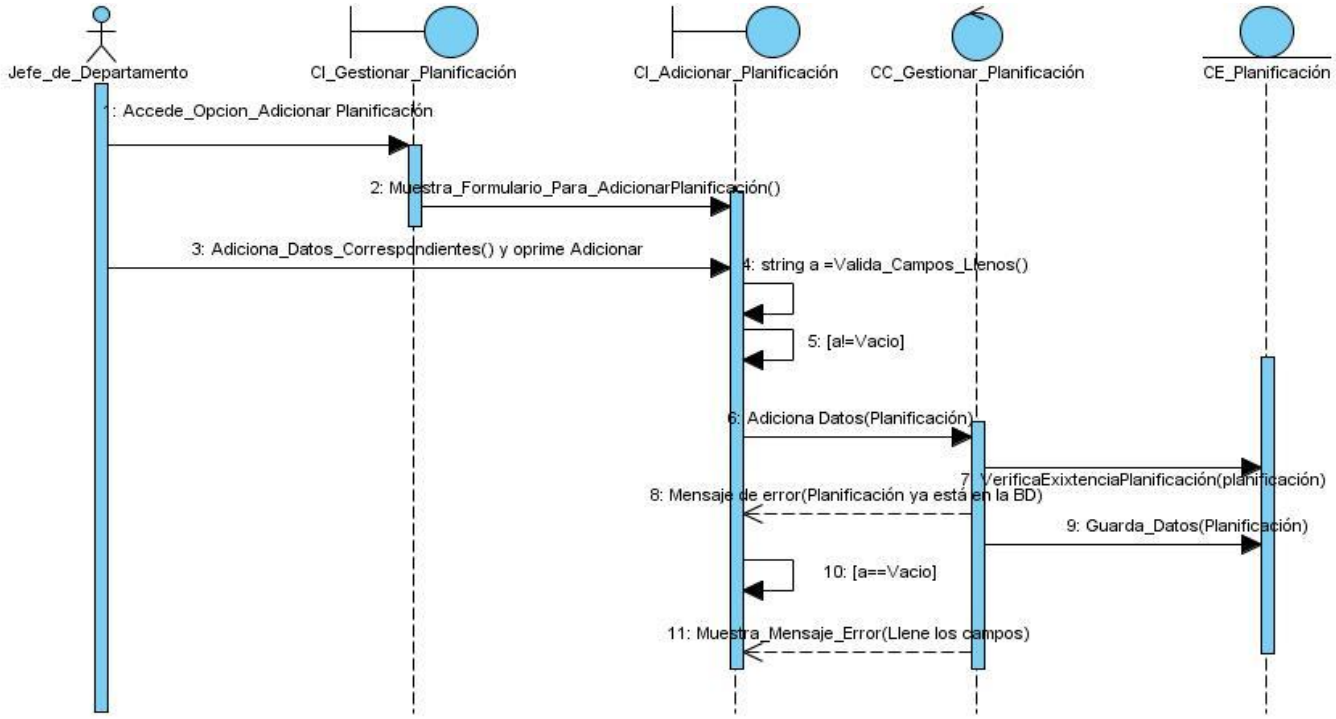


Figura 6: Diagrama de secuencia del caso de uso Gestionar planificación (Adicionar planificación).

3.3 Modelo de Diseño

En la fase de diseño se modela el sistema de manera que soporte todos los requisitos, tanto funcionales como no funcionales, creándose así una entrada apropiada para las actividades de implementación. El diseño debe ser suficiente para que el sistema pueda implementarse sin ambigüedades. Por esta razón es que tiene como propósito adquirir una comprensión de los aspectos relacionados con los requisitos no funcionales. Además de crear entradas apropiadas y un punto de partida para la implementación, para que de una forma más manejable esta pueda ser llevada a cabo por diferentes equipos de desarrollo.

3.3.1. Diagramas de Clases de Diseño

El diseño tiene en cuenta los requisitos no funcionales: cómo el sistema cumple sus objetivos. Se realiza para facilitar la implementación del mismo. A continuación se muestra el diagrama de clases de diseño del caso de uso Gestionar planificación, encontrándose los restantes en el Anexo 4.

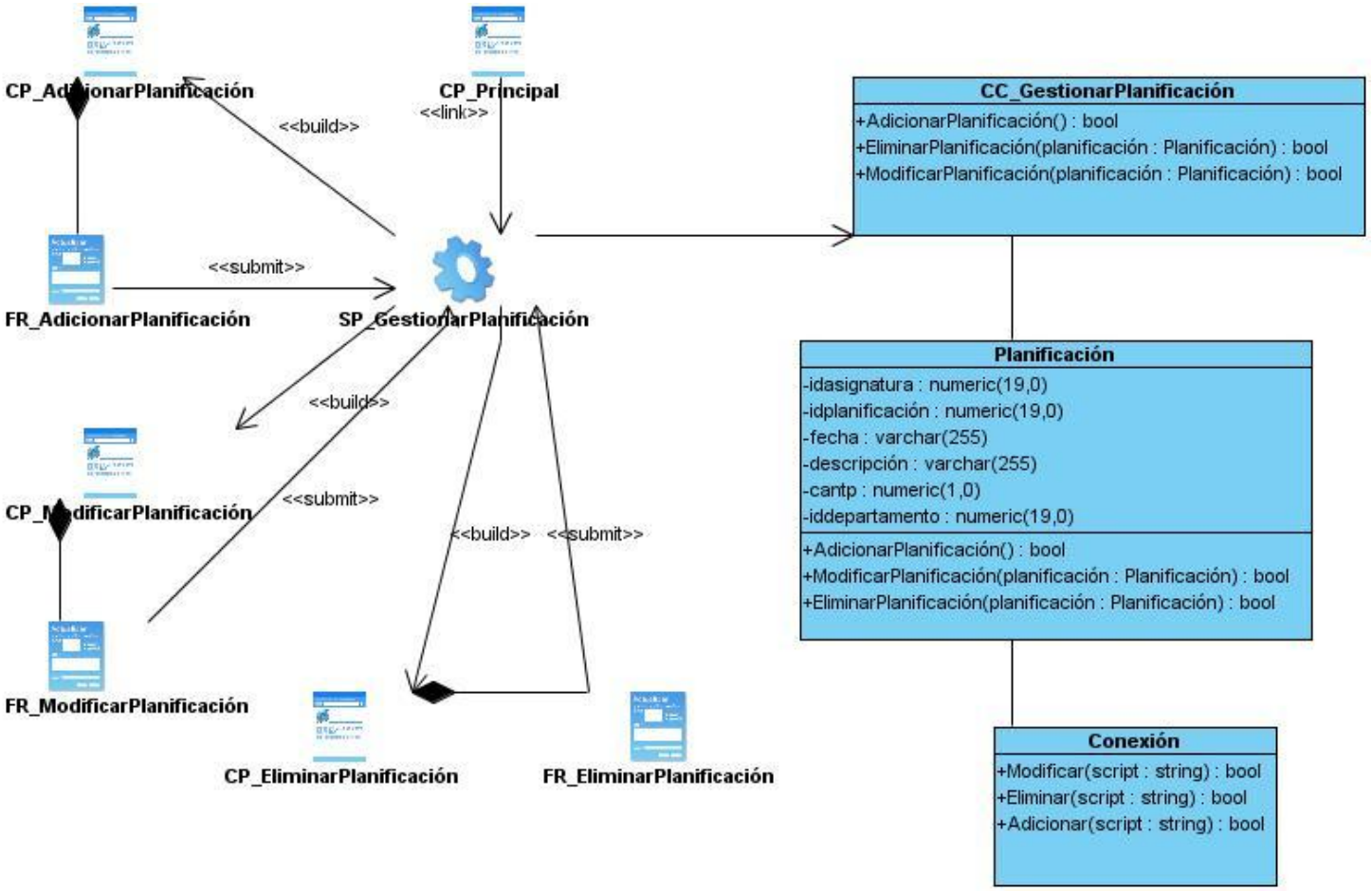


Figura 7: Diagrama de clases de diseño del caso de uso Gestionar planificación.

3.4 Descripción de las tablas de Modelo de Datos

A continuación se muestra la descripción de la tabla de Planificación encontrándose las restantes en el Anexo 5.

Nombre: Planificación		
Descripción: En esta tabla se guardan datos de una planificación		
Atributo	**Tipo**	**Descripción**

idplanificacion	**numeric(19,0)**	**Identificador de la planificación.**
Idasignatura	**numeric(19,0)**	**Identificador de la asignatura.**
Fecha	**varchar(255)**	**Día que le toca cuidar el examen.**
descripción	**varchar(255)**	**Tipo de examen.**
iddepartamento	**numeric(19,0)**	**Identificador del departamento.**
cantp	**numeric(1,0)**	**Cantidad de profesores que cuidan por grupo(1 ó 2)**

Tabla 6: Descripción de la tabla del Modelo de Datos (Planificación).

3.5 Diseño de la Base de Datos

El diseño de la base de datos modela el tratamiento de la información con carácter persistente dentro del sistema. Para este trabajo fueron construidos dos modelos para la representación de los datos persistentes: el Modelo Lógico de Datos y el Modelo Físico de Datos, ambos proporcionan una flexibilidad óptima para el soporte de la informatización entre el Modelo de Análisis, Modelo de Diseño, y la Base de Datos Física.

3.5.1 Modelo Lógico de datos

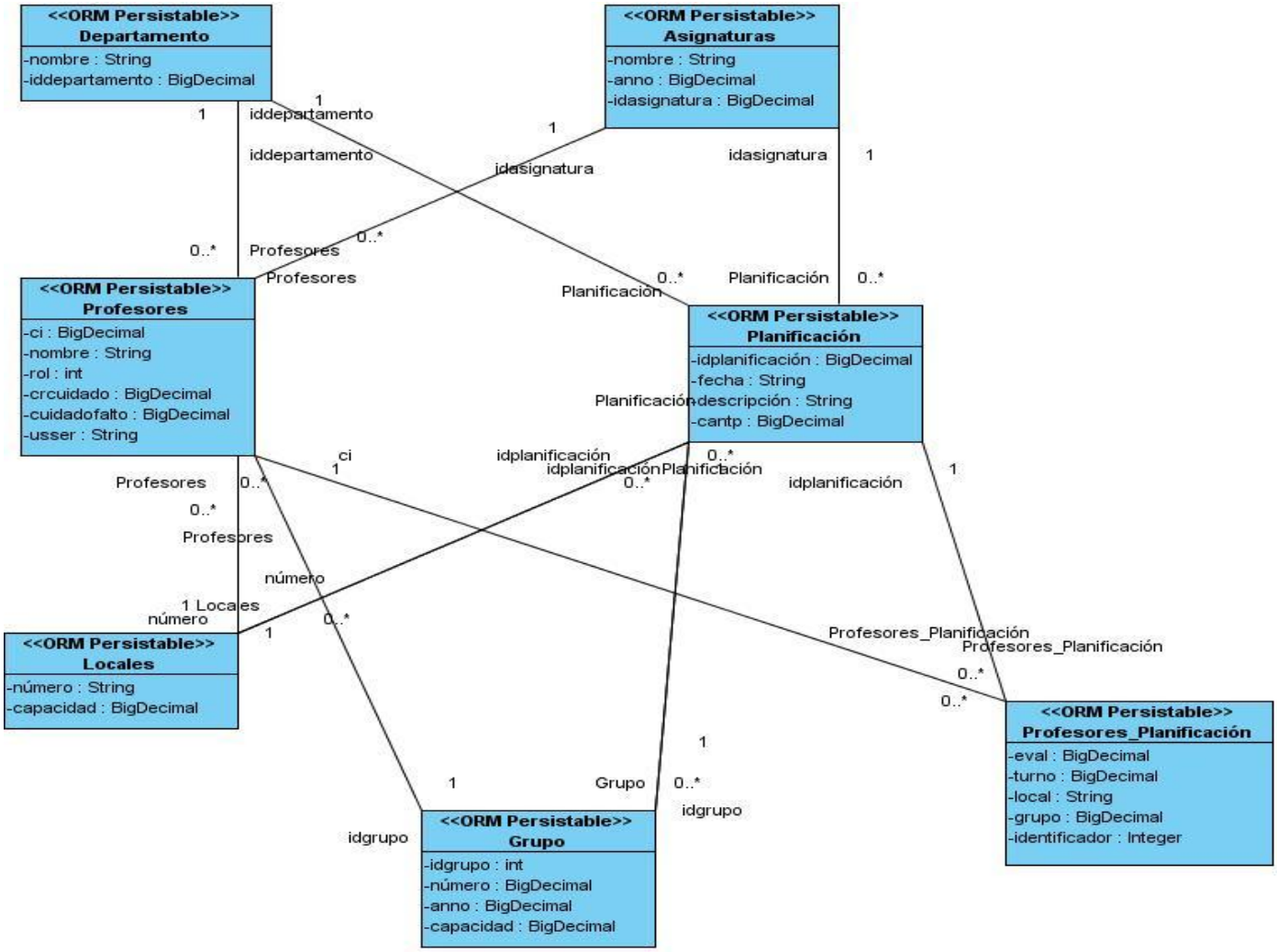


Figura 8: Modelo Lógico de la Base de Datos.

3.5.2 Modelo Físico de datos

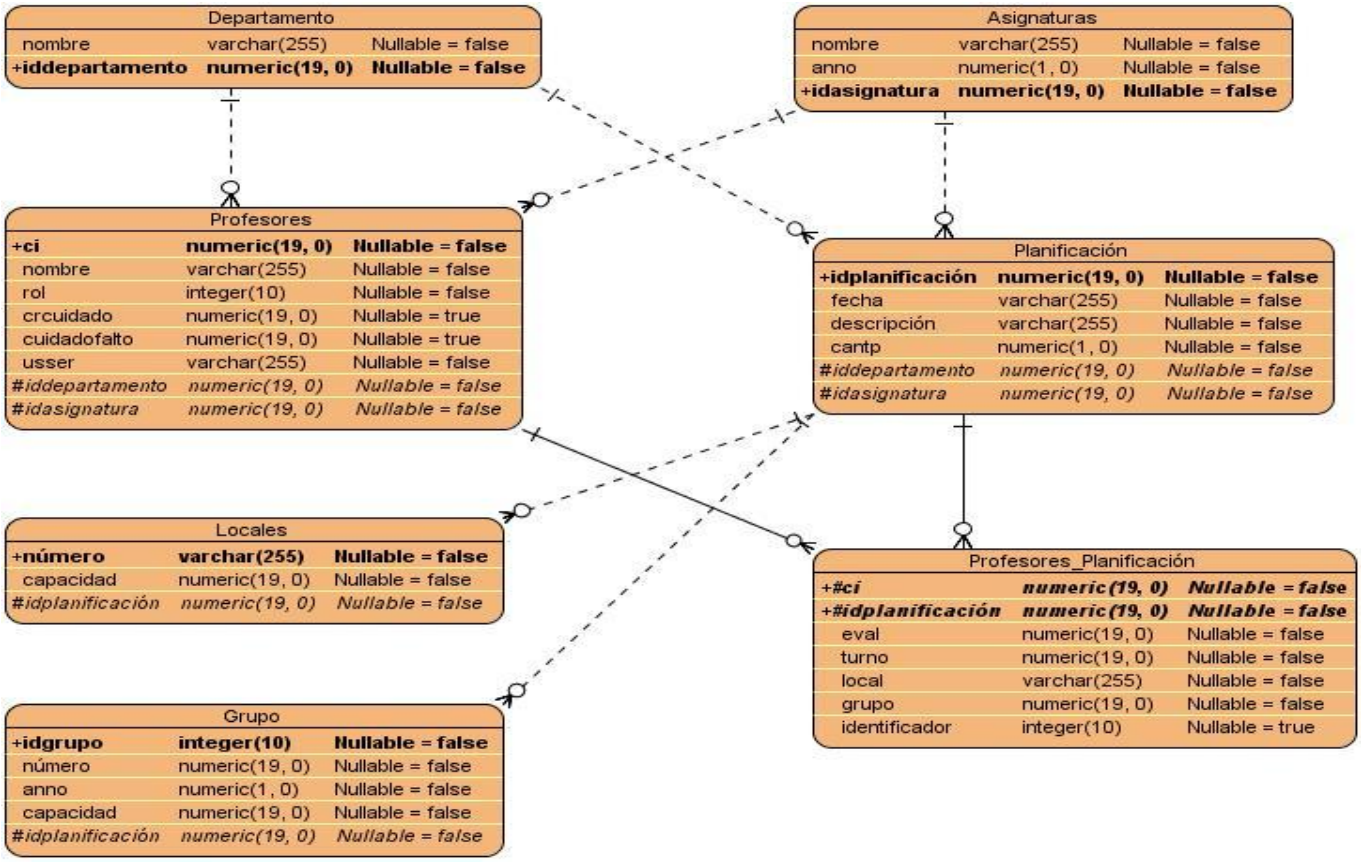


Figura 9: modelo Físico de la Base de Datos.

3.6 Seguridad

En el mundo de hoy es de vital importancia mantener la información, bajo la seguridad e integridad requerida, se hace necesaria la existencia de mecanismos que permitan una protección en cuanto a los datos del sistema. A partir de los principios de poder lograr la integridad, confiabilidad y disponibilidad, de la idea antes expuesta, se tuvo en cuenta para el acceso a la aplicación, una previa autenticación en el sistema donde los únicos que pueden accederlo sean los profesores registrados en la base de datos de la aplicación. De esta manera cada profesor según el privilegio o los permisos asignados, podrá acceder únicamente a las tareas o acciones que le corresponde ejecutar. El encargado de asignar los permisos o asignar los roles a cada usuario es el Jefe de Departamento.

3.7 Conclusiones

En este capítulo fueron modelados los artefactos que tienen lugar en los flujos de trabajo de análisis, diseño e implementación, sustentándose de esta forma la base para la construcción del sistema.

Se desarrolló el análisis y diseño del sistema, permitiendo el avance de la propuesta de solución. En este sentido se construyeron los diagramas de clases de análisis y del diseño para cada caso de uso del sistema, así como los diagramas de interacción correspondientes a cada caso de uso en cada flujo. Se diseñó además la base de datos, conjuntamente con las descripciones de sus tablas correspondientes. De esta manera se traza el camino para la implementación y para las pruebas del sistema posteriormente.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

4.1 Introducción

En el presente capítulo se desarrollarán dos flujos de trabajo, el de implementación y posteriormente se le dará paso a las pruebas. Se presentará el modelo de implementación, mediante el diagrama de componentes, obtenido a partir del modelo del diseño.

Se diseñará además, el modelo de despliegue de la aplicación mostrando la distribución física de sus nodos. Luego se realizará, diseñarán y ejecutarán los casos de prueba para chequear la funcionalidad de la aplicación en aras de solucionar los problemas detectados.

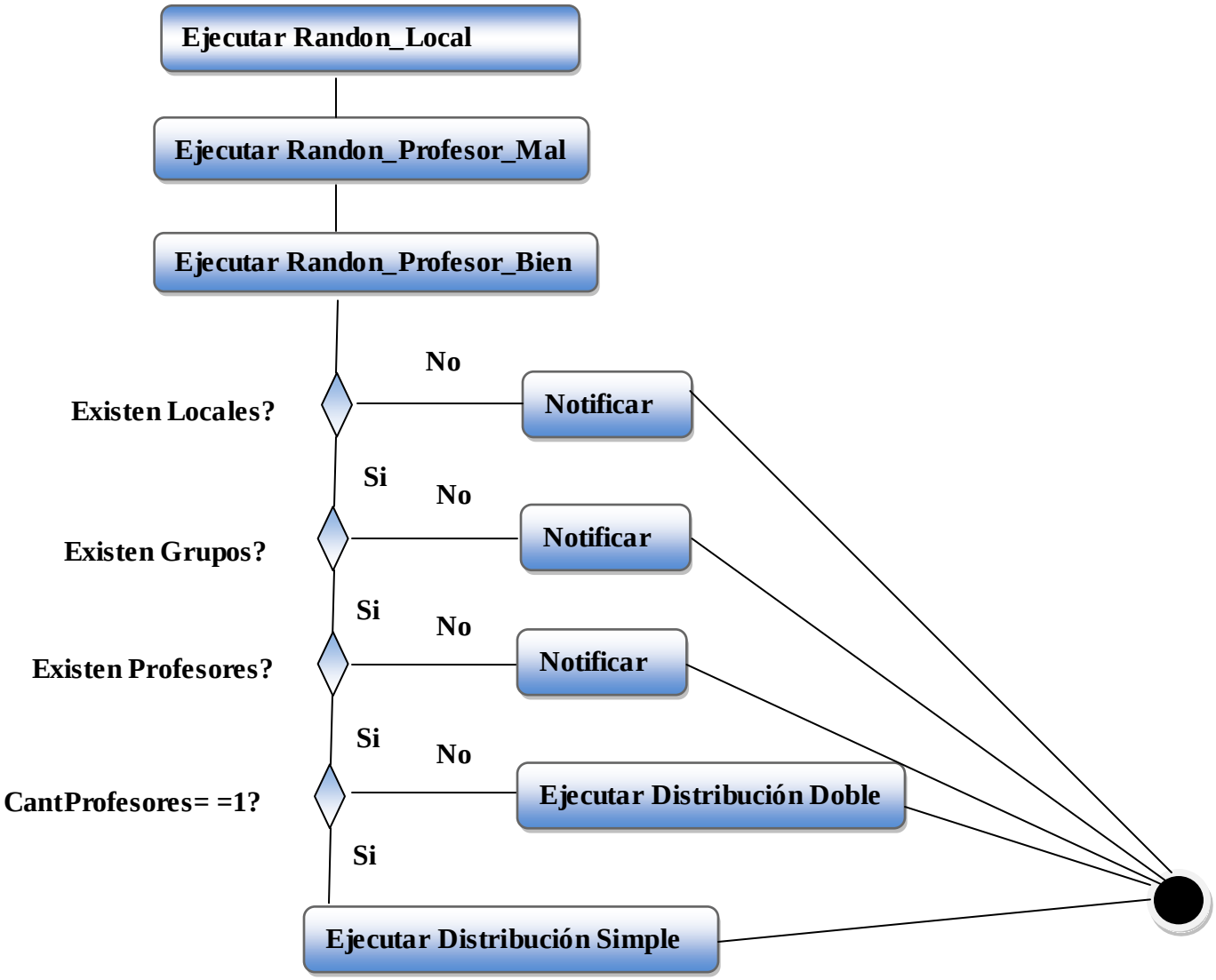
4.2 Implementación

Luego de obtener el resultado del diseño con el modelo obtenido, se procede a ejecutar el flujo de trabajo de implementación.

En este sentido se realiza la implementación en términos de componentes, mostrándose su organización de acuerdo a nodos específicos en el modelo de despliegue. Se establece una organización del sistema en términos de subsistemas de implementación organizados en capas. Para el desarrollo de la implementación del sistema se trabajó con el patrón de arquitectura Modelo - Vista - Controlador. Este se divide en tres partes:

- Modelo: se representa la información con la que se trabaja, es decir, son una serie de datos de los cuales se pueden derivar datos nuevos a través de operaciones.
- Vista: es la representación del Modelo en términos óptimos para la interacción con el usuario, usualmente como la interfaz de usuario.
- Controlador: responde a acciones o eventos solicitados por el usuario generalmente en la Vista y que invocan cambios o modificaciones en el Modelo. [13]

4.2.1 Descripción de la funcionalidad Generar Distribución



Para realizar la distribución de los profesores para el cuidado de examen, primero se realiza de manera aleatoria la repartición de los locales donde les tocará cuidar el examen, luego se hace el mismo procedimiento aleatorio para los profesores que tienen evaluación de mal en un cuidado de examen al menos, de alcanzar con estos, no se escogen los profesores con evaluación de bien, en caso contrario se hace aleatoriamente la selección de los profesores con evaluación de bien. Si existen locales para dicha planificación se ve que hallan grupos donde cuidar, en caso contrario se notifica. Ya con los grupos y los locales ver si hay profesores, si no existieran se notifica y acaba el proceso. Teniendo grupos, locales y profesores hay que saber si la prueba requiere de un profesor para realizar la acción o dos. En caso que sea un solo profesor se realiza la distribución simple y en el caso que se necesiten dos se realizaría entonces la distribución doble.

4.2.2 Modelo de Despliegue

El diagrama de despliegue es un artefacto que modela la arquitectura en tiempo de ejecución de un sistema. En él se indica la situación física de los componentes lógicos desarrollados, lo que significa que sitúa el software en el hardware que lo contiene. A continuación se muestra el diagrama de despliegue, que representa la distribución física del sistema en términos de cómo se distribuirán las funcionalidades entre los nodos, donde cada nodo representa un recurso de cómputo, siendo estos procesadores o dispositivos hardware que se necesitan para el despliegue del sistema:

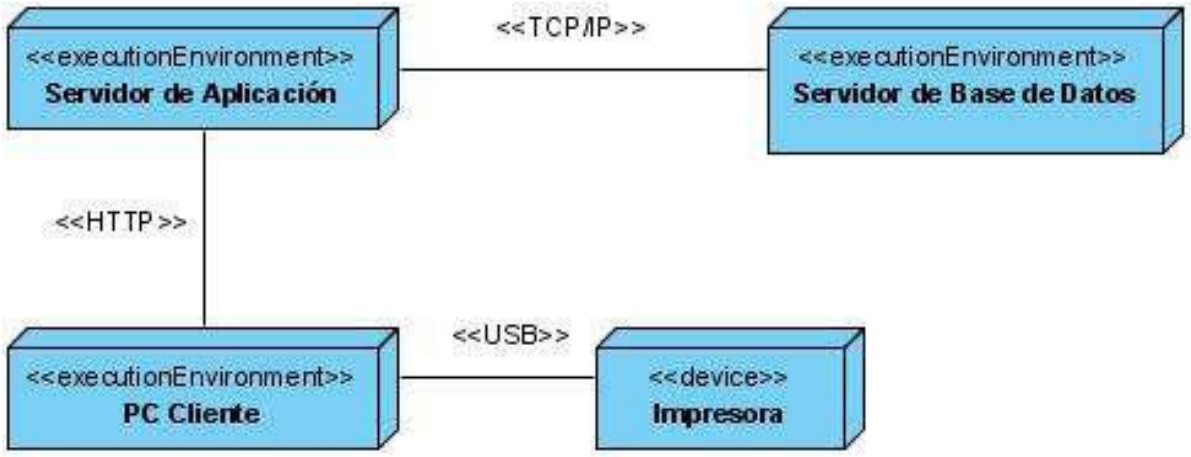


Figura 10: Diagrama de despliegue.

A continuación se muestra la justificación de los nodos citados con anterioridad en el diagrama de despliegue:

- Servidor de Aplicación: Contendrá y facilitará el acceso a la aplicación desarrollada. Se ejecuta continuamente en una computadora que interpreta las peticiones HTTP que recibe por parte de un cliente y las satisface. Este servidor se debe ajustar a las especificaciones definidas en los requerimientos de hardware del sistema.
- Servidor de Base de Datos: Concentrará y facilitará el acceso a los datos relacionados con este proceso, lo cual favorecerá la ejecución del mismo. Permitirá además crear, modificar, mantener, realizar consultas, actualizar, y generar informes en la Base de datos, asegurando su integridad, confidencialidad y seguridad. Este servidor se debe ajustar a las especificaciones definidas en los requerimientos de hardware del sistema.
- PC Cliente: El usuario necesitará de una computadora para acceder o interactuar con la aplicación, que debe cumplir con los requerimientos no funcionales definidos para el hardware.
- Impresora: Estará disponible y será utilizada en casos en que sea necesaria la impresión de los reportes generados con la aplicación.

4.3 Modelo de Prueba

Las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo condiciones o requerimientos especificados. Los resultados son observados y registrados y una evaluación es hecha de algún aspecto del sistema o componente [13]. Siendo un elemento que representa una revisión final de las especificaciones del diseño y de las codificaciones, de manera que encontrar un error que no había sido detectado con anterioridad es un objetivo fundamental. De ahí que el objetivo de la prueba sea: diseñar pruebas que saquen a la luz diferentes clases de errores con la menor cantidad de tiempo y espacio [13].

4.3.1 Métodos de Prueba

Dos de los más conocidos métodos de prueba que se le realizan a un sistema se pueden encontrar: los métodos de caja blanca y los de caja negra, permitiendo que se tracen pruebas en la menor cantidad de tiempo.

La prueba de caja blanca del software comprueba los caminos lógicos del software proponiendo casos de prueba donde se ejerciten conjuntos específicos de condiciones y/o bucles [13]. Estas requieren del

conocimiento de la estructura interna de las aplicaciones pudiendo examinar el estado del programa en varios puntos con el objetivo de verificar si el estado real coincide con el esperado.

La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. De esta manera los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene [13]. Las cuales no consideran la estructura interna del programa ni validan funciones ocultas.

Las pruebas desarrolladas en el sistema son las de caja negra, llevadas a cabo sobre la interfaz. Pretendiendo demostrar que las funcionalidades del software son operativas y que las entradas se aceptan de forma adecuada.

4.4 Casos de pruebas de la aplicación

En este epígrafe se muestra el diseño de los casos de pruebas a partir de algunos de los casos de uso más significativos del sistema.

4.4.1 Caso de prueba para la funcionalidad Adicionar Profesor

Nombre de la funcionalidad	**Descripción General**	**Escenario de Pruebas**	**Flujo del Escenario**
Adicionar Profesor.	El sistema permite adicionar un nuevo profesor a un departamento determinado.	EP 1.1 Adicionar un profesor introduciendo los datos correctamente al presionar el botón **Aceptar.**	- Se presiona el botón **Adicionar Profesor**. - Se introducen los datos correctamente. - Se presiona el botón **Aceptar**. - Se muestra la notificación “Se insertó correctamente el Profesor”
		EP 1.2 Adicionar un profesor introduciendo	- Se presiona el botón **Adicionar**

		datos inválidos al presionar el botón **Aceptar.**	**Profesor.** - Se introducen los datos inválidos. - Se presiona el botón **Aceptar**. - Se muestra la notificación "No se pudo insertar correctamente el Profesor"
		EP 1.3 Adicionar un profesor dejando información requerida en blanco al presionar el botón **Aceptar.**	- Se presiona el botón **Adicionar Profesor.** - Se introducen los datos y se deja información requerida en blanco. - Se presiona el botón **Aceptar**. - Se muestra la notificación "Existen campos obligatorios en blanco"
		EP 1.4 Al presionar el botón **Cancelar.**	- Se presiona el botón **Adicionar Profesor.** - Se introducen los datos o no en los campos. - Se presiona el botón **Cancelar.**

Tabla 7: Escenario de flujos para la funcionalidad Adicionar Profesor.

4.4.2 Caso de prueba para la funcionalidad Adicionar Planificación

Nombre de la funcionalidad	Descripción General	Escenario de Pruebas	Flujo del Escenario
Adicionar Planificación.	El sistema permite adicionar una nueva planificación.	EP 2.1 Adicionar una planificación introduciendo los datos	- Se presiona el botón **Adicionar Planificación**. - Se introducen los datos

		correctamente al presionar el botón **Aceptar.**	correctamente. - Se presiona el botón **Aceptar**. - Se muestra la notificación "Se actualizó correctamente la Planificación"
		EP 2.2 Adicionar una planificación introduciendo datos inválidos al presionar el botón **Aceptar.**	- Se presiona el botón **Adicionar Planificación.** - Se introducen los datos inválidos. - Se presiona el botón **Aceptar**. - Se muestra la notificación "Existe campos obligatorios en blanco"
		EP 2.3 Adicionar una planificación dejando información requerida en blanco al presionar el botón **Aceptar.**	- Se presiona el botón **Adicionar Planificación.** - Se introducen los datos y se deja información requerida en blanco. - Se presiona el botón **Aceptar**. - Se muestra la notificación "Existe campos obligatorios en blanco"
		EP 2.4 Al presionar el botón **Cancelar.**	- Se presiona el botón **Adicionar Planificación.** - Se introducen los datos o no en los campos. - Se presiona el botón **Cancelar.**

Tabla 8: Escenario de flujos para la funcionalidad Modificar Planificación.

4.4.3 Caso de prueba para la funcionalidad Modificar Planificación

Nombre de la funcionalidad	Descripción General	Escenario de Pruebas	Flujo del Escenario
Modificar Planificación.	El sistema permite modificar una planificación que ya esté creada.	EP 2.1 Modificar una planificación introduciendo los datos correctamente al presionar el botón **Aceptar.**	- Se presiona el botón **Modificar Planificación**. - Se introducen los datos correctamente. - Se presiona el botón **Aceptar**. - Se muestra la notificación "Se actualizó correctamente la Planificación"
		EP 2.2 Modificar una planificación introduciendo datos inválidos al presionar el botón **Aceptar.**	- Se presiona el botón **Modificar Planificación.** - Se introducen los datos inválidos. - Se presiona el botón **Aceptar**. - Se muestra la notificación "Existe campos obligatorios en blanco"
		EP 2.3 Modificar una planificación dejando información requerida en blanco al presionar el botón **Aceptar.**	- Se presiona el botón **Modificar Planificación.** - Se introducen los datos y se deja información requerida en blanco. - Se presiona el botón **Aceptar**. - Se muestra la notificación "Existe campos obligatorios en blanco"

		EP 2.4 Al presionar el botón **Cancelar.**	- Se presiona el botón **Modificar Planificación.** - Se introducen los datos o no en los campos. - Se presiona el botón **Cancelar.**
		EP 2.5 Modificar una planificación introduciendo los datos correctamente al presionar el botón **Aplicar.**	- Se presiona el botón **Modificar Planificación**. - Se introducen los datos correctamente. - Se presiona el botón **Aplicar**. - Se muestra la notificación "Se actualizó correctamente la Planificación."
		EP 2.6 Modificar una planificación introduciendo datos inválidos al presionar el botón **Aplicar.**	- Se presiona el botón **Modificar Planificación.** - Se introducen los datos inválidos. - Se presiona el botón **Aplicar**. - Se muestra la notificación "Existe campos obligatorios en blanco" - Se presiona el botón **Aceptar.**
		EP 2.7 Modificar una planificación dejando información requerida en blanco al presionar el botón **Aplicar.**	- Se presiona el botón **Modificar Planificación.** - Se introducen los datos y se deja información requerida en blanco. - Se presiona el botón **Aplicar.**

			- Se muestra la notificación “Existe campos obligatorios en blanco” - Se presiona el botón **Aceptar.**

Tabla 8: Escenario de flujos para la funcionalidad Modificar Planificación.

4.5 Conclusiones

En este capítulo se realizó el diagrama de despliegue que indica la situación física de los componentes lógicos desarrollados, también se le realizó al software propuesto las pruebas de caja negra que se llevan a cabo sobre la interfaz del mismo. De esta manera los casos de prueba demostraron que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto. Demostrándose así que el sistema se encuentra en óptimas condiciones para el proceso de gestión de cuidados de examen.

CONCLUSIONES GENERALES

Con el desarrollo del trabajo, se ha confirmado la verdadera importancia de la implementación del sistema para la automatización de los procesos de gestión de cuidados de examen, pues permitirá gestionar de forma eficiente la asignación de profesores para dicho proceso.

Se estudió detalladamente el problema existente, estado del arte, y las necesidades reales que existían para lograr una buena planificación. Se sentaron las bases para la implementación del sistema, demostrando la importancia de entender qué es realmente lo que desea el cliente, y hasta qué punto se puede cumplir. Se utilizaron para el correcto funcionamiento de esta aplicación las metodologías, herramientas y lenguajes acordes a las necesidades del país de migrar al software libre.

Se cumplió el objetivo de la investigación, pues se detallaron las características del sistema, el análisis y diseño del mismo, la implementación y las pruebas para la automatización del proceso de gestión de cuidados de examen en la Facultad 15 de la Universidad de las Ciencias Informáticas, cumpliendo con todos los requisitos establecidos.

RECOMENDACIONES

- Hacerles pruebas pilotos al sistema a nivel de la facultad 15 de la Universidad de las Ciencias Informáticas.
- Tener en cuenta en el proceso de informatización de la Universidad.
- La funcionalidad Generar Distribución debe tener en cuenta otras tareas que le son asignadas a los profesores.

REFERENCIAS BIBLIOGRÁFICAS

[1] Sistema de Gestión Docente del centro de demostración TECNAUSA. [Online]. [Fecha de consulta 20 enero 2010]. Disponible en: http://www.tecnausa.com/?alias=ense.presentacion

[2] Generador de Horarios Para Centros de Enseñanza. [Online]. [Fecha de consulta 20 enero 2010]. Disponible en: http://www.penalara.com/ghc.asp.

[3] Sitio de descargas de software. [Online] [Fecha de consulta 22 enero 2010.] Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/.

[4] Pressman, Roger S. 2005. Ingeniería del Software. Un enfoque práctico. 2005.

[5] Espinoza, Humberto. 2005. Consultores Informáticos Integrales. PostgreSQL. [Online]. [Fecha de consulta 23 enero 2010]. Disponible en: http://www.openworldconsult.com.ve/psql.

[6] Colectivo de autores. 2008. Entorno Virtual de Aprendizaje. [Online]. [Fecha de consulta 23 enero 2010.] Disponible en: http://eva.uci.cu/mod/resource/view.php?inpopup=true&id=16179.

[7] CIBERAULA. Internet Information Servers 2006 [Online]. [Fecha de consulta 24 de enero de 2010] Disponible en: http:// www.ciberaula.com/curso/iis/que_es/

[8] Servidor HTTP Apache [Online]. [Fecha de consulta 6 de febrero de 2010]. Disponible en: http://es.wikipedia.org/wiki/Apache_http_server

[9] MaestrosdelWeb. 2008. MaestrosdelWeb. [Online]. [Fecha de consulta 8 de febrero 2010]. Disponible en: http://www.maestrosdelweb.com/principiantes/%C2%BFque-son-las-bases-de-datos/.

[10] Monteiro, Lázaro. DesarrolloWeb.com. [Online]. [Fecha de consulta 13 de Febrero de 2010.] Disponible en: http://www.desarrolloweb.com/articulos/26.php.

[11] Ingeniería de Software 1. "UML y RUP." UCI. Curso 2009_2010. Clase Teórico Práctica # 1.

[12] Bellows, Jeannie, Castek (2000). Activity Diagrams and Operation Architecture.

[13] Ingeniería de Software 1. UCI. Curso 2009_2010.

BIBLIOGRAFÍA

1. Spiral Universe. [Online]. [Fecha de consulta 28 enero 2010]. Disponible en: http://www.spiraluniverse.com/es/home.html
2. Zend Studio. [Online]. [Fecha de consulta 29 enero 2010]. Disponible en: http://www.tufuncion.com/zend-studio
3. Tutorial de PostgreSQL. [Online]. [Fecha de Consulta 30 enero 2010]. Disponible en: http://es.tldp.org/Postgresql-es/web/navegable/tutorial/tutorial.html
4. Sistema Gestor de base de datos SGBD. [Online]. [Fecha de consulta 30 enero 2010]. Disponible en: http://www.error500.net/garbagecollector/bases_de_datos/sistema_gestor_de_base_de_dato.html
5. Paradigm, Visual. [Online]. [Fecha de consulta 30 enero 2010]. Disponible en: http://www.visual-paradigm.com/product/vpuml/
6. Pressman, Roger S. 2005. Ingeniería del Software. Un enfoque práctico. 2005.
7. RUMBAUHG, J. 2000. El Lenguaje Unificado de Modelado. Manual de referencia. Madrid : s.n., 2000.
8. Sánchez, María A. Mendoza. 2004. Metodologías De Desarrollo De Software. Perú : s.n., 2004.
9. Torres, José Luis. 2009. Especificación de requisitos en Ingeniería de Software. [Online]. [Fecha de consulta 27 febrero 2010] Disponible en: http://www.uag.mx/ieee/contsep01/requerimientos.htm.
10. Sistema de Gestión Docente del centro de demostración TECNAUSA. [Online]. [Fecha de consulta 20 enero 2010]. Disponible en: http://www.tecnausa.com/?alias=ense.presentacion
11. Espinoza, Humberto. 2005. Consultores Informáticos Integrales. PostgreSQL. [Online]. [Fecha de consulta 23 enero 2010]. Disponible en: http://www.openworldconsult.com.ve/psql.
12. Ingeniería de Software 1. “UML y RUP.” UCI. curso 2009_2010. Clase Teórico Práctica # 1.
13. Flanagan, David (2002). JavaScript: The Definitive Guide, 4 ª Edición.

GLOSARIO

ANSI SQL 92: Estándar definido en el año 1992 por el Instituto Nacional Americano de Estándares (ANSI).

ASP: (Active Server Pages), Tecnología web para la creación de páginas web dinámicas creada por Microsoft Corporation.

ASP.NET: Plataforma para aplicaciones web desarrollada y comercializada por Microsoft Corporation. Se considera la evolución de ASP.

Benchmark: Es una técnica utilizada para medir el rendimiento de un sistema o componente del mismo.

C: Lenguaje de programación de alto nivel creado en 1972, en sus inicios orientado a la creación de Sistemas Operativos.

CASE: (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.

C++: Es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos.

Framework: Es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, en base a la cual otro proyecto de software puede ser organizado y desarrollado.

FTP: (File Transfer Protocol), Protocolo para transferencia de archivos. Pertenece a la familia de protocolos de internet. Se utiliza para intercambiar archivos entre computadoras conectadas a una red.

HTML: (Hypertext Markup Language), lenguaje de marcas de hipertexto. Usado para escribir documentos como páginas web.

HTTP: (Hypertext Transfer Protocol), Protocolo de transferencia de hipertexto. Protocolo que permite el intercambio de información variada entre clientes y servidores. Es el protocolo fundamental de la web.

Java: Es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90.

LINUX: Sistema operativo libre, altamente configurable. Se considera el producto insignia del Software Libre.

Log: Archivo de registro donde se almacena información sobre los determinados eventos que ocurren en una aplicación, sistema operativo o dispositivo.

PDA: (Personal Digital Assistant), en español Asistente Digital Personal, es un computador de mano originalmente diseñado como agenda electrónica con un sistema de reconocimiento de escritura.

Performance: Rendimiento, resultados.

Perl: Lenguaje de programación interpretado creado en 1987. Es ampliamente utilizado para crear páginas web dinámicas.

PHP: Es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas.

phpDocumentor: Es un generador de documentación de código abierto escrito en PHP.

PL/SQL: Lenguaje de programación incrustado en los gestores de bases de datos Oracle y PostgreSQL, soporta todas las características de SQL y añade nuevas funcionalidades.

RAM: (Ramdom Access Memory), Memoria de acceso aleatorio. Componente de una computadora digital que permite el almacenamiento rápido y volátil de datos.

SMTP: (Simple Mail Transfer Protocol), Protocolo simple de transferencia de correo. Protocolo para el intercambio de correo electrónico.

UML: (Unified Modeling Language), Lenguaje unificado de modelado. Lenguaje gráfico para modelar sistemas informáticos.

UNIX: Sistema Operativo creado en 1969. Es la base de muchos sistemas operativos modernos como Linux.

WINDOWS: Familia de sistemas operativos creados por Microsoft Corporation desde 1985 hasta la fecha.

ANEXOS

Anexo 1. Descripción de los principales casos de uso del sistema

CU-2	Gestionar profesores.
Actores	Jefe de departamento (inicia).
Propósito	Permitir al jefe de departamento gestionar los profesores del sistema.
Resumen	El caso de uso se inicia cuando el jefe de departamento selecciona la opción Gestionar profesores, el sistema le muestra las funcionalidades de gestión y el escoge la que se corresponde con la acción que vaya a realizar.
Referencias	RF2
Precondiciones	El jefe de departamento se debe haber autenticado con anterioridad.
Poscondiciones	Información de los profesores gestionada en la base de datos.
Escenario 1: Adicionar profesor	
Curso Normal de los Eventos	
Acción del Actor	**Respuesta del Sistema**
1 El jefe de departamento selecciona la opción Adicionar profesor.	2 El sistema muestra una interfaz con los datos a llenar.
3 El jefe de departamento ingresa los datos solicitados por el sistema y oprime el botón “Adicionar”.	4 El sistema verifica que los campos del formulario no estén vacíos.

	5 El sistema verifica que el profesor no esté registrado en la base de datos.
	6 El sistema registra el profesor en la base de datos correspondiente.
Curso Alternativo de los Eventos	
	4.1 Muestra un mensaje de error indicando que debe llenar todos los campos (“llene todos los campos”) y da la posibilidad nuevamente de llenar los campos requeridos.
	5.1 Muestra un mensaje de error indicando que el profesor ya existe.
Escenario 2: Modificar professor	
Curso Normal de los Eventos	
1 El jefe de departamento selecciona un profesor y luego la opción Modificar profesor.	2 El sistema muestra un formulario con los campos a modificar del profesor seleccionado.
3 El jefe de departamento modifica los campos deseados y oprime el botón “Aplicar” y luego “Aceptar”.	4 El sistema verifica que los campos modificados estén correctos.
	5 El sistema modifica los datos del usuario seleccionado.
Curso Alternativo de los Eventos	
	4.1 El sistema muestra un mensaje indicando el

	error.
Escenario 3: Eliminar professor	
Curso Normal de los Eventos	
1 El jefe de departamento selecciona un profesor y luego la opción Eliminar profesor.	2 El sistema elimina el profesor seleccionado y actualiza la base de datos.
Curso Alternativo de los Eventos	
	2.1 Se muestra un mensaje de confirmación antes de eliminar definitivamente.
Prioridad	Crítico

Tabla 9: Descripción del caso de uso del sistema Gestionar profesores.

CU-4	Generar distribución.
Actores	Jefe de Departamento (inicia)
Propósito	Permitir al Jefe de departamento generar la distribución de los profesores que cuidarán un determinado examen.
Resumen	El caso de uso se inicia cuando el Jefe de departamento selecciona una planificación creada y luego oprime el botón “Distribución” que genera la misma.
Referencias	RF4
Precondiciones	El Jefe de departamento debe estar autenticado en el sistema.

Poscondiciones	Generar la distribución de profesores de una determinada planificación de un examen.
Curso Normal de los Eventos	
Acción del Actor	**Respuesta del Sistema**
1 Selecciona una planificación y oprime el botón “Distribución”.	2 El sistema genera la distribución de profesores para cuidar el examen seleccionado y muestra los datos.
Curso Alternativo de los Eventos	
	2.1 Si no existen planificaciones adicionadas en la base de datos, el sistema no puede generar la distribución.
Prioridad	Crítico

Tabla 10: Descripción del caso de uso del sistema Generar distribución.

CU-9	Gestionar jefe de departamento.
Actores	Administrador (inicia).
Propósito	Gestionar los jefes de departamentos
Resumen	El caso de uso se inicia cuando el administrador selecciona la opción Gestionar jefe de departamento y el sistema le muestra las opciones de gestión.
Referencias	RF9
Precondiciones	El administrador debe estar registrado en el sistema.

Poscondiciones	Gestionar los jefes de departamentos.
Escenario 1: Adicionar jefe de departamento	
Curso Normal de los Eventos	
Acción del Actor	**Respuesta del Sistema**
1 El administrador selecciona la opción Adicionar jefe de departamento.	2 El sistema muestra un formulario para entrar los datos del jefe de departamento.
3 El administrador ingresa los datos solicitados por el sistema y oprime el botón “Aceptar”.	4 El sistema verifica que los campos del formulario no estén vacíos.
	5 El sistema verifica que el jefe de departamento no esté registrado en la base de datos.
	6 El sistema registra el jefe de departamento en la base de datos correspondiente.
Curso Alternativo de los Eventos	
	4.1 Muestra un mensaje de error indicando que debe llenar todos los campos y da la posibilidad nuevamente de llenar los campos requeridos.
	5.1 Muestra un mensaje de error indicando que el jefe de departamento ya existe.
Escenario 2: Modificar jefe de departamento	
Curso Normal de los Eventos	

1 El administrador selecciona un jefe de departamento y luego la opción Modificar.	2 El sistema muestra un formulario con los campos a modificar.
3 El administrador modifica los campos deseados y oprime el botón “Aceptar”.	4 El sistema verifica que los campos modificados estén correctos.
	5 El sistema modifica los datos del jefe de departamento seleccionado.
Curso Alternativo de los Eventos	
	4.1 El sistema muestra un mensaje indicando el error.
Escenario 3: Eliminar jefe de departamento	
Curso Normal de los Eventos	
1 El administrador selecciona un jefe de departamento y luego la opción Eliminar.	2 El sistema elimina el jefe de departamento seleccionado y actualiza la base de datos.
Curso Alternativo de los Eventos	
	2.1 Se muestra un mensaje de confirmación antes de eliminar definitivamente.
Prioridad	Crítico

Tabla 12: Descripción del caso de uso del sistema Gestionar jefe de departamento.

Anexo 2. Diagramas de Clases de Análisis para Casos de Usos

Figura 11: Diagrama de clase de análisis del caso de uso Autenticar usuario.

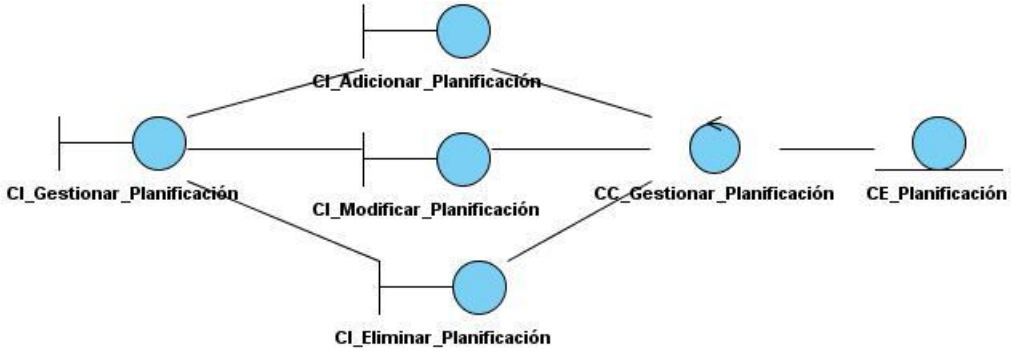


Figura 13: Diagrama de clase de análisis del caso de uso Gestionar planificación.

Figura 14: Diagrama de clase de análisis del caso de uso Generar distribución.

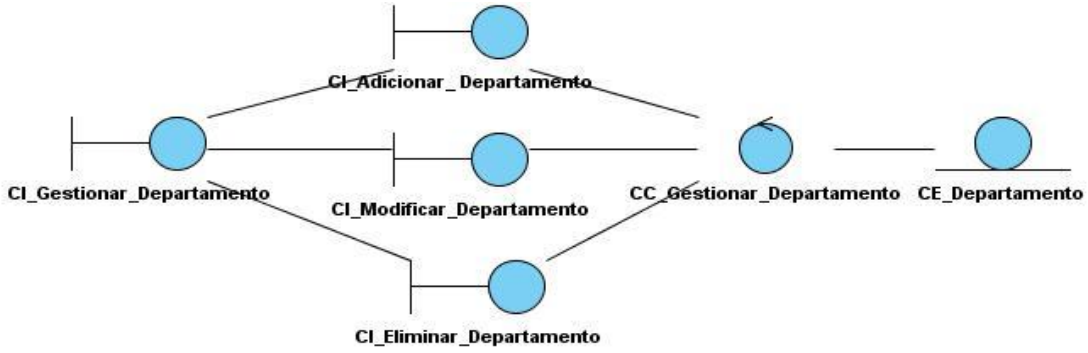


Figura 16: Diagrama de clase de análisis del caso de uso Gestionar departamento.

Anexo 3. Diagramas de interacción (Diagramas de Secuencia)

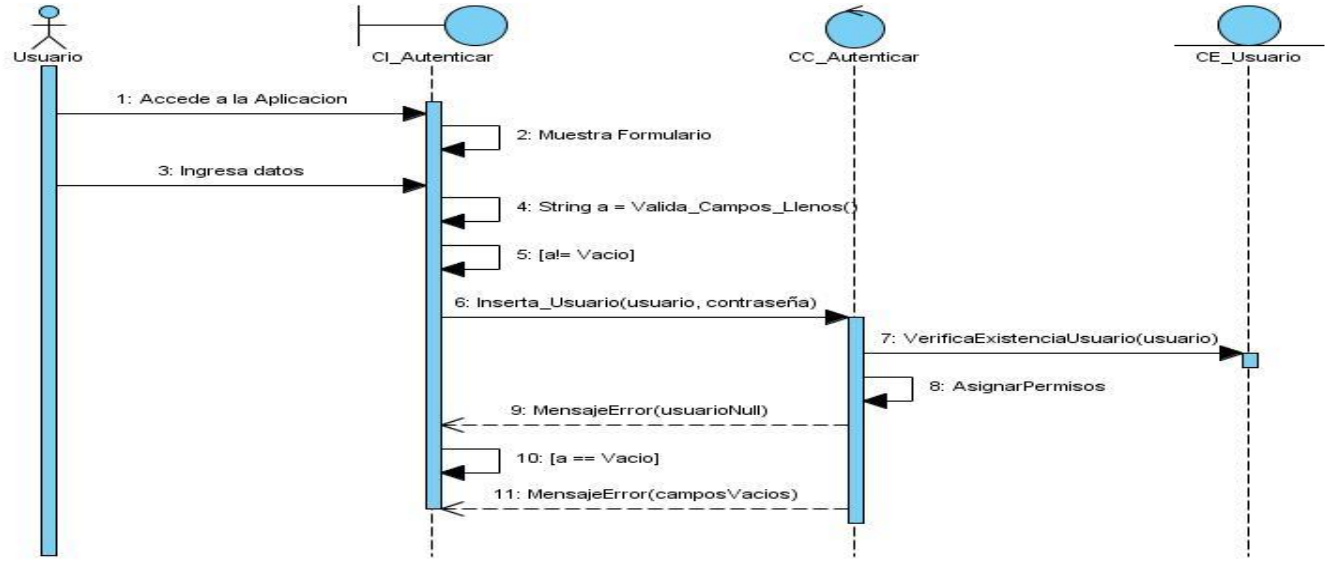


Figura 18: Diagrama de secuencia del caso de uso Autenticar usuario.

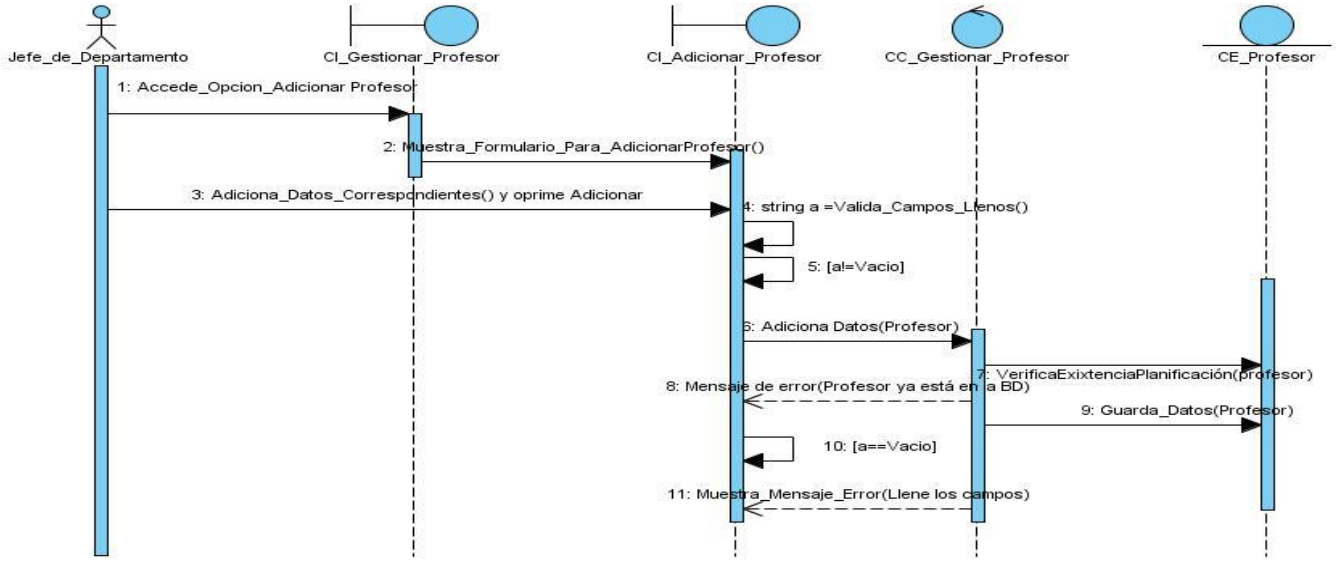


Figura 19: Diagrama de secuencia del caso de uso Gestionar profesor (Adicionar profesor).

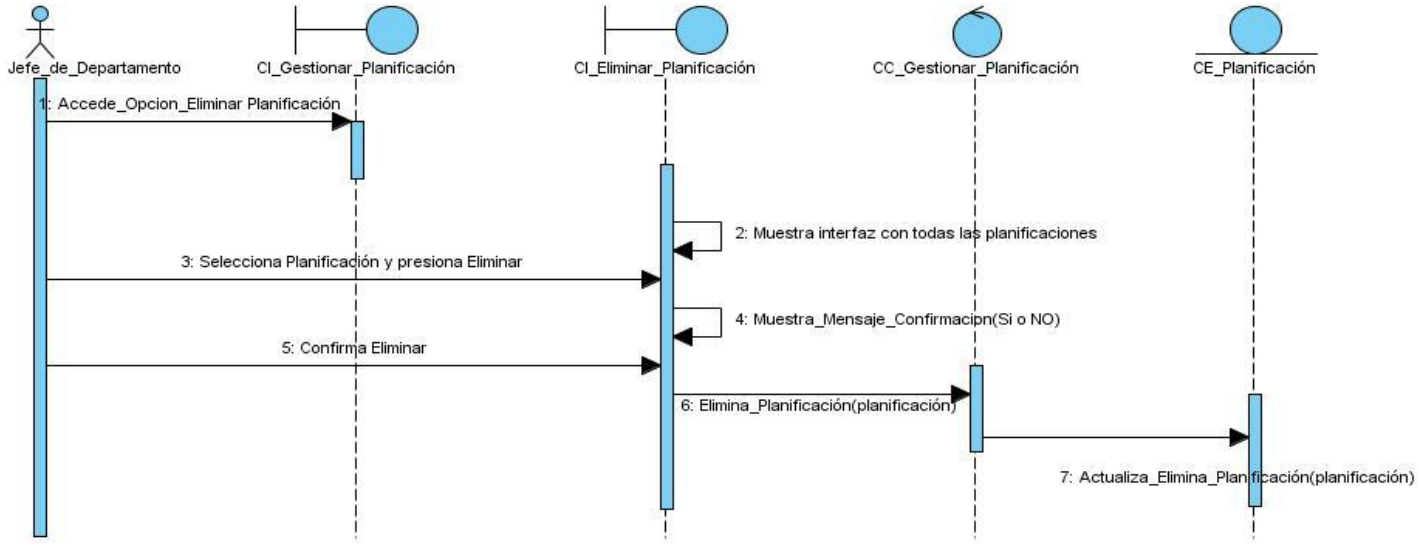


Figura 22: Diagrama de secuencia del caso de uso Gestionar planificación (Eliminar planificación).

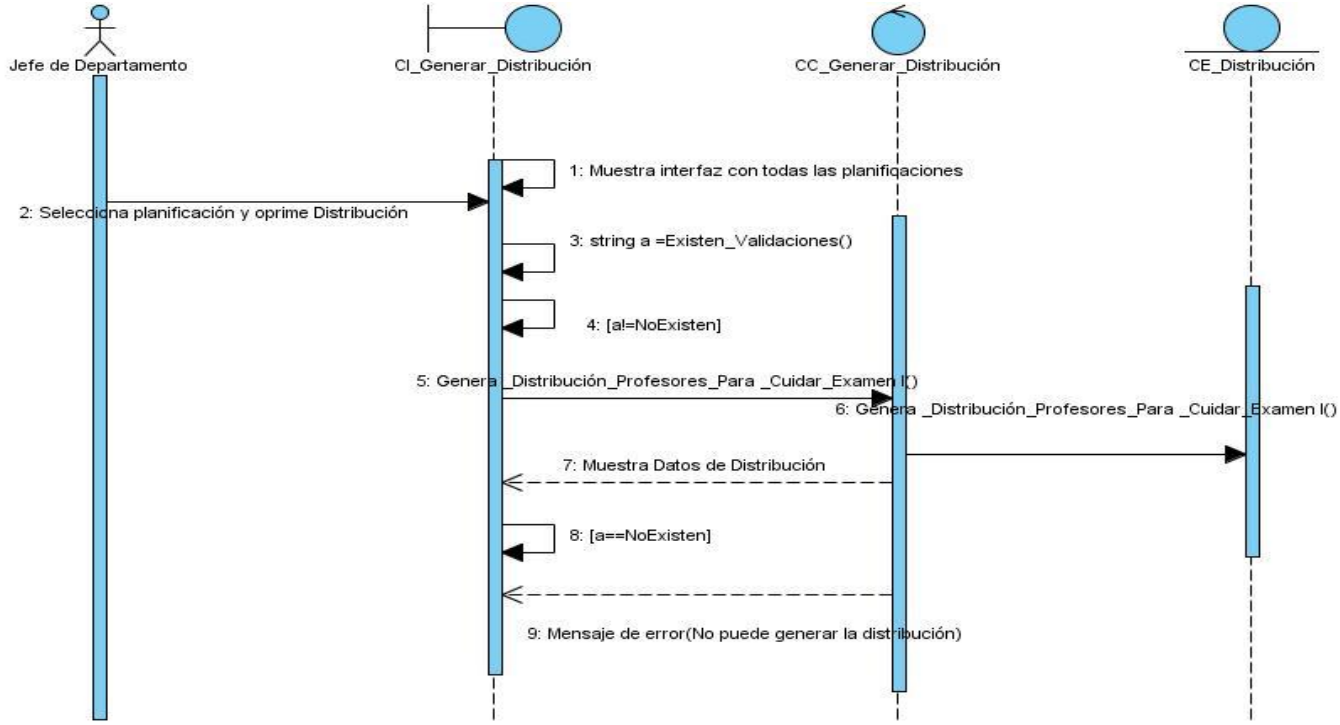


Figura 23: Diagrama de secuencia del caso de uso Generar distribución.

Anexo 4. Diagramas de Clases de Diseño

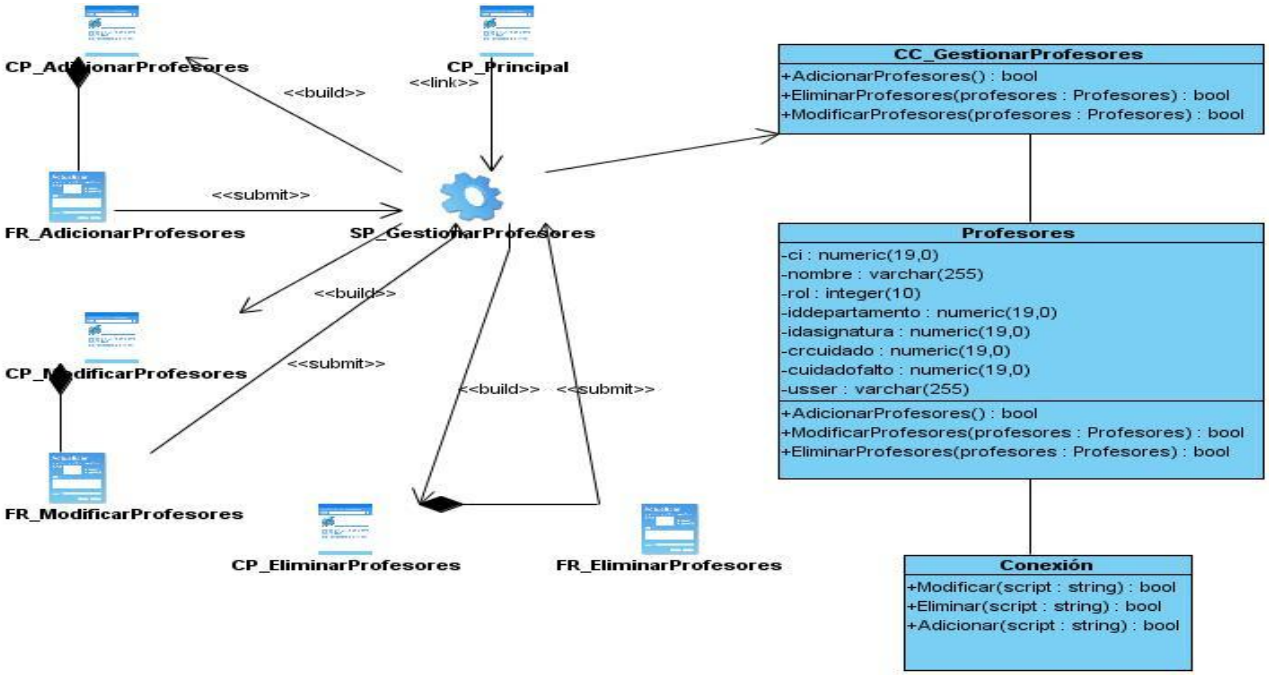


Figura 25: Diagrama de clases de diseño del caso de uso Gestionar profesor.

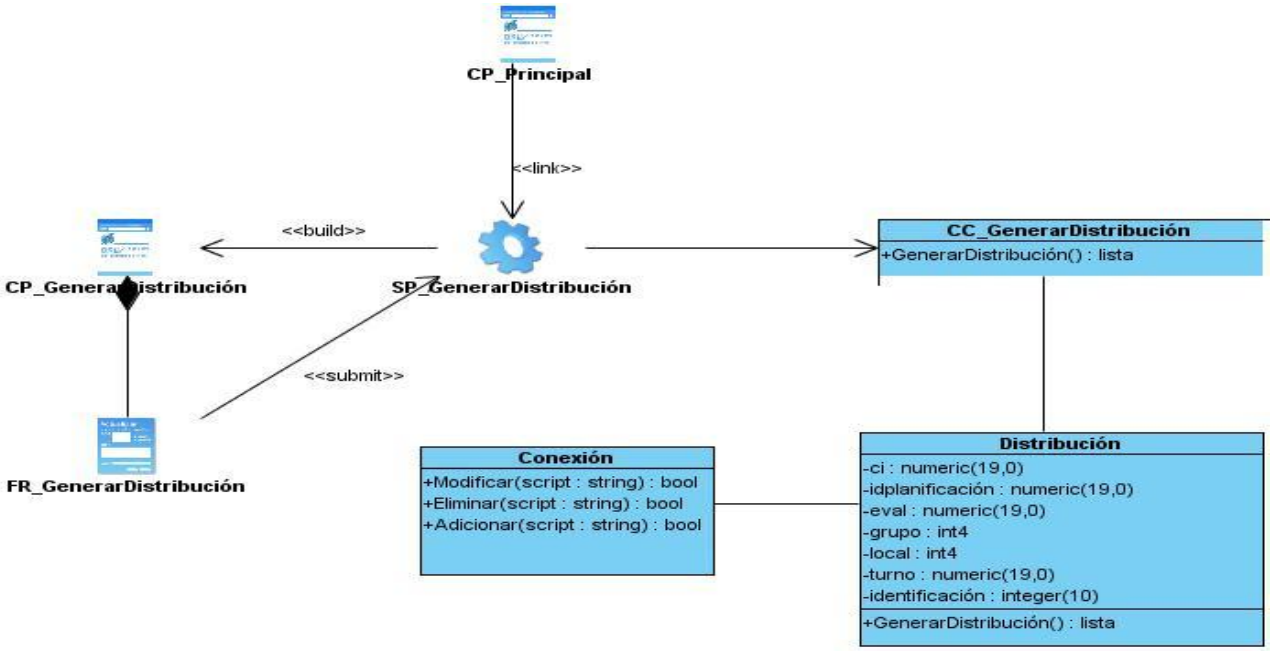


Figura 27: Diagrama de clases de diseño del caso de uso Generar distribución.

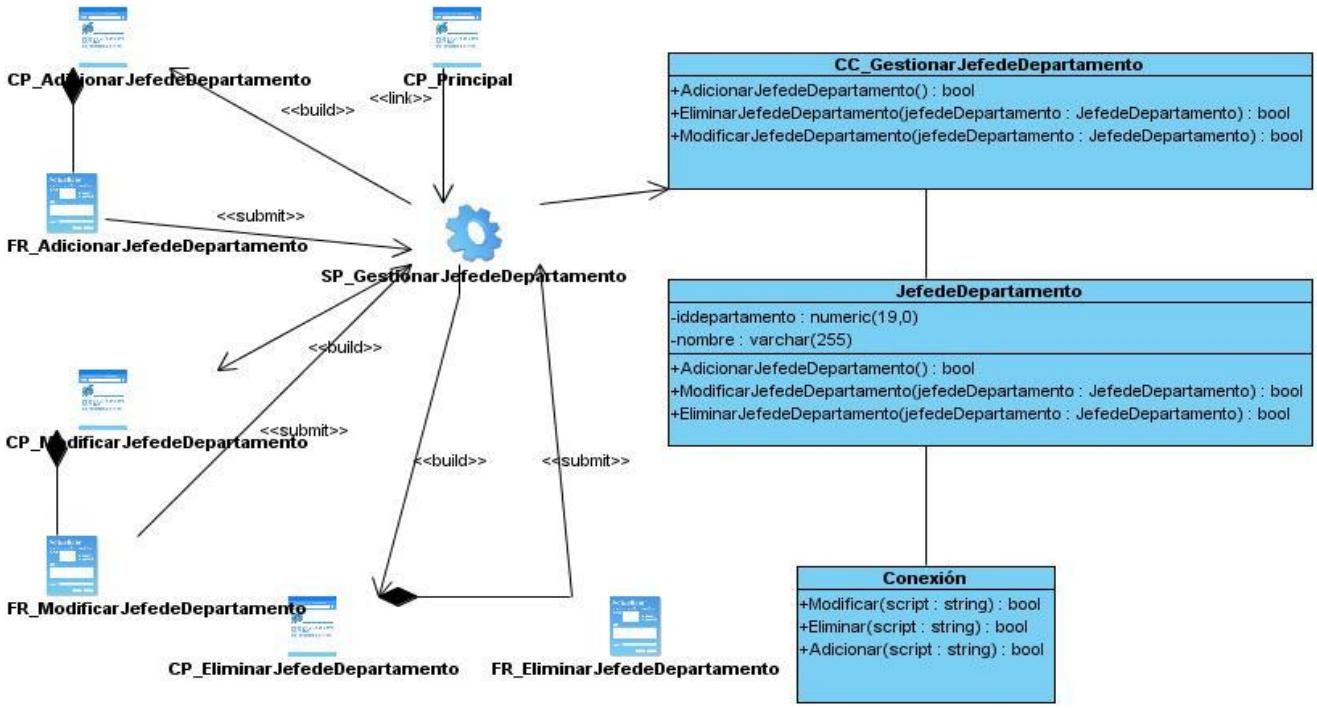


Figura 29: Diagrama de clases de diseño del caso de uso Gestionar jefe de departamento.

Anexo 5. Descripción de las tablas de Modelo de Datos

Nombre: Departamento		
Descripción: En esta tabla se guarda el nombre de todos los departamentos		
Atributo	Tipo	Descripción
iddepartamento	numeric(19,0)	Identificador de la tabla departamento.
nombre	varchar(255)	Nombre del departamento.

Tabla 13: Descripción de la tabla del Modelo de Datos (Departamento).

Nombre: Profesores		
Descripción: En esta tabla se guardan datos de todos los profesores		
Atributo	Tipo	Descripción
ci	numeric(19,0)	Número de solapín del profesor.
nombre	varchar(255)	Nombre del profesor.
rol	int4	Pribilegio para navegar en la aplicación.
iddepartamento	numeric(19,0)	Identificador del departamento.
idasignatura	numeric(19,0)	Identificador de la asignatura.
crcuidado	numeric(19,0)	Cantidad de cuidados asignados a un profesor.

cuidadofalto	**numeric(19,0)**	**Cantidad de faltas a un cuidado de un profesor.**
user	**varchar(255)**	**Usuario del profesor.**
número	**varchar(255)**	**Número del local.**
idgrupo	**integer(10)**	**Identificador de la tabla grupo.**

Tabla 14: Descripción de la tabla del Modelo de Datos (Profesores).

Nombre: Asignaturas		
Descripción: En esta tabla se guarda el nombre de todas las asignaturas		
Atributo	**Tipo**	**Descripción**
idasignatura	**numeric(19,0)**	**Identificador de la tabla asignatura.**
nombre	**varchar(255)**	**Nombre de la asignatura.**
anno	**numeric(1,0)**	**Año al que el profesor imparte clases.**

Tabla 15: Descripción de la tabla del Modelo de Datos (Asignaturas).

Nombre: Locales		
Descripción: En esta tabla se guarda la información referente a los locales		
atributo	**Tipo**	**Descripción**
número	**varchar(255)**	**Número del local para orientarse.**

capacidad	**Numeric**	**Cantidad de personas que pueden estar en el local.**
idplanificación	**numeric(19,0)**	**Identificador de la tabla planificación.**

Tabla 16: Descripción de la tabla del Modelo de Datos (Locales).

Nombre: Grupo		
Descripción: En esta tabla se guarda el nombre de todas las asignaturas		
Atributo	**Tipo**	**Descripción**
idgrupo	**Integer**	**Identificador de la tabla grupo.**
número	**numeric(19,0)**	**Número del grupo.**
anno	**numeric(1,0)**	**Año que cursan los estudiantes.**
capacidad	**Numeric**	**Cantidad de estudiantes del grupo.**
idplanificación	**numeric(19,0)**	**Identificador de la tabla planificación.**

Tabla 27: Descripción de la tabla del Modelo de Datos (Grupo).