UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 3

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

Análisis y Diseño del Sistema de Ayuda para la planificación docente en la facultad 3.

Autores:

Ramón Sixto Ramos Rodríguez
Alfredo Saltaren Montiel

Tutor:

Lic. Rolan Rober Bullaín Diéguez

Ciudad de La Habana, Cuba.

Junio, 2008

DECLARACIÓN DE AUTORÍA.

Declaramos ser autores de la presente tesis y reconocemos 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 2008.

Ramón Sixto Ramos Rodríguez

_____

(Autor)

Alfredo Saltarén Montiel

_____

(Autor)

Lic. Rolan Rober Bullaín Diéguez

_____

(Tutor)

Agradecimientos.

Quisiéramos agradecer ante todo al Comandante en Jefe Fidel Castro y a la Revolución por habernos dado la posibilidad de estar hoy aquí y hacernos soñar cada día con un futuro mejor para nuestro país y la humanidad.

A nuestro Tutor Lic. Rolan Rober Bullaín Diéguez por encargarse de nosotros en el momento más difícil de nuestra tesis, por sus consejos y ayuda siempre útiles.

A nuestros padres por ser nuestra inspiración eterna y por sabernos enseñar el camino a seguir en la vida.

A todos los que de una forma u otra nos apoyaron para realizar este trabajo.

Dedicatoria de Ramón Sixto.

A mis Padres, por su amor y confianza

A mi madre, porque sencillamente se lo debo todo en esta vida, la mejor del mundo

A mi padre, por sus consejos, por apoyarme y comprenderme siempre

A mi hermano, por apoyarme en los momentos difíciles

A mi Abuelo Papá Rodríguez, por ser como un padre para mí

A mi Abuela Mamá Arelis, por su eterno cariño

A Tita, por quererme tanto

A mi abuelo Mon, aunque no esté con nosotros, sé que estaría orgulloso

A mis Tías y Tío

A mis primos todos: Nenita, Ramirito, Yanet, Enmanuel, Meylín y Yasmín

Dedicatoria de Alfredo.

Quisiera agradecer a mis padres que son lo que más quiero en esta vida, a mi familia en especial a mi abuelo con el que no pude estar en sus últimos días, a todos los que me ayudaron en la tesis, a mis amistades, a Rolan, a Pedro Piñeiro y su esposa por su apoyo. A todos por comprenderme, por convivir con mis defectos que son muchos y por darme razones para seguir adelante.

Resumen.

La planificación docente siempre ha resultado ser un proceso complejo para los grandes centros universitarios, los cuales tienen que lidiar con una vorágine de situaciones y problemas provocados por el dinamismo que impone una sociedad cada vez más informatizada en la que los cambios deben ser resueltos con eficiencia y rapidez para no afectar el proceso docente. En la Universidad de las Ciencias Informáticas esta situación se acentúa si se tiene en cuenta que la misma constituye un centro educacional atípico, de corte productivo y con una matrícula de estudiantes altamente variable.

El presente trabajo esta centrado en el análisis y diseño de un sistema de ayuda para la planificación docente en la Facultad 3, destinado a mejorar la eficiencia del proceso de planificación en la facultad teniendo en cuenta las características específicas de la misma. Para ello se realizó un estudio acerca de conceptos y factores que influyen en la planificación. También se llevó a cabo un análisis y selección de herramientas de desarrollo de software a utilizar, quedando finalmente plasmados los resultados del Análisis y Diseño y recomendándose algunos aspectos para el mejoramiento futuro del sistema.

Tabla de Contenidos.

AGRADECIMIENTOS. II

DEDICATORIA DE RAMÓN SIXTO. III

DEDICATORIA DE ALFREDO. IV

RESUMEN. V

TABLA DE CONTENIDOS. VI

INTRODUCCIÓN. 1

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA. 4

1.1. Introducción. 4

1.2. La planificación docente. 4

1.2.1. ¿En que consiste la planificación docente?. 4

1.2.2. Factores que inciden y participan en la planificación docente. 4

1.2.3. La Planificación docente en la universidad. 5

1.2.4. Sistemas de planificación académica existentes. 6

1.3. Metodologías para el desarrollo de Software. 7

1.3.1. Rational Unified Process 7

1.3.2. Programación Extrema. 9

1.3.3. Microsoft Solution Framework. 11

U1.3.4. Fundamentación de la selección de la metodología de desarrollo de software a utilizar. 11

1.4. Lenguajes de modelado. 12

1.5. Selección de herramientas a utilizar en el sistema. 13

1.5.1. Herramientas CASE. 13

1.6. Lenguajes de Programación y Tecnologías del lado del Cliente. 15

1.7. Lenguajes de Programación y Tecnologías del lado del Servidor. 16

1.7.1. Sistemas Gestores de Bases de Datos. 17

1.7.2. Servidores Web. 20

1.8. Análisis y diseño de sistemas. 21

1.8.1. Análisis. 21

1.9. Diseño. 22

1.9.1. Principios del diseño. 23

1.10. Patrones de diseño 23

1.10.1. ¿Porqué usar patrones?. 23

1.11. Patrones a utilizar en el trabajo. 24

1.11.1. Modelo-Vista-Controlador (MVC). 24

1.11.2. Patrones GRASP. 25

1.11.3. Patrones GoF. 26

1.11.4. Patrones de casos de uso. 28
1.12. Requerimientos. 28
1.12.1. Ingeniería de requisitos. 29
1.12.2. Actividades de la Ingeniería de requisitos. 29
1.13. Programación por Capas. 30
1.14. Conclusiones Parciales. 32
CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA. **34**
2.1. Introducción. 34
2.2. Aplicación al sistema de las principales actividades de la IR. 34
2.3. Principales procesos objeto de automatización. 36
2.3.1. Descripción general de los procesos. 36
2.4. Datos a tratados en el sistema. 37
2.5. Propuesta de Sistema. 37
2.6. Modelo del Negocio. 38
2.6.1. Reglas del Negocio. 38
2.6.2. Actores del Negocio. 38
2.6.3. Trabajadores del Negocio. 39
2.7. Casos de uso del Negocio. 39
2.7.1. Casos de uso del negocio objeto de estudio. 40
2.7.2. Entidades del Negocio. 40
2.7.3. Descripción de los casos de uso del negocio. 40
2.7.4. Diagrama de casos de uso del negocio. 48
2.7.5. Modelo de objetos del Negocio. 48
2.7.6. Diagrama de clases del Modelo de objetos del Negocio. 49
2.8. Diagrama de Actividades 49
2.9. Requisitos funcionales y no funcionales del Sistema. 49
2.9.1. Requisitos Funcionales. 50
2.9.2. Requisitos no Funcionales. 52
2.10. Actores del sistema de ayuda para la docencia. 54
2.11. Diagrama de Casos de uso del Sistema. 56
2.12. Análisis de las características del Sistema de Ayuda. 60
2.12.1. Análisis de los requerimientos. 61
2.12.2. Métricas para determinar la ambigüedad. 61
2.12.3. Métrica para determinar el número de requisitos que no son considerados en ningún CU. 62
2.12.4. Esquema QFD (Quality Function Deployment). 62
2.14.5. Métricas para la evaluación de la calidad del diagrama de CU. 63
2.15. Conclusiones Parciales. 66

CAPÍTULO III: ANÁLISIS Y DISEÑO DEL SISTEMA..... 67

3.1. Introducción. 67

3.2. Modelo de análisis..... 67

3.2.1. Clases de análisis..... 67

3.2.2. Modelo de clases del análisis..... 68

3.2.3. Diagrama de clases del análisis. 68

3.3. Diagramas de Interacción. 72

3.3.1. Diagramas de Secuencia..... 72

3.4. Modelo de Diseño..... 73

3.4.1. Diagrama de clases del Diseño. 73

3.5. Extensiones UML para Web utilizadas en el sistema. 74

3.6. Patrones aplicados en el Sistema de Ayuda. 74

3.7. Capas utilizadas en el Sistema. 77

3.8. Descripción de la arquitectura propuesta..... 78

3.9. Diagrama de clases Persistente..... 79

3.10. Conclusiones Parciales. 80

CONCLUSIONES. 81

RECOMENDACIONES. 82

BIBLIOGRAFÍA. 83

REFERENCIAS BIBLIOGRÁFICAS..... 84

GLOSARIO DE TÉRMINOS. 85

Introducción.

En la actualidad, es incuestionable el impacto que genera en la sociedad el uso de las tecnologías de la informática y las comunicaciones, estas potencian la automatización de muchos de los procesos que se ejecutan manualmente en diversas instituciones, obteniendo en un período de tiempo relativamente corto, una eficiencia casi exponencial. A esta realidad no escapa la vida universitaria, los centros universitarios más prestigiosos del mundo utilizan intensivamente las TIC's para hacer frente sus problemas de planificación docente, proceso intrínsecamente complejo, sobre todo en las grandes universidades donde es necesario planificar cientos de diferentes actividades docentes en un marco de tiempo estrecho. En Cuba, en las principales instituciones de la educación superior, el proceso de planificación docente es en su mayoría desarrollado manualmente.

La Universidad de las Ciencias Informáticas (UCI) constituye el centro universitario que más tecnología de avanzada incorpora en su sistema docente en Cuba, contando con una red de miles de computadoras a través de las cuales los profesores y estudiantes tienen acceso a casi cualquier información nacional e internacional en cualquier momento. Tratando de llevar el proceso de enseñanza-aprendizaje al unísono con las últimas tendencias de las tecnologías educativas y con el empleo intensivo de las TIC's. El volumen de actividades, derivado del modelo docencia-producción, fusionado con la tarea de informatizar la sociedad, contribuir a la economía mediante el desarrollo de software, cantera en la batalla político-ideológica que libra el país y actividades extensionistas imponen una dinámica poco común; dinamismo que afecta e influye en cada uno de los procesos que se desarrollan en el centro, dentro de ellos la planificación docente.

En el caso particular de la Facultad 3 de la UCI, el planificador de los horarios docentes realiza la actualización de los horarios de forma manual, muchas veces sobrescribiendo la misma planilla varias veces. El personal docente tiene que dirigirse a la oficina para realizar la reservación de un local, lo cual aparte de sobrecargar a la secretaria del vicedecano de docencia afecta el proceso de actualización, calidad, distribución de los locales, planificación de asignaturas así como el flujo de información sobre los cambios a los estudiantes. Mecanismo totalmente deficiente comparado con la dinámica que impone la universidad.

Por todo lo antes explicado se plantea la necesidad de crear un sistema que automatice la gestión y el control de la información referente a la planificación docente y de esta forma automatizar el proceso de confección de horarios en la facultad. Con este objetivo se tratará específicamente el Análisis y el Diseño del sistema propuesto. En este caso se puede resumir que la **situación problemática** gira en torno a que el mecanismo actualmente existente no permite agilizar los procesos de planificación docente en la facultad, al mismo tiempo que constituye no solo una mala forma de trabajar la distribución y actualización de los locales docentes sino que representa una sobrecarga de trabajo para el planificador que no este especializado en este trabajo; así como gastos de materiales de

oficina y de tiempo útil de trabajo de la dirección de la facultad en lo que a docencia se refiere. Para resolver esta problemática no existe todavía una solución Informática que haga viable la gestión de estos procesos en la facultad, todo esto de una forma permanente y que permita el intercambio de la información con el personal docente en general.

Ante la situación problemática antes planteada se define el siguiente **problema**: ¿Cómo elaborar el análisis y diseño de un sistema de ayuda para la planificación docente en la Facultad 3, que potencie la mitigación de errores y agilice los procesos? Por consiguiente el **objeto de estudio** de esta investigación comprende el proceso de desarrollo de software y los procesos de gestión y control de la información referente la planificación docente. El **campo de acción** esta orientado al análisis y diseño en los procesos de planificación docente. Por tanto se define como **objetivo general** desarrollar el análisis y diseño del sistema de ayuda para la planificación docente en la Facultad 3.

Concretándose la **hipótesis** de la investigación como: Si se desarrolla el Análisis y el Diseño del sistema de ayuda para la planificación docente, entonces los programadores podrán implementar el sistema que potenciará un proceso de planificación docente más eficiente mitigando el efecto negativo que implica la planificación manual.

Para dar cumplimiento al objetivo general planteado se proponen los siguientes **objetivos específicos**:

- Realizar la especificación de requisitos para el sistema.
- Realizar el modelado de casos de uso del Negocio.
- Analizar los casos de uso del sistema por paquetes.
- Elaborar la actividad de diseño del sistema.
- Elaborar la actividad del análisis del sistema.

Para llevar a cabo este trabajo y cumplir con los objetivos específicos propuestos se concibieron las siguientes **tareas de investigación**.

- Realización el estudio del estado de arte de las principales tendencias del diseño de software.
- Realización el estudio del estado del arte sobre software de gestión y planificación académica.
- Selección de la metodología para el diseño del sistema de acuerdo a la tecnología seleccionada.
- Captura de requerimientos que respondan a las necesidades del sistema.
- Descripción de los casos de uso del sistema.
- Realización del diagrama de casos de usos del sistema por paquetes.
- Estudio de los principales patrones a tratar en el sistema.

- Realización de una propuesta de diseño acorde con las condiciones planteadas en el análisis.
- Realización de los diagramas de clases del diseño.

Los métodos y técnicas de investigación utilizados fueron los siguientes:

Histórico - Lógico:

Utilizado para realizar el analizar de la trayectoria de la metodología de desarrollo de software y herramientas que se utilizarán en el proyecto.

Método de la modelación:

Empleado en la construcción de modelos para explicar las principales estructuras del sistema a desarrollar.

Métodos Empíricos:

La entrevista para recopilar la información y principales características de los procesos de negocio al cual responde el sistema.

Para un mejor entendimiento del trabajo se decidió estructurar el mismo en tres capítulos los cuales se resumen a continuación: "*Capítulo 1*: *Fundamentación Teórica"*, se abordan los principales conceptos y tendencias referentes a la planificación académica en el ámbito internacional y nacional. Se realiza un estudio del estado del arte de las posibles técnicas y herramientas a utilizar en el trabajo, realizándose consecuentemente la selección de las más adecuadas y argumentando el por qué de la selección. "*Capítulo 2: Modelación del Sistema"*, se estudian las principales características que tendrá el sistema tales como los requisitos funcionales y no funcionales, la arquitectura de la aplicación, así como las funcionalidades de la misma, se establecen también los casos de uso y sus descripciones. "*Capítulo 3: Análisis y Diseño"*, se expone el análisis del sistema basándose en una detallada descripción de las diferentes funcionalidades para una mejor comprensión de las mismas facilitando el trabajo de los desarrolladores y mediante el diseño del sistema se ofrece una visión de cómo será el proyecto que proponemos.

Capítulo I: Fundamentación Teórica.

1.1. Introducción.

En el presente capítulo se tratarán los principales conceptos relacionados con el tema que se propone, se explica la existencia y surgimiento de sistemas similares en la actualidad en el área internacional, las características de las principales tecnologías y herramientas existentes utilizadas para el desarrollo de estas aplicaciones, también se abordará acerca de las diferentes metodologías y se hará una valoración de las ventajas y desventajas de cada elemento tratado llegando finalmente a la conclusión de qué es lo que se debe usar en la solución propuesta.

1.2. La planificación docente.

1.2.1. ¿En que consiste la planificación docente?

La planificación implica un proceso consciente de estudio y selección del mejor curso de acción a seguir, estableciendo prioridades frente a una variedad de alternativas posibles y factibles de acuerdo a los recursos con que se cuenta.

La planificación docente no es más que la organización coherente y funcional de un conjunto de actividades destinadas al eficaz aprendizaje de los estudiantes tratando siempre de lograr el óptimo uso de los recursos disponibles en un marco de tiempo adecuado.

1.2.2. Factores que inciden y participan en la planificación docente.

Se aborda el tema de la planificación concibiéndola como la primera función administrativa que sirve de base para muchas otras funciones, la cual determina por anticipado cuáles son los objetivos que deben cumplirse y qué debe hacerse para alcanzarlos; por tanto, es un modelo teórico para actuar en el futuro. La planificación comienza por establecer los objetivos y detallar los planes necesarios para alcanzarlos de la mejor manera posible. Determina a dónde se pretende llegar, qué debe hacerse además del cómo, cuándo y en qué orden deben suceder los acontecimientos.

Luego de un estudio realizado, se concibe dicha actividad como un proceso complejo, pues se deben tener en cuenta muchos aspectos de singular importancia referentes a la actividad docente, ya que es mediante esta donde se logra la asimilación de conocimientos científicos y la formación de habilidades correspondientes, objetivo y resultado según Maria Onelia Chiang Molina en su libro "Higiene de la actividad docente", esenciales de la propia actividad para los educandos, la cual es

una forma de actividad cognoscitiva dirigida mediante el proceso de enseñanza en la escuela, regida por un conjunto de características que le dan elementos de complejidad, entre los que encontramos:

- Basada en contenidos previamente determinados en el plan de estudio de la carrera y en programas establecidos.
- Puede hacerse por bloques lectivos, ciclos o niveles, en dependencia de la carrera, el curso o el nivel de enseñanza.

Unido a esto debe considerarse la organización de la actividad docente desde el punto de vista higiénico, la cual está muy relacionada con uno de los principios más importantes de la organización científica del trabajo: el principio de la optimización, el cual exige la selección de una variante óptima de la actividad para evitar trastornos a los estudiantes. Esta organización supone:

- La existencia de un balance de carga docente normalizado (equilibrio de clases para un grupo de estudiantes en un período).
- Equilibrio de los tipos de actividades docentes (exámenes, conferencias, clases prácticas, seminarios, laboratorios, etc.), pues está comprobado que muchas actividades evaluativas alteran los biorritmos de aprovechamiento académico, de alimentación, sueño y descanso, llegando a ocasionar alteraciones en el calibre de los vasos sanguíneos y en la respiración.
- Planificación adecuada de medios y materiales de enseñanza, así como de volumen de la información (Molina, 1995).

1.2.3. La Planificación docente en la universidad.

Históricamente, en la solución a muchas situaciones y problemas organizativos a los que se ha enfrentado el hombre, ha sido de especial importancia su capacidad para planificar las actividades. El hecho de conducir un proceso se evidencia en cualquier área o sector de la sociedad, desde la producción industrial hasta la educación en todos sus niveles.

Las Universidades forman el personal capacitado para asumir tareas específicas de diversa índole, que son animadas por procesos que tienen lugar en su interior, llamados procesos sustantivos, destacándose entre estos el Proceso de Enseñanza Aprendizaje(PEA), que tiene carácter sistémico, organizado y planificado por un personal especializado con la finalidad de formar al profesional que requiere la sociedad.

El PEA en su organización como sistema, requiere de una distribución adecuada de las diferentes asignaturas: por semestres, semanas, días y horas lectivas, siempre teniendo en cuenta las exigencias propias de las disciplinas que se imparten a los estudiantes y su importancia para la especialidad, así como las características de cada grupo de educandos, el tipo de enseñanza, la forma de organización de sus actividades docentes y las actividades extra - docentes que se

realizan, ya que estos factores, y otros, afectan directamente la manera en que han de ser distribuidos los contenidos a lo largo de la formación del profesional: elementos claves para lograr una planificación balanceada de acuerdo con los principios y normas de la higiene de la actividad docente.

En el caso particular de la Enseñanza Superior en Cuba, teniendo en cuenta los cambios que han venido ocurriendo en ella a raíz de la Universalización, conlleva al incremento del número de estudiantes y a la demanda de docentes, unido a la heterogénea organización de las actividades docentes, que incluye distintas modalidades tales como presenciales, semi-presenciales, a distancia, etc., conduce a un proceso de planificación complejo, en extremo dinámico y altamente propenso a irregularidades y descoordinaciones que tienden a afectar la calidad del Proceso Docente Educativo. (Molina, 1995) Esta situación es aún más crítica en la Universidad de las Ciencias Informáticas, la cual cada año ha estado aumentando su matricula a razón de 2000 estudiantes por año, añadiéndose a todo esto el hecho de que la UCI es una universidad atípica, de corte productivo, donde el sistema de planificación docente se ve seriamente afectado.

1.2.4. Sistemas de planificación académica existentes.

En el mundo la necesidad de una planificación académica más eficiente ha impulsado el desarrollo de sistemas de planificación docentes cada vez más avanzados e “intuitivos”, en este sentido es destacable el trabajo hecho por la Universidad de Wyoming , USA, que cuenta con una aplicación Web para la Office of Institutional Analysis (Oficina de Análisis Institucional) la cual apoya la planificación y la gestión docente mediante el suministro de información de alta calidad a los directivos y secretarios de la Universidad encargados de la elaboración y aplicación de políticas y planificaciones docentes en general, contando con un avanzado sistema de reportes (Academic Planning Report Generator). Otro caso interesante es el Sistema de Gestión Académica UXXI (Universitas XXI) desarrollado en la PUCE (Pontificia Universidad Católica del Ecuador) el cual es un sistema de gestión académica que facilita la participación de profesores, estudiantes y personal administrativo en las gestiones administrativas de planificación. Las áreas funcionales y de gestión que cubre UXXI son la actividad académica del estudiante, la planificación de los recursos docentes y el acceso al campus virtual. En el caso de Cuba no se ha registrado la existencia de sistemas similares a pesar de contar con un elevado por ciento de jóvenes realizando estudios universitarios en las decenas de institutos de educación superior con que cuenta el país. Por lo que los sistemas mencionados fueron de gran utilidad para tener una idea del diseño y funcionalidades que debía tener el sistema de planificación docente que proponemos.

1.3. Metodologías para el desarrollo de Software.

En un proyecto de desarrollo de software la metodología define Quién debe hacer Qué, Cuándo y Cómo debe hacerlo, de ahí la importancia que reviste una correcta selección de la metodología a emplear. Hoy día no existe una metodología que sea enteramente universal. Las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigen que el proceso sea configurable. Algunas de las metodologías orientadas a objetos más importantes existentes en la actualidad son: Rational Unified Process (RUP), Extreme Programming (XP), OPEN, Metrica3 y Microsoft Solution Framework (MSF) entre otras.

1.3.1. Rational Unified Process

El proceso unificado de desarrollo de software fue publicado en el año 1998 como resultado de varios años de experiencia, el mismo es un proceso de desarrollo de software que unido al Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada en la actualidad para el análisis, implementación y documentación de sistemas orientados a objetos.

Esta metodología divide en 4 fases el desarrollo de software:

- **Inicio:** El Objetivo en esta etapa es determinar la visión del proyecto.
- **Elaboración:** En esta etapa el objetivo es determinar la arquitectura óptima.
- **Construcción:** En esta etapa el objetivo es llegar a obtener la capacidad operacional inicial.
- **Transición:** El objetivo es llegar a obtener el release del proyecto.

En RUP las actividades son agrupadas en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo.

Flujos de trabajo:

- **Modelamiento del negocio:** Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- **Requerimientos:** Define qué es lo que el sistema debe hacer, para lo cual se identifican
- las funcionalidades requeridas y las restricciones que se imponen.
- **Análisis y diseño:** Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.
- **Implementación:** Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.

- **Prueba (Testeo):** Busca los defectos a los largo del ciclo de vida.
- **Instalación:** Produce release del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.
- **Administración del proyecto:** Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- **Administración de configuración y cambios:** Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, etc.
- **Ambiente:** Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

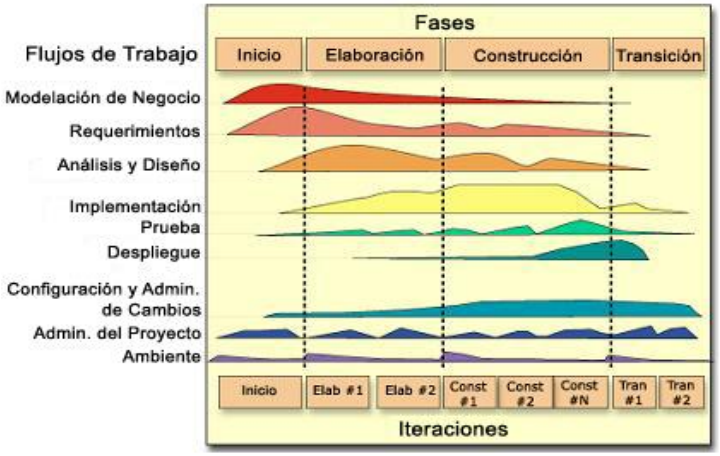


Figura 1.1 Rational Unified Process (RUP).

Entre otras facilidades RUP proporciona un enfoque disciplinado de cómo asignar tareas y responsabilidades dentro de una organización de desarrollo de software. Con el uso de RUP el objetivo primordial de la organización pasa a ser "asegurar la producción de software de alta calidad que satisfaga las necesidades de los usuarios finales" (Ivar Jacobson, 2000). Otra de las características interesantes que ofrece RUP es que tiende a aumentar considerablemente la productividad en equipo, proporcionando a cada miembro del equipo un fácil acceso a una base de conocimientos con directrices, plantillas y herramientas de ayuda para todas las actividades de desarrollo críticas. Precisamente, gracias a que todos los miembros del equipo acceden a la misma base de conocimientos, no importa si cada uno de ellos está trabajando en diferentes flujos de trabajo como: Diseño, Prueba, Administración del proyecto o configuración de Administración. Con RUP aseguramos que todos los miembros de nuestro equipo compartan un lenguaje común, proceso y visión de cómo desarrollar software. RUP esta soportado por herramientas las cuales automatizan gran parte de los procesos, mediante estas se crean y se mantienen numerosos artefactos del proceso de la ingeniería de software. Dichas herramientas son invaluables a la hora de realizar la

documentación asociada a la gestión de cambios y a la gestión de configuración que acompaña cada iteración. RUP, también es un proceso configurable. Partiendo primeramente de que ningún único proceso es adecuado para todas las líneas de desarrollo de software, el proceso unificado se ajusta tanto a equipos de desarrollo pequeños como a grandes organizaciones de desarrollo. Este se basa en un simple y claro proceso de arquitectura que proporciona concordancia alrededor de toda una familia de procesos. No obstante quizá la característica más importante de RUP sea que captura muchas de las mejores prácticas en el desarrollo de software moderno de manera que resulta adecuado para una amplia gama de proyectos y organizaciones.

1.3.2. Programación Extrema.

En los últimos años la concepción de lo que representa una metodología para el desarrollo de software a evolucionado drásticamente, de tal forma que hoy por hoy se estima que cualquier metodología de desarrollo de software de cualquier proyecto necesita ser adaptada a sus necesidades y circunstancias específicas y que ninguna metodología se remite solo a una mera colección de reglas a realizar mecánicamente y en un momento en el que esté en boga.

Extreme Programming (XP) o programación extrema es una de las metodologías ágiles que más terreno ha ganado en los últimos tiempos, la misma fue diseñada para su uso en proyectos de corto plazo que por lo general necesitan desarrollar software de manera rápida en un ambiente en el cual los requerimientos suelen ser rápidamente cambiantes.

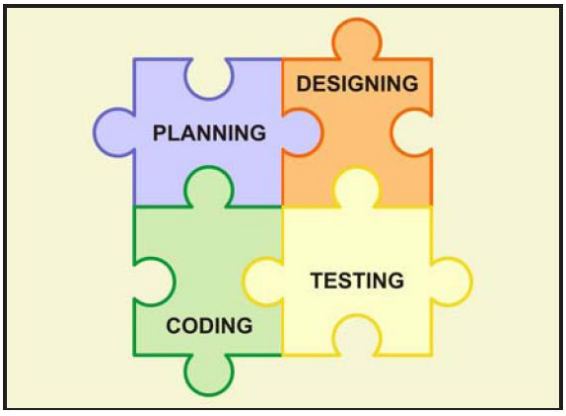


Figura 1.2 Metodología Programación Extrema.

Esta metodología que fue desarrollada por Kent Beck propone las siguientes prácticas:

- **El Proceso de Planificación:** El proceso de planificación de XP permite al cliente definir el valor de negocio de determinadas características y usa y utiliza las estimaciones de costos proporcionados por los programadores para escoger que necesidades necesitan ser realizadas y cuales no.

- **Pequeñas liberaciones (Releases):** El equipo de desarrollo pone tempranamente un sistema simple en producción, y este es actualizado frecuentemente en un ciclo bastante corto.
- **Metáfora:** El equipo de desarrollo utiliza un "sistema de nombres" común y un sistema descriptivo común que guíe el desarrollo y la comunicación.
- **Diseño Sencillo:** "Un programa construido usando XP debe ser el más sencillo programa que responda al requerimiento correspondiente", es decir lo importante en XP es proporcional valor de negocio tratando de ser lo mas concreto posible siempre intentando un máximo de eficacia.
- **Prueba:** Los equipos de desarrollo en XP se enfocan la validación del software todo el tiempo. Los programadores desarrollan software escribiendo pruebas primero y luego el software que satisface los requerimientos es reflejado en estas pruebas. También los clientes proporcionan pruebas de aceptación que les permite estar seguros de que las características que necesitan están proporcionadas.
- **Refabricación:** Se basa en mejorar el diseño del sistema a durante todo el proceso de desarrollo. Esto es llevado a cabo manteniendo el software limpio: sin la duplicación, con un nivel alto de comunicación, simple y aun así completo.
- **Programación en pares:** Se basa en que los programadores escriben todo el código en pareja, o sea dos programadores trabajando juntos en un mismo ordenador. Se ha demostrado mediante diversos experimentos que de este modo es posible lograr mejor software a un costo similar o menor que si estuviese trabajando un programador solo en un ordenador.
- **Propiedad Colectiva:** Todo el código pertenece a todos lo programadores. Esta filosofía permite al equipo avanzar a toda velocidad, sencillamente porque cuando algo necesita ser cambiado esto se realiza sin la menor demora por cualquier desarrollador disponible.
- **Integración continua:** El equipo de desarrollo integra y construye el software del sistema en múltiples ocasiones por día permitiendo esto rápidos progresos, debido a que se eliminan los problemas de integración que comúnmente plagan los equipos de desarrollo que integran con menos frecuencia el trabajo realizado.
- **40 horas por semana:** Un programador cansado comete a menudo muchos errores. Mediante el uso de XP los equipos de desarrollo no trabajan de manera excesiva horas extras, manteniéndose frescos, saludables y efectivos.
- **Cliente On-site:** Un proyecto que use la metodología XP es dirigido por un individuo dedicado el cual es el encargado de determinar requerimientos, establecer prioridades y responder todas las inquietudes de los programadores. Esto asegura que la comunicación fluya y evita que la documentación y manejo de la información sea trabajoso lo cual frecuentemente es una de las partes más costosas de un proyecto de software.

- **Codificación estándar:** Para que un equipo de desarrollo trabaje efectivamente en pareja y comparta la propiedad de todo el código, todos los programadores necesitan escribir el código de la misma forma, con reglas que aseguren que el código se comunique correctamente.

1.3.3. Microsoft Solution Framework.

El Microsoft Solutions Framework es una metodología para el desarrollo de software elástica, manejada por escenarios o contextos que incorpora las prácticas probadas desarrolladas en Microsoft con respecto a los requerimientos, diseño, seguridades, rendimiento y pruebas. La misma proporciona un sistema de modelos, principios, y pautas para dar soluciones a empresas que diseñan y desarrollan de una manera que se asegure de que todos los elementos de un proyecto, tales como gente, procesos, y herramientas, puedan ser manejados con éxito.

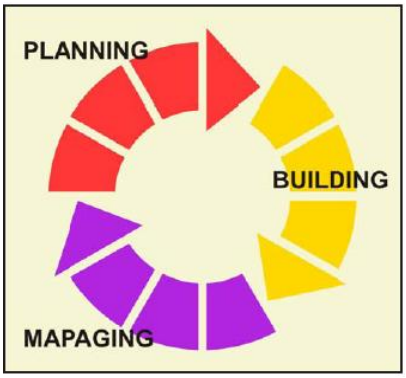


Figura 1.3 Microsoft Solution Framework.

Entre las características más importantes que tiene el MSF están: adaptable, porque es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar; escalable, puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas a más; flexible, es utilizada en el ambiente de desarrollo de cualquier cliente; tecnología agnóstica, porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

1.3.4. Fundamentación de la selección de la metodología de desarrollo de software a utilizar.

Para cualquier proyecto de desarrollo de software es fundamental definir la metodología a utilizar, ya que ésta asegura en gran medida la calidad y el mantenimiento del producto final. Estas metodologías no deben ser aplicadas como una colección de pasos firmemente establecidos sino que deben ser adaptables al contexto y necesidades de cada organización. Basándose en el estudio

realizado anteriormente y de acuerdo a la situación actual de la Facultad 3 se puede arribar a la conclusión de que lo que necesita el Sistema de ayuda para la planificación es una metodología fuerte, que responda a los intereses de la facultad y del proyecto y sobre la cual los estudiantes tengan un mayor conocimiento y práctica. En este caso se decidió utilizar el Rational Unified Process (RUP, por sus siglas en inglés) la cual sin lugar a dudas constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. También influyó en esta decisión las facilidades que ofrece RUP en cuanto a organización, documentación que genera y sus características flexibles sobradamente probadas en la práctica, así como el dominio que presentan en esta metodología la gran mayoría de los estudiantes de la Facultad 3.

1.4. Lenguajes de modelado.

Un lenguaje de modelado de objetos es un conjunto estandarizado de símbolos y de modos de disponerlos para modelar (parte de) un diseño de software orientado a objetos. El más conocido internacionalmente es UML (Unified Modeling Language) o Lenguaje Unificado de Modelado que es el que se utilizará en el trabajo, el mismo permite visualizar, especificar, construir y documentar un sistema de software. UML básicamente ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. Es importante destacar que UML es aplicable en una gran cantidad de formas para soportar una metodología de desarrollo de software (Tal como el Proceso Unificado de Rational) pero a su vez no específica en sí mismo que metodología o proceso usar. En el trabajo se utilizará la versión UML 2.0 la cual constituye una de las versiones más recientes publicada. La misma cuenta con 13 tipos diferentes de diagramas de los cuales la gran mayoría son utilizados en el trabajo y se mencionan a continuación:

Los Diagramas de Estructura enfatizan en los elementos que deben existir en el sistema modelado:

- Diagrama de clases
- Diagrama de componentes
- Diagrama de objetos
- Diagrama de estructura compuesta (UML 2.0)
- Diagrama de despliegue
- Diagrama de paquetes

Los Diagramas de Comportamiento enfatizan en lo que debe suceder en el sistema modelado:

- Diagrama de actividades

- Diagrama de casos de uso
- Diagrama de estados

Los Diagramas de Interacción son un subtipo de diagramas de comportamiento, que enfatiza sobre el flujo de control y de datos entre los elementos del sistema modelado:

- Diagrama de secuencia
- Diagrama de colaboración
- Diagrama de tiempos (UML 2.0)
- Diagrama de vista de interacción (UML 2.0)

1.5. Selección de herramientas a utilizar en el sistema.

Las herramientas soportan el proceso de desarrollo de cualquier software en la actualidad. Se afirma que "Hoy día, es impensable desarrollar software sin utilizar un proceso soportado por herramientas." (Ivar Jacobson, 2000). Estas son muy útiles para automatizar procesos repetitivos, mantener las cosas estructuradas, gestionar grandes cantidades de información, incrementando en gran medida la productividad y la calidad del trabajo y reduciendo el tiempo de.

1.5.1. Herramientas CASE.

Dentro de las herramientas más utilizadas para el desarrollo de software hoy día están las herramientas CASE (Computer Aided Software Engineering) o Ingeniería de Software Asistida por Computadora, éstas representan un conjunto de aplicaciones informáticas que tienen como objetivo automatizar los aspectos claves de todo el proceso de desarrollo de un sistema, compensando de esta manera su coste inicial en forma de ahorro de tiempo y recursos para el proyecto en cuestión. En la actualidad existe gran diversidad de estas herramientas destacándose en cuanto a diseño y construcción de sistemas Rational Rose la cual fue desarrollada por los creadores de UML (Booch, Rumbaugh y Jacobson), que cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases y entregables. Esta herramienta propone la utilización de cuatro tipos de modelo para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software. Rational Rose también utiliza un proceso de desarrollo iterativo controlado (controlled iterative process development), donde el desarrollo se lleva a cabo en una secuencia de iteraciones. Cada iteración comienza con una primera aproximación del análisis, diseño e implementación para identificar los riesgos del diseño, los cuales

se utilizan para conducir la iteración, primero se identifican los riesgos y después se prueba la aplicación para que éstos se hagan mínimos.

Cuando la implementación pasa todas las pruebas que se determinan en el proceso, ésta se revisa y se añaden los elementos modificados al modelo de análisis y diseño. Una vez que la actualización del modelo se ha modificado, se realiza la siguiente iteración. Otra de las características mas interesantes de Rose es que permite que haya varias personas trabajando a la vez en el proceso iterativo controlado, para ello posibilita que cada desarrollador opere en un espacio de trabajo privado que contiene el modelo completo y tenga un control exclusivo sobre la propagación de los cambios en ese espacio de trabajo. También es posible descomponer el modelo en unidades controladas e integrarlas con un sistema para realizar el control de proyectos que permite mantener la integridad de dichas unidades. Otra característica importante es que usando Rose se puede generar código en distintos lenguajes de programación a partir de un diseño en UML. Es destacable también señalar que Rational Rose proporciona mecanismos para realizar la denominada Ingeniería Inversa, es decir, a partir del código de un programa, se puede obtener información sobre su diseño.

Otra herramienta case sumamente interesante y a tomar en consideración es Visual Paradigm la cual da soporte para el modelado visual tanto con UML 2.0 como 2.1. La misma soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Visual Paradigm también ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Las siguientes características demuestran las verdaderas potencialidades de esta herramienta: entorno de creación de diagramas para UML 2.0; diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad; uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación; capacidades de ingeniería directa e inversa; modelo y código que permanece sincronizado en todo el ciclo de desarrollo; disponibilidad de múltiples versiones, para cada necesidad; disponibilidad de integrarse en los principales IDEs (Integrated Development Environment); disponibilidad en múltiples plataformas; ofrece un mecanismo general para la organización de los modelos/subsistemas/capas agrupando elementos de modelado y versión gratuita (licencia para Community Edition). (VISUAL-PARADIGM 2007)

No obstante a todas las facilidades que ofrece Visual Paradigm se escogió Rational Rose como herramienta CASE para realizar el modelado visual de los procesos en el sistema debido a la mayor experiencia que se tenia en el uso de esta herramienta y a que se requieren menos requisitos de hardware.

1.6. Lenguajes de Programación y Tecnologías del lado del Cliente.

Los lenguajes de programación del lado del cliente son aquellos que solamente pueden ser interpretados por una aplicación cliente como el navegador Web entre estos lenguajes están DHTML, Javascript, etc.

El HTML Dinámico o DHTML (del inglés Dynamic HTML) es el arte de construir sitios Web dinámicos combinando HTML estático con un lenguaje interpretado con el lado del cliente (Ej., Javascript), las CSS (Cascading Style Sheets) y la jerarquía de objetos de un DOM (Document Object Model). Otra manera de entender qué es DHTML es mediante lo afirmado por la W3C (World Wide Web Consortium) "El HTML dinámico es un término usado por algunos vendedores para describir la combinación del HTML, de las hojas del estilo y las scripts que permiten que las documentos Web sean animados" (Introduction to DHTML). Con el DHTML se pueden desarrollar entre otras cosas menús desplegables, imágenes que cambian al pasar el cursor sobre ellas, objetos en movimiento, botones que permiten desplazar el texto que se está mostrando, textos explicativos que aparecen al situar el cursor sobre ciertas palabras clave, cronómetros, etc. DHTML ofrece a diferencia de otras herramientas la ventaja de que no requiere ningún tipo de plug-in para poder utilizarlo.

Javascript

Básicamente Javascript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con Javascript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. De Javascript se puede decir que es un lenguaje orientado a objetos, ya que dispone del mecanismo de herencia, si bien esta se realiza siguiendo el paradigma de programación basada en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad. Javascript también utilizado para crear pequeños algoritmos, funciones, etc., encargados de realizar acciones dentro del ámbito de una página Web.

En la arena internacional Javascript fue utilizado de forma masiva por la mayoría de sitios de Internet hasta la aparición de Flash ya que este permitía realizar algunas acciones imposibles de llevar a cabo mediante el uso de Javascript. a su vez sin embargo, la aparición de las aplicaciones AJAX programadas con JavaScript le ha devuelto una popularidad sin igual dentro de los lenguajes de programación Web.

CSS

Las Hojas de Estilo en Cascada (Cascading Style Sheets) o CSS, es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo

va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos. En la Web las CSS son utilizadas para dar estilo a documentos y XML, separando el contenido de la presentación. Los Estilos definen la forma de mostrar los elementos HTML y XML. CSS permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas Web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento. (Guía Breve de CSS). Las CSS sin duda constituyen una de las tecnologías que han permitido revolucionar el concepto de Webs personalizadas.

AJAX

AJAX o (Asinchronous Javascript And XML) no es más que una técnica de desarrollo Web, mediante la cual se puede crear aplicaciones Web más rápidas y cómodas para el usuario. Por medio de esta técnica el cliente puede interactuar con el servidor de manera asincrónica, actualizando las páginas, sin necesidad de volver a cargarlas. Esto se traduce en un aumento significativo de la interactividad, velocidad y usabilidad en las mismas. De AJAX se ha dicho: “Ajax no es una tecnología en sí mismo. En realidad, se trata de varias tecnologías independientes que se unen de formas nuevas y sorprendentes.” (Garrett, 2005)

Entre las tecnologías que conforman AJAX están:

- XHTML y CSS, para crear una presentación basada en estándares.
- DOM, para la interacción y manipulación dinámica de la presentación.
- XML, XSLT y JSON, para el intercambio y la manipulación de información.
- XMLHttpRequest, para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías.

1.7. Lenguajes de Programación y Tecnologías del lado del Servidor.

Cuando se habla de lenguajes del lado servidor se refiere a aquellos lenguajes que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él. Entre los lenguajes del lado del servidor más utilizados a escala internacional para el desarrollo de páginas Web están PHP, ASP, PERL y JAVA. Específicamente para el desarrollo del sistema que se propone se escogió PHP debido a una serie de características que como se verá mas adelante lo hacen el ideal en lo que a la aplicación se refiere.

PHP (HyperText Pre-Processor) es un lenguaje de programación interpretado del lado del servidor de alto nivel, embebido en páginas HTML y es utilizado para el desarrollo de sitios Web dinámicos e interactivos.

A continuación se muestran las características de PHP más interesantes y las que motivaron la decisión de usarlo para el desarrollo del sistema:

- Un potente soporte para gran cantidad de bases de datos entre las que se pueden mencionar InterBase, SQL, MySQL, Oracle, Informix, PostgreSQL, entre otras.
- Integración con varias bibliotecas externas, que dan al desarrollador la posibilidad de realizar cualquier tarea, desde generar documentos en pdf hasta analizar código XML.
- Es software libre, lo que implica menos costes y servidores más baratos que otras alternativas.
- Muy rápido y su integración con la base de datos MySQL y el servidor Apache, le permite constituirse como una de las alternativas más atractivas del mercado.
- Multiplataforma, funciona tanto para Unix (con Apache) como para Windows (con Microsoft Internet Information Server) de forma que el código que se haya creado para una de ellas no tiene porqué modificarse al pasar a la otra.
- Sintaxis inspirada en C, ligeramente modificada para adaptarlo al entorno en el que trabaja.
- Tiene una de las comunidades más grandes en Internet, con lo que no es complicado encontrar ayuda, documentación, artículos, noticias, y más recursos.

1.7.1. Sistemas Gestores de Bases de Datos.

Con el transcurso de los años el volumen y la diversidad de información manejada en el mundo entero han ido aumentando a pasos agigantados, constituyendo una necesidad de primer orden la creación de herramientas que faciliten el trabajo con esta información así como su almacenamiento, con este objetivo es que surgen las Bases de Datos. Paradójicamente estas Bases de Datos a pesar de los indudables beneficios que reportan no son capaces por si solas de actualizarse, o auto mantenerse sin que requieran de un trabajo manual muy engorroso e impensable en una sociedad tan dinámica e informatizada como la que vivimos. Esta situación a su vez ha dado lugar al surgimiento de unos programas denominados Sistemas Gestores de Bases de Datos (SGBD) los cuales constituyen una suerte de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Los SGBD permiten de manera sencilla almacenar la información y posteriormente acceder a la misma de forma rápida y estructurada lo cual facilita un eficiente manejo de datos para cualquier organización.

De manera general se pueden definir los objetivos más importantes de los SGBD como los siguientes:

- Evitar la redundancia de los datos, eliminando así la inconsistencia de los mismos.
- Mejorar los mecanismos de seguridad de los datos y la privacidad. Podemos distinguir cuatro tipos de contextos para usar mecanismos de seguridad: seguridad contra accesos indebidos a los datos, seguridad contra accesos no autorizados a la base de datos, seguridad contra destrucción causada por el entorno (fuego, inundación, robo, etc.), seguridad contra fallos del propio sistema (fallos del hardware, del software, etc.).
- Asegurar la independencia de los programas y los datos, es decir, la posibilidad de modificar la estructura de la base de datos (esquema) sin necesidad de modificar los programas de las aplicaciones que manejan esos datos.
- Mantener la integridad de los datos realizando las validaciones necesarias cuando se realicen modificaciones en la base de datos.
- Mejorar la eficacia de acceso a los datos, en especial en el caso de consultas imprevistas.

Entre los sistemas Gestores de Bases de Datos más usados en la actualidad se encuentran MySQL, Oracle, SQL Server, PostgreSQL, Interbase, Etc.

A continuación se analizarán 4 de los SGBD más utilizados tanto en el mundo como en nuestro país y universidad:

- **MySQL:** se puede decir que es un sistema de gestión de base de datos relacional, multihilo y multiusuario. Fue creada por la empresa sueca MySQL AB, que mantiene el copyright del código fuente del servidor SQL, así como también de la marca. Gracias a la colaboración de muchos usuarios, la base de datos se ha ido mejorando optimizándose en velocidad razón por la cual una de las bases de datos más usadas en la Internet. Por su sencillez y sus características es usado por muchas personas ya que consume muy pocos recursos, es usado tanto en aplicaciones sencillas como complejas. Es utilizado también en aplicaciones Web como Drupal, en plataformas (Linux/Windows-Apache-MySQL-PHP/Perl/Python), y por herramientas de seguimiento de errores como Bugzilla, además sus conexiones generalmente son muy seguras.
- **Oracle (Relational Database Management System):** básicamente una herramienta cliente/servidor para la gestión de base de datos, es un producto vendido a nivel mundial, aunque la gran potencia que tiene y su elevado precio hace que solo se vea en empresas muy grandes y multinacionales, por norma general. El mismo es considerado uno de los sistemas de bases de datos más completos destacándose por su:
    - Soporte de transacciones.
    - Estabilidad.
    - Escalabilidad.
    - Es multiplataforma.

Entre los aspectos que le han sido fuertemente criticados a Oracle por los especialistas están la seguridad de la plataforma y las políticas de suministro de parches de seguridad, modificadas a comienzos de 2005 y que incrementan el nivel de exposición de los usuarios. En los parches de actualización provistos durante el primer semestre de 2005 fueron corregidas 22 vulnerabilidades públicamente conocidas, algunas de ellas con una antigüedad de más de 2 años.

- **Microsoft SQL Server:** sistema de gestión de bases de datos relacionales (SGBD) basado en el lenguaje Transact-SQL, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea. Microsoft SQL Server se ha convertido en la actualidad en una suerte de plataforma que permite el análisis y la administración de datos empresariales con herramientas de Inteligencia de Negocios, análisis, reporteo, integración y notificación integradas. Provee de mayor escalabilidad, disponibilidad y seguridad al tiempo que simplifica la creación, implementación y gestión de aplicaciones altamente disponibles. Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente. También Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo acceden a la información.
- **PostgreSQL:** sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS) libre y bajo la licencia BSD. PostgreSQL en la actualidad está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo, a esta realidad contribuye mucho el hecho de que no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales la cual trabaja en su desarrollo y perfeccionamiento, esta comunidad es denominada (PostgreSQL Global Development Group). PostgreSQL también sirve de soporte a muchos de los lenguajes más utilizados hoy día tal es el caso de PHP, C, C++, Java, Python, Ruby, etc, constituyendo una alternativa seria a otros sistemas de bases de datos de código abierto (como MySQL, Firebird y MaxDB), así como sistemas propietarios como Oracle y SQLServer.

Entre las innumerables ventajas que ofrece PostgreSQL están las siguientes:

- Posee una gran escalabilidad. Es capaz de ajustarse al número de procesadores y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta (en algunos benchmarks se dice que ha llegado a soportar el triple de carga de lo que soporta MySQL).
- Implementa el uso de rollbacks, subconsultas y transacciones, haciendo su funcionamiento mucho más eficaz.
- Tiene la capacidad de comprobar la integridad referencial, así como también la de almacenar procedimientos en la propia base de datos.

- Es Multiusuario, multiprogramado, con arquitectura cliente-servidor y control de privilegios de acceso.

Para el desarrollo del sistema de planificación académica se recomienda fuertemente el uso del último de los sistemas gestores de bases de datos que se han estado tratando en este epígrafe, es decir, PostgreSQL debido a que gracias al estudio realizado del tema así como la recomendación realizada por otros profesionales y de personal de experiencia en la Facultad 3 de la UCI se pudo constatar que la herramienta constituye la principal alternativa dentro del software libre lo cual es una de las principales premisas que se trata de seguir en la facultad e universidad en general, en adición a esto podríamos sumar entre otra bondades que esta base de datos no necesita ningún tipo de configuración, ni mantenimiento para los desarrolladores lo cual ahorraría significativamente tiempo de trabajo a los desarrolladores.

1.7.2. 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 de Internet) 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 al cliente o navegador que realizó la petición.

En nuestros días, a pesar de que existen cierto número de servidores Web, el mercado de estos softwares esta prácticamente dominado por Apache. Se estima según estudios realizados en la red mundial que alcanzó su máxima cota de mercado en 2005 siendo el servidor empleado en el 70% de los sitios Web en el mundo, sin embargo ha sufrido un descenso en su cuota de mercado en los últimos años. (www.netcraft.com)

Entre las características más destacables de apache están las siguientes:

- Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.
- Apache es una tecnología gratuita de código fuente abierto. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto. Esto le da una transparencia a este software de manera que si queremos ver que es lo que estamos instalando como servidor, puede ser estudiado, sin ningún secreto o puerta trasera nos sorprenda.
- Apache es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor Web Apache. Actualmente existen muchos módulos para Apache que son adaptables a este, y están ahí para que los instalemos cuando los necesitemos. Otra

cosa importante es que cualquiera que posea cierta experiencia en la programación de C o Perl puede escribir un modulo para realizar una función determinada.

- Apache trabaja con gran cantidad de Perl, PHP y otros lenguajes de script. Perl destaca en el mundo del script y Apache utiliza su parte del pastel de Perl tanto con soporte CGI como con soporte mod perl. También trabaja con Java y páginas jsp. Teniendo todo el soporte que se necesita para tener páginas dinámicas.
- Apache te permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.
- Tiene una alta configurabilidad en la creación y gestión de logs. Apache permite la creación de ficheros de log a medida del administrador, de este modo puedes tener un mayor control sobre lo que sucede en tu servidor. (Ricardo J. Vargas Del Valle, Programación en Capas.)

Debido a las características anteriormente mencionadas y a que Apache servidor Web potente y flexible que puede funcionar una amplia variedad de plataformas y entornos fue que determinamos que este es el servidor Web que se debe utilizar para el desarrollo del sistema. En esta decisión también influyó el amplio dominio que presentan los estudiantes de la universidad en la herramienta.

1.8. Análisis y diseño de sistemas.

1.8.1. Análisis.

Durante el análisis, se consideran los requisitos que se describieron en la captura de los mismos, refinándolos y estructurándolos con el objetivo de conseguir una comprensión más precisa de los mismos y lograr estructurar el sistema de forma global incluyendo su arquitectura.

El lenguaje que se utiliza en el análisis se basa en un modelo de objetos conceptual denominado modelo de análisis el cual nos ayuda a refinar los requisitos así como razonar sobre los aspectos internos del sistema.

Las tareas que constituyen el análisis se pueden agrupar en una serie de etapas que se suceden de forma iterativa hasta validar el proceso completo como en el siguiente caso:

- Conceptualización: Consiste en obtener una visión de muy alto nivel del sistema, identificando sus elementos básicos y las relaciones de éstos entre sí y con el entorno.
- Análisis funcional: Describe las acciones o transformaciones que tienen lugar en el sistema. Dichas acciones o transformaciones se especifican en forma de procesos que reciben una entradas y producen unas salidas.

- Análisis de condiciones (o constricciones): Debe reflejar todas aquellas limitaciones impuestas al sistema que restringen el margen de las soluciones posibles.
- Construcción de modelos: Una de las formas más habituales y convenientes de analizar un sistema consiste en construir un prototipo (un modelo en definitiva) del mismo.
- Validación del análisis: A fin de comprobar que el análisis efectuado es correcto y evitar en su caso la posible propagación de errores a la fase de diseño, es imprescindible proceder a la validación del mismo.

Principios del análisis:

En la pasada década, se desarrollaron varios métodos de análisis y especificación del software. Los investigadores han identificado los problemas y sus causas y desarrollando reglas y procedimientos para resolverlos. Cada método de análisis tiene una única notación y punto de vista. Sin embargo, todos los métodos de análisis están relacionados por un conjunto de principios fundamentales:

- El dominio de la información, así como el dominio funcional de un problema debe ser representado y comprendido.
- El problema debe subdividirse de forma que se descubran los detalles de una manera progresiva (o jerárquica).
- Deben desarrollarse las representaciones lógicas y físicas del sistema.

Aplicando estos principios, el analista enfoca el problema sistemáticamente. Se examina el dominio de la información de forma que pueda comprenderse su función de manera más completa. La partición se aplica para reducir la complejidad. La visión lógica y física del software, es necesaria para acomodar las ligaduras lógicas impuestas por los requerimientos de procesamiento, y las ligaduras físicas impuestas por otros elementos del sistema.

¿Que es lo que se obtiene en este flujo de trabajo?

En el análisis se obtienen las descripciones de los objetos de datos, los diagramas entidad – relación, los diagramas de flujo de datos, los diagramas de transición de estados, las especificaciones del proceso y las especificaciones de control son creadas como resultados de las actividades del análisis.

1.9. Diseño.

El Diseño de Sistemas se ocupa de desarrollar las directrices propuestas durante el análisis en función de aquella configuración que tenga más posibilidades de satisfacer los objetivos planteados tanto desde el punto de vista funcional como del no funcional (lo que antes hemos denominado

constricciones). El proceso de diseño de un sistema complejo se suele realizar de forma descendente:

- Diseño de alto nivel (o descomposición del sistema a diseñar en subsistemas menos complejos).
- Diseño e implementación de cada uno de los subsistemas:
    - Especificación consistente y completa del subsistema de acuerdo con los objetivos establecidos en el análisis.
    - Desarrollo según la especificación.
    - Prueba.
- Integración de todos los subsistemas.
- Validación del diseño.

1.9.1. Principios del diseño.

Los principios de diseño son aplicables a todos los niveles del diseño del software, desde la arquitectura a la microarquitectura, además todos estos principios se relacionan entre si. Entre los principios universales más utilizados están los siguientes: abstracción y refinamiento, modularidad, variaciones protegidas, acoplamiento, cohesión, refactorización y reutilización.

1.10. Patrones de diseño

En términos generales, un patrón es un conjunto de información que proporciona respuesta a un conjunto de problemas similares, es decir, un patrón es una solución a un problema en un contexto, donde:

- Contexto son las situaciones recurrentes a las que es posible aplicar el patrón.
- Problema es el conjunto de metas y restricciones que se dan en ese contexto.
- Solución es el diseño a aplicar para conseguir las metas dentro de las restricciones.

1.10.1. ¿Porqué usar patrones?

Las principales razones que inducen al uso de los patrones son las siguientes:

- Producción de Software más flexible al cambio.
- Establece problemas Pareja-Solución.
- Ayudan a especificar interfaces.
- Reutilización del Código.
- Uso de Documentación Estándar.

Categorías de patrones

Según la escala o nivel de abstracción:

- Patrones de arquitectura: Aquéllos que expresan un esquema organizativo estructural fundamental para sistemas software.
- Patrones de diseño: Aquéllos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas software.
- Idiomas: Patrones de bajo nivel específicos para un lenguaje de programación o entorno concreto.

1.11. Patrones a utilizar en el trabajo.

1.11.1. Modelo-Vista-Controlador (MVC).

En el trabajo se utilizó el patrón de diseño MVC (Model View Controller) o Modelo Vista Controlador el cual constituye uno de los patrones más usados para aplicaciones Web. Las razones por la que usamos este patrón pueden quedar definidas de la manera siguiente:

“La lógica de una interfaz de usuario cambia con más frecuencia que el almacenado de datos y la lógica de negocio. Si realizamos un diseño ofuscado, es decir, un pastiche que mezcle los componentes de interfaz y de negocio, entonces la consecuencia será que, cuando necesitemos cambiar el interfaz, tendremos que modificar trabajosamente los componentes de negocio. Mayor trabajo y más riesgo de error”. (Lago, Patrones de diseño software, 2007)

Se trata de realizar un diseño que desacople la vista del modelo, con la finalidad de mejorar la reusabilidad. De esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos. (Lago, Patrones de diseño software, 2007)

Los elementos principales del patrón son:

- Modelo: datos y reglas de negocio
- Vista: muestra la información del modelo al usuario
- Controlador: gestiona las entradas del usuario

El modelo es el responsable de:

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor".

- Lleva un registro de las vistas y controladores del sistema.
- Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un fichero bath que actualiza los datos, un temporizador que desencadena una inserción, etc.).

El controlador es responsable de:

- Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
- Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada al método "Actualizar()". Una petición al modelo puede ser "Obtener_tiempo_de_entrega (nueva_orden_de_venta)".

Las vistas son responsables de:

- Recibir datos del modelo y mostrarlos al usuario.
- Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
- Pueden dar el servicio de "Actualización()", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes). (Lago, Patrones de diseño software, 2007)

Todas estas características permitieron que el trabajo ganara en organización, se lograra una mayor reutilización de código ahorrando tiempo y lográndose una mayor escalabilidad.

1.11.2. Patrones GRASP.

Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades y de cierto modo ayudan a determinar las clases que estarán en el diseño.

A continuación se ofrece una relación de los principales patrones GRASP y el objetivo especifico que cumple cada uno de ellos:

Experto: Cada objeto es responsable por mantener su propia información (principio de encapsulamiento) o sea que este conoce y puede informar el valor de sus atributos así como modificarlos. Si tiene relación de agregación (fuerte) con otros objetos (sus partes), también será responsable de conocer la información de ellos, de crearlos (patrón creador) y de delegarles las operaciones.

Creador: El objeto B tiene la responsabilidad de crear objetos de la clase A si:

- B agrega objetos A.

- B contiene objetos A.
- B registra objetos A.
- B usa (exhaustivamente) objetos A.
- B posee la información necesaria para inicializar A.

Bajo Acoplamiento: El acoplamiento es la medida de cuánto una clase esta conectada (tiene conocimiento) de otras clases por lo que éste patrón es evaluativo. Un bajo acoplamiento permite que el diseño de clases sea más independiente reduciendo el impacto de los cambios y aumentando la reutilización.

Alta Cohesión: La cohesión funcional dentro de una clase es una medida que indica cuán relacionadas están las responsabilidades de una clase por lo que también éste es un patrón evaluativo. Entre más alta cohesión resulta más fácil de entender, cambiar y reutilizar.

Controlador: Consiste en Asignar la responsabilidad de manejar los mensajes eventos del sistema a una clase facilitando centralizar las actividades (tales como validación, seguridad entre otras).

1.11.3. Patrones GoF.

Patrones de creación

- Abstract Factory: proporciona una interfaz para crear familias de objetos o que dependen entre sí, sin especificar sus clases concretas.
- Builder: separa la construcción de un objeto complejo de su representación, de forma que el mismo proceso de construcción pueda crear diferentes representaciones.
- Factory Method: define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar. Permite que una clase delegue en sus subclases la creación de objetos.
- Prototype: especifica los tipos de objetos a crear por medio de una instancia prototípica, y crear nuevos objetos copiando este prototipo.
- Singleton: garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella.

Patrones estructurales

- Adapter: convierte la interfaz de una clase en otra distinta que es la que esperan los clientes. Permiten que cooperen clases que de otra manera no podrían por tener interfaces incompatibles.
- Bridge: desvincula una abstracción de su implementación, de manera que ambas puedan variar de forma independiente.

- Composite: combina objetos en estructuras de árbol para representar jerarquías de parte-todo. Permite que los clientes traten de manera uniforme a los objetos individuales y a los compuestos.
- Decorator: añade dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender la funcionalidad.
- Facade: proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema se más fácil de usar.
- Flyweight: usa el compartimiento para permitir un gran número de objetos de grano fino de forma eficiente.
- Proxy: proporciona un sustituto o representante de otro objeto para controlar el acceso a éste.

Patrones de comportamiento

- Chain of Responsibility: evita acoplar el emisor de una petición a su receptor, al dar a más de un objeto la posibilidad de responder a la petición. Crea una cadena con los objetos receptores y pasa la petición a través de la cadena hasta que esta sea tratada por algún objeto.
- Command: encapsula una petición en un objeto, permitiendo así parametrizar a los clientes con distintas peticiones, encolar o llevar un registro de las peticiones y poder deshacer la operaciones.
- Interpreter: dado un lenguaje, define una representación de su gramática junto con un intérprete que usa dicha representación para interpretar las sentencias del lenguaje.
- Iterator: proporciona un modo de acceder secuencialmente a los elementos de un objeto agregado sin exponer su representación interna.
- Mediator: define un objeto que encapsula cómo interactúan un conjunto de objetos. Promueve un bajo acoplamiento al evitar que los objetos se refieran unos a otros explícitamente, y permite variar la interacción entre ellos de forma independiente.
- Memento: representa y externaliza el estado interno de un objeto sin violar la encapsulación, de forma que éste puede volver a dicho estado más tarde.
- Observer: define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos.
- State: permite que un objeto modifique su comportamiento cada vez que cambia su estado interno. Parecerá que cambia la clase del objeto.
- Strategy: define una familia de algoritmos, encapsula uno de ellos y los hace intercambiables. Permite que un algoritmo varíe independientemente de los clientes que lo usan.

- Template Method: define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos. Permite que las subclases redefinan ciertos pasos del algoritmo sin cambiar su estructura.
- Visitor: representa una operación sobre los elementos de una estructura de objetos. Permite definir una nueva operación sin cambiar las clases de los elementos sobre los que opera. (Gracia, 2005).

1.11.4. Patrones de casos de uso.

Patrón CRUD (Creating Reading Updating and Deleting): Consiste en un caso de uso, llamado "Información CRUD" o "Administrar Información", que modela todas las operaciones que se pueden realizar sobre una parte de información de cierto tipo, tal como crearla, leerla, actualizar y eliminar. Este patrón debe ser usado como todos los flujos contribuyen al mismo valor de negocio.

- Actores Múltiples: Rol Común (Multiple Actors: Common Role): dos actores juegan el mismo papel hacia el caso de uso. Este rol es representado por otro actor, heredado por los actores que comparten este rol. Este patrón es aplicable cuando, desde del punto de vista de un caso de uso hay solo una entidad externa interactuando con cada instancia del caso de uso.
- Concrete Inclusión or Extensión: el objetivo principal de este patrón es modelar flujos como parte de un caso de uso base y otra parte en otro caso de uso que complete al caso de uso base, es un patrón estructural.

    1. Inclusión: en este patrón hay una relación de inclusión del caso de uso base al caso de uso incluido, el cual puede ser instanciado por si mismo, este patrón puede ser usado cuando un flujo puede ser incluido en flujo de otro caso de uso.
    2. Exclusión: consiste en la extensión de un caso de uso a otro caso de uso base. El caso de uso extendido es concreto, o sea, puede estar instanciado por si mismo tanto como el caso de uso base. Este patrón puede ser usado cuando un flujo pude extender el flujo de otro caso de uso.

1.12. Requerimientos.

Los requerimientos son uno de los aspectos más importantes en el proceso de desarrollo de software, debido a que reflejan las necesidades que debe cumplir el sistema. Las capacidades que los clientes exigen a las aplicaciones, cubren necesidades referentes básicamente a funcionalidades que debe hacer el sistema, qué información se va a manejar u otras características no funcionalidades como la facilidad de uso y el rendimiento. Según la IEEE Standard Glossary of Software Engineering Terminology, define un requerimiento como:

1. Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
2. Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.
3. Una representación documentada de una condición o capacidad como en 1 o 2.

1.12.1. Ingeniería de requisitos.

La ingeniería de requisitos del software es un proceso de descubrimiento, refinamiento, modelado y especificación. Se refina en detalle los requisitos del sistema y el papel asignado al software. En la ingeniería de requisitos tanto el desarrollador como el cliente tienen un papel activo. El cliente intenta replantear un sistema confuso, mientras que el desarrollador como un interrogador, consultor, como una persona que resuelve problemas y como un negociador.

La ingeniería de requisitos facilita el mecanismo apropiado para comprender lo que quiere el cliente, analizando necesidades, confirmando su viabilidad, negociando una solución razonable, especificando la solución sin ambigüedad, validando la especificación y gestionando los requisitos para que se transformen en un sistema operacional. (Pressman, 2005).

1.12.2. Actividades de la Ingeniería de requisitos.

Identificación de requisitos: de las seis actividades tratadas es la de mayor interacción con los clientes, en esta actividad se recurre a la realización de entrevistar, talleres entre otras técnicas, además de ser en la que el analista y los expertos en el tema trabajan conjuntamente con el cliente para obtener los requerimientos reales de una forma óptima. En esta etapa es donde se adquiere toda la información referente al trabajo del cliente y se busca comprender sus necesidades y se detallan las restricciones del negocio objeto de estudio y como resultado se obtiene los requerimientos de todas las parte involucradas en el negocio.

Análisis y Negociación de requisitos: se realiza un estudio de los requerimientos obtenidos previamente para detectar la presencia de áreas no especificadas, requisitos contradictorios y peticiones vagas e irrelevantes. Como resultado a este análisis puede que se necesite volver a la primera actividad para eliminar todo tipo de inconsistencia.

Especificación de requisitos: la especificación del sistema es el producto final sobre los requisitos del sistema obtenido por el analista. Esta especificación describe la información que entra y sale del sistema y delimita cada elemento del sistema. (Pressman, 2005).

Modelado del sistema: en esta actividad se modela el sistema con el objetivo de evaluar los compontes del sistema y sus relaciones entre sí, además de reflejar en dicho sistema todos los requerimientos capturados.

Validación de requisitos: la validación de requisitos examina las especificaciones para asegurar que todos los requisitos del sistema han sido establecidos sin ambigüedad, sin inconsistencias, sin omisiones, que los errores detectados hayan sido corregidos, y que el resultado del trabajo se ajusta a los estándares establecidos para el proceso, el proyecto y el producto. (Pressman, 2005).

Gestión de requisitos: la gestión de requisitos es un conjunto de actividades que ayudan al equipo de trabajo a identificar, controlar y seguir los requisitos y los cambios en cualquier momento (Pressman, 2005).

1.13. Programación por Capas.

La Programación por capas es un estilo de programación en el que el principal objetivo es organizar la forma de desarrollar un software mediante la separación de la lógica del negocio de la lógica del diseño. La ventaja principal de esta técnica es que el desarrollo del producto es llevado en varios niveles o capas lo que por lo que en caso de que se produzca algún cambio en el código solo se afectara la capa o el nivel requerido sin tener que revisar en un código mezclado o poco estructurado.

Para lograr sacarle el mayor provecho a la programación por capas se necesita seguir una serie de pasos complejos los cuales primeramente deben ser definidos para cada proyecto en específico, luego deben ser revisados para asegurarse de que el modelo adoptado cumpla con las normas necesarias para que la aplicación sea del agrado del usuario, y por último debe ser implementado por el grupo de desarrollo encargado para tal fin, los cuales siguiendo el modelo propuesto obtienen una herramienta útil para facilitar la labor de programación dividiendo la aplicación en módulos y capas fáciles de pulir. (Lago, Patrones de diseño software, 2007)

El diseño más utilizado en la actualidad es el diseño en 3 niveles (o en 3 capas) el cual se describe a continuación:

Capa de Presentación o Frontera:

La presentación del programa ante el usuario, debe manejar interfaces que cumplan con el objetivo principal de este componente, el cual es facilitar al usuario la interacción con la aplicación. Para esto se utilizan patrones predefinidos para cada tipo de aplicación y para cada necesidad del usuario. La interfaz debe ser amigable y fácil de utilizar, ya que el usuario final es el que se va a encargar de

utilizar el sistema y de dar retroalimentación al equipo de desarrollo en caso de que haya algo que mejorar.

Las interfaces deben ser consistentes con la información que se requiere, no se deben utilizar más campos de los necesarios, así como la información requerida tiene que ser especificada de manera clara y concisa, no debe haber más que lo necesario en cada formulario y por último, las interfaces deben satisfacerlos requerimientos del usuario, por lo cual no se debe excluir información solicitada por el usuario final y no se debe incluir información no solicitada por el mismo.

Dentro de la parte técnica, la capa de presentación contiene los objetos encargados de comunicar al usuario con el sistema mediante el intercambio de información, capturando y desplegando los datos necesarios para realizar alguna tarea. En esta capa los datos se procesan de manera superficial por ejemplo, para determinar la validez de su formato o para darles algún orden específico.

Esta capa se comunica únicamente con la capa de Reglas de Negocio o Control.

Capa de Lógica de Negocio o Control:

- Es llamada capa de reglas de negocio porque en esta se definen todas las reglas que se deben cumplir para una correcta ejecución del programa.
- Es aquí donde se encuentra toda la lógica del programa, las estructuras de datos y objetos encargados para la manipulación de los datos existentes y el procesamiento de la información ingresada o solicitada por el usuario en la capa de presentación.
- Representa el corazón de la aplicación ya que se comunica con todas las demás capas para poder llevar a cabo las tareas. Por ejemplo, mediante la capa de presentación obtiene la información ingresada por el usuario, y despliega los resultados. Si la aplicación se comunica con otros sistemas que actúan en conjunto, lo hace mediante esta capa. También se comunica con la capa de datos para obtener información existente o ingresar nuevos datos.
- Recibe los datos que ingresó el usuario del sistema mediante la capa de presentación, luego los procesa y crea objetos según lo que se necesite hacer con estos datos; esta acción se denomina encapsulamiento.
- Al encapsular los datos, el programa asegura mantener la consistencia de los mismos, así como obtener información precisa de las bases de datos e ingresar en las mismas, solamente la información necesaria, asegurando así no tener datos duplicados ni en las bases de datos, ni en los reportes solicitados por el usuario.

Capa de Datos:

- Es la encargada de realizar transacciones con bases de datos y con otros sistemas para obtener o ingresar información al sistema.
- El manejo de los datos debe realizarse de forma tal que haya consistencia en los mismos, de tal forma los datos que se ingresan así como los que se extraen de las bases de datos, deben ser consistentes y precisos.
- Es en esta capa donde se definen las consultas a realizar en la base de datos, tanto las consultas simples como las consultas complejas parla generación de reportes más específicos.
- Esta capa envía la información directamente a la capa de reglas de negocio para que sea procesada e ingresada en objetos según se necesite, esta acción se denomina encapsulamiento. (Ricardo J. Vargas Del Valle, Programación en Capas.)

Ventajas

Al implementar este modelo de programación, se asegura un trabajo de forma ordenada y separada, debido a que sigue el principio de “divide y vencerás”.

Cada capa está dividida según su funcionalidad cuando se quiere modificar el sistema basta con cambiar un objeto o conjunto de objetos de una capa. Esto se llama modularidad.

Desventajas

Cuando se implementa un modelo de programación en capas, se debe llegar a un balance entre el número de capas y subcapas que componen el programa. Este debe ser el necesario y suficiente para realizar un trabajo específico con eficiencia y ser lo más modular posible. De lo contrario se tiene una serie de desventajas como: pérdida de eficiencia, realización de trabajo innecesario o redundante entre capas, gasto de espacio de la aplicación debido a la expansión de las capas, o bien una alta dependencia entre los objetos y capas que contradice el objetivo principal del modelo (Ricardo J. Vargas Del Valle, Programación en Capas.).

1.14. Conclusiones Parciales.

En este capítulo se describió una serie de procesos y características que intervienen en la planificación docente, se definieron conceptos claves para el objeto de estudio así como se desarrolló un análisis de algunos sistemas de planificación académica existentes con características similares al que se pretende desarrollar. Se realizó un estudio y selección de las principales

metodologías y herramientas de desarrollo de software lo cual es de suma importancia para el logro de un elevado nivel de productividad en el desarrollo del sistema que se propone.

Como resultado del estudio realizado se arribó a las siguientes conclusiones:

- Utilizar como metodología de desarrollo de software RUP en lo referente al flujo de trabajo de Análisis y Diseño.
- Emplear como herramienta CASE Rational Rose para elaborar los diagramas y modelos tanto en el Análisis como el en Diseño del sistema.
- Emplear como sistema gestor de bases de datos a PostgreSQL debido a que posee gran escalabilidad, es una herramienta libre que ofrece seguridad y soporta un número ilimitado de bases de datos entre otras facilidades.
- Utilizar como servidor Web Apache porque es una tecnología gratuita de código fuente abierto y multiplataforma, además de que es altamente configurable.
- Emplear como lenguaje del lado servidor PHP en vista de que es software libre, es rápido, multiplataforma y posee abundante documentación.
- Se recomienda utilizar como lenguaje de programación del lado del cliente al JavaScript por su rapidez y sencillez, así como hacer uso de DHTML para dar cierta interactividad al sistema.

Capítulo II: Características del Sistema.

2.1. Introducción.

En este capítulo se tratarán las características que tendrá el sistema, y para ir profundizando en el estudio y dominio de estas comenzaremos con la investigación de los principales servicios y funcionalidades que este sistema encierra dentro de la facultad, sus objetivos y principales procesos. Se hará una descripción de los procesos más importantes, que permitirá determinar cuáles son las actividades que cumplen las condiciones para ser automatizadas. Por último, se realizará la toma de requisitos funcionales y no funciones de nuestro sistema objeto de estudio. Además de efectuar la descripción de los casos de uso del sistema y la realización de los diagramas pertinentes.

2.2. Aplicación al sistema de las principales actividades de la IR.

Antes de describir la aplicación de las actividades de la IR en el Sistema de Ayuda se muestran dichas actividades en orden de desarrollo para facilitar su comprensión:

Actividades de la ingeniería de requisitos:

- Identificación de requisitos.
- Análisis y negociación de requisitos.
- Especificación de requisitos.
- Modelado del sistema.
- Validación de requisitos.
- Gestión de requisitos.

Seguidamente explicaremos el desarrollo de las seis actividades fundamentales utilizando diferentes técnicas de obtención de requerimientos. Inicialmente se comenzó a desarrollar la primera actividad que tenía como principal objetivo la compresión de los problemas de negocio y la evaluación de las necesidades de todos los involucrados en el proyecto y la creación de una propuesta de solución que responda con dichas necesidades. Para desarrollar dicha actividad, se realizaron una serie de entrevistas a los clientes para familiarizarse con el negocio, se documentan los detalles de las necesidades recogidas para llevar a cabo un intercambio de ideas y proponer nuevas alternativas una vez conocidos los procesos de negocio. Además, se obtuvo un glosario común con el objetivo de reducir los términos ambiguos desde el principio y ahorrar tiempo asegurando que todos los participantes interpretaban correctamente el significado de cada término utilizado y evitando de esta forma los riesgos de malos entendidos entre el equipo de desarrollo.

Unido a esto, se utilizaron los diagramas de actividades como una nueva técnica para profundizar en la comprensión de los procesos de negocio identificados hasta el momento. Con el entendimiento de los procesos de negocios explicados a fondo posteriormente en el presente capítulo dimos por concluida a las dos primeras actividades que trataremos en este epígrafe, pues ya contábamos con las bases para dar paso a la siguiente actividad.

A continuación pasamos a la etapa de captura de los requerimientos. Primeramente se puso al tanto todo el equipo de desarrollo para llevar entre todos el análisis del impacto que puedan causar los cambios en los requerimientos, principalmente en los de mayor prioridad, o sea los que son obligatorios para que el sistema cumpla con sus funciones más importantes. Una vez obtenido los requerimientos, pasamos a la realizar la especificación de estos, definiendo los requerimientos funcionales y no funcionales.

Seguidamente, se llevó a cabo la validación de los requerimientos, donde se demostró que los requerimientos definidos eran los que realmente el cliente necesitaba, se comprobó además que los requerimientos capturados no eran ambiguos, redundantes, ni inconsistentes y que no se había omitido ninguna funcionalidad. En las dos últimas actividades, validación y gestión de requisitos son las relacionadas con la administración de cambios en los requerimientos, las cuales pueden sufrir modificaciones debido a muchas razones, entre las más frecuente encontramos: no se realizan las preguntas correctas a las personas correctas, se cambia el problema que se está resolviendo, los usuarios cambian su forma de pensar o porque cambia el ambiente de negocios, entre otras. Estas modificaciones traen implicaciones como la modificación del tiempo en el que se va a implementar una determinada propiedad y hasta posibles afectación de otros requerimientos.

Seguidamente mostraremos una tabla donde se encuentran las técnicas usadas en cada una de las seis actividades de la Ingeniería de requisitos:

	Actividades de la IR					
Técnicas Usadas	**1**	**2**	**3**	**4**	**5**	**6**
Entrevistas	X	X				
Tormenta de ideas	X	X	X	X	X	X
Glosario	X	X	X	X	X	X
Diagrama de actividad	X	X	X	X		
Caso de Uso	X	X	X	X	X	X
Casa de calidad			X	X	X	X

Tabla 2.1. Actividades de la Ingeniería de Requisitos vs Técnicas Usadas.

2.3. Principales procesos objeto de automatización.

A continuación se listan los principales procesos objeto de automatización:

La gestión de solicitud de locales libres por parte del personal docente ya sean profesores o estudiantes. Los locales pueden ser laboratorios, aulas y salones de conferencias.

La publicación de avisos y noticias de situaciones especiales y actividades que afecten con la distribución del horario docente existente hasta el momento.

Gestionar todo tipo de información referente a las distribuciones de los turnos de clases, local en donde se imparte, profesor y momento en que se realiza la clase, todo esto orientado a organizar e informar a la comunidad docente de la Facultad 3.

La gestión para otorgar locales para diferentes actividades docentes de parte del planificador docente de la Facultad 3.

El flujo de información relacionado con las afectaciones docentes de la facultad.

2.3.1. Descripción general de los procesos.

El proceso inicia con el curso docente, en el cual los profesores deben plantear las afectaciones a sus respectivos jefes de departamentos, quienes son responsables de hacerle llegar a el planificador el listado de las afectaciones, una vez definido estos elementos y conociendo la disponibilidad de locales, cantidad de grupos, años o grupos que darán clases en la sesión mañana o tarde y el plan calendario de cada una de las asignaturas, se procede a planificar el horario.

Este proceso de planificación es muy dinámico, debido a que las afectaciones de los profesores van cambiando a medida que avanza el curso, por reajuste en los planes calendarios de las asignaturas y por actividades orientadas desde la dirección de la universidad, no planificadas desde el principio. A medida que van surgiendo estos cambios el planificador atiende todas las solicitudes planteadas por parte de los jefes de departamentos docentes, tratando de efectuar los cambios en el horario docente de forma tal que cumpla con las necesidades reales del profesorado y no se afecte el proceso de formación.

Las solicitudes que pueden implicar cambios en el horario, puede ser de local, dígase aula, salón de conferencia o laboratorio. En el caso de ser profesor, este puede realizar también una solicitud de turno a recuperar o consulta sobre una determinada asignatura para lo que tiene que ofrecer los siguientes datos: la asignatura que imparte, grupo o grupos que atiende, nombre, departamento (etc.). En caso de los estudiantes además de solicitar locales para consulta, también puede solicitarlo para reuniones del C/B o turnos FEU u otro tipo de actividad docente. A lo antes expuesto hay que añadir que el vicedecanato tiene que informar cada vez que surge un nuevo cambio.

Otra de las acciones que perjudican la estabilidad del horario docente son la perdida de turnos, reajuste en la duración de los turnos y cambio en los tiempos intermedio entre turno y turno. Para tratar estos problemas se envían constantemente avisos que son de carácter urgente tanto para estudiantes como para profesores, y otros son tratados en forma de noticias que en un futuro pueden afectar el curso de las actividades docentes como exámenes, actividades deportivas, docentes, políticas o recreativas.

2.4. Datos a tratados en el sistema.

En el sistema de ayuda a la planificación docente se tratan todas las informaciones y afectaciones de las actividades docentes de la facultad.

- Publicaciones de Avisos.
- Publicación de Noticias.
- Información sobre las solicitudes del personal docente.
- Información sobre los locales de la facultad.
- Información sobre el personal docente de la facultad.
- Información sobre las afectaciones docentes de la facultad.
- Búsquedas sobre la distribución y organización de las actividades docentes.
- Búsquedas sobre los locales de la facultad.
- Mapa del Sitio.
- Sitios propiamente relacionados con la docencia.

2.5. Propuesta de Sistema.

El sistema tiene como objetivo primeramente la gestión de toda la información relacionada con las actividades docentes en la facultad. Por tanto, debe proporcionar acceso para todos los estudiantes y profesores de la institución. Además se toma en cuenta que se realizará la aplicación para usuarios que tienen un cierto conocimiento sobre las tecnologías por ser implementado para una universidad de ciencias informáticas por lo que el principal objetivo será enfocado hacia la facilidad de acceso y usabilidad del sistema.

El sistema deberá automatizar una serie de procesos que en la facultad se realizan de forma manual y muy lentos, para la dinámica en que vive el personal docente de la universidad, y susceptible a errores. Su funcionamiento está basado, fundamentalmente orientado a los servicios que brinda hacia la comunidad, entre estos encontramos: la solicitud para la asignación de locales libres, cambio de actividades docente y cambios propiamente relacionados con las afectaciones docente, proporcionar información sobre los locales y el personal docente que imparte una determinada

actividad docente, también debe permitir realizar consultas, la publicación de avisos y noticias de múltiples temas relacionada con actividades que puedan afectar la organización docente en la facultad.

También debe de existir la posibilidad de aprobar los diferentes temas agrupados en la sección: avisos y noticias. La aplicación tendrá acceso a muchos de los servicios que brinda la universidad, todos estos que estén explícitamente relacionados con la docencia como ayuda en el proceso de estudio y superación de los estudiantes como: Teleformación, sitios propios de las asignaturas, sitios de investigación y foros relacionados con estas actividades, a partir de la dirección URL para acceder a ellos.

2.6. Modelo del Negocio.

El primer paso para la automatización de un sistema de software es comprender los procesos que en ella se desarrollan para lograr un mejor entendimiento del problema a resolver, esto se logra con una buena comunicación entre los clientes y el equipo de desarrollo creando de esta manera un marco propicio para la modelación de negocio.

El Modelo del Negocio es un modelo de casos de uso que describe los procesos del negocio en términos de casos de uso y actores. Es una técnica para comprender los procesos de negocio de una organización como funciones que se desarrollan en el ambiente o entorno que definimos como negocio. El objetivo del modelado del negocio es describir los procesos existentes u observados con el propósito de comprenderlos y especificar qué procesos del negocio soportará el sistema.

2.6.1. Reglas del Negocio.

- El profesor con clase a 3er turno no debe dar clase a 4to turno.
- Los profesores externos no debe impartir clases a 6to.
- No se puede planificar la asignatura de EF a 3er o 4to turno.
- No planificar nada en la sesión que se imparte EF.
- Un profesor puede impartir clases a varios grupos de diferentes años.

2.6.2. Actores del Negocio.

El actor de negocio es el rol que juega algo o alguien que tiene cierta interacción con el negocio y logra beneficiarse de los procesos desarrollados en el negocio. Puede ser un individuo, un grupo de personas, una organización, máquina o sistema de información.

Actores del Negocio	Justificación
Personal Docente (estudiantes y profesores)	Realiza solicitud y búsqueda de locales así como de información referente al horario docente. Además tiene acceso a los avisos y noticias publicados en la Web, también puede solicitar cualquier tipo de información de las actividades docentes en general.
Jefe Departamento	Es el encargado de solicitar cambios de las actividades docentes al vicedecanato, además de realizar la solicitud de cambios relacionado con las afectaciones recogidas a cada profesor por departamento.

2.6.3. Trabajadores del Negocio.

Representa a personas, o sistemas (software) dentro del negocio que son las que realizan las actividades que están comprendidas dentro de un caso de uso.

Trabajadores del Negocio	Justificación
Vicedecano docente	Es el encargado de aprobar la propuesta de cambios de las actividades docentes introducidas por el planificador y de los avisos y noticias. Define pautas que debe seguir el planificador para la realización del horario. Tiene la potestad de modificar los cambios efectuado por el planificador así como la potestad de insertar avisos y noticias. También puede realizar muchas consultas de información necesarias para la asignación de las actividades docentes de la facultad.
Planificador	Es el encargado de introducir los datos y actualizar la información referente al horario docente. Además de publicar las noticias y avisos publicados en la Web. También puede realizar muchas consultas de información necesarias para la asignación de las actividades docentes.

2.7. Casos de uso del Negocio.

Los casos de uso del negocio son una secuencia de acciones que producen un resultado observable para los actores de negocio, representan concretamente procesos de negocio.

2.7.1. Casos de uso del negocio objeto de estudio.

- Solicitar cambio de actividades docentes
- Solicitar un local libre
- Consultar Información
- Enviar Información.
- Enviar Horario.
- Crear Horario.
- Solicitar cambios en las afectaciones docentes.

2.7.2. Entidades del Negocio.

Afectaciones, Informaciones Docentes, Horario, Locales, Grupos Docentes, Plan Calendario.

2.7.3. Descripción de los casos de uso del negocio.

Descripción del caso de uso Solicitar Local Libre:

Nombre del Caso de Uso:	**Solicitar local libre**
Actores del Negocio:	Personal Docente (estudiantes y profesores)
Propósito	Realizar un pedido de local para una actividad docente ya sea aula, laboratorio o salón de conferencia.
Resumen: El caso de uso inicia cuando un estudiante o profesor de la facultad se presenta en el vice-decanato docente para efectuar una solicitud para un local libre para una actividad docente. Entonces es cuando le informa a la (planificador) su situación y es cuando el planificador busca un turno libre del horario para resolver el problema, esto teniendo en cuanto las afectaciones que existen para la asignación en el caso que resuelve.	
Casos de Uso asociados	**-**
Flujo de trabajo	
Acción del actor	**Respuesta de Negocio**
1- El estudiante o el profesor solicita local libre para una actividad docente.	2- El planificador escucha la solicitud de local libre.

4- El estudiante o el profesor según sea el caso informa el motivo de su petición. 8- El interesado escucha la propuesta realizada por el planificador y se va.	3- El planificador pregunta la justificación del pedido. 5- El planificador busca un turno libre para efectuar la asignación. 6- Si no existe ningún problema con el actor en relación a las afectaciones docentes. 7- Se informa el local que existe para realizar la asignación de cambio. 9- Ya asignado el local se envía este a la facultad como una nueva actualización o al grupo afectado en particular.
Prioridad:	**Alto**
Mejoras:	La automatización de las solicitudes de locales libres.
Cursos Alternos:	
Acción del actor	**Respuesta de Negocio**
	6.1- Si presenta alguna afectación docente vuelve al paso 5.

Descripción del caso de uso Cambiar Actividades Docentes:

Nombre del Caso de Uso:	**Solicitar Cambio de Actividades Docentes**
Actores del Negocio:	**Jefe Departamento**
Propósito	Realizar la solicitud de un cambio de actividades docentes anteriormente planificada y organizada por vice-decanato docente de la facultad.
Resumen:	

El caso de uso inicia cuando el jefe de departamento realiza una solicitud para modificar alguna actividad docente en el horario porque le coincide con alguna otra actividad, para recuperar alguna o por otro tipo de interés de algún profesor o alumno ayudante de su departamento. Entonces es cuando le informa al (vice-decano docente) su situación, este busca el turno en el horario para efectuar el cambio y resolver su problema, esto teniendo en cuenta las afectaciones que existen para la asignación en el caso que resuelve. El cambio de actividad docente puede ser por otra actividad en concreto o hacia un turno que este libre en ese momento.	
Casos de Uso Asociados	**Solicitar local libre**
Flujo de trabajo	
1- El Jefe de Departamento realiza una solicitud para cambiar alguna actividad docente. 4- Se justifica el motivo del cambio. 11- Se informa del resultado y se va.	2- El vicedecano docente escucha la solicitud de cambio. 3 - Se pide la justificación del cambio a realizar. 5- El vicedecano docente valora la aceptación del cambio en el horario. 6- Si acepta cambiar informa a El planificador para que realice el cambio. 7- El planificador se informa del cambio a realizar. 8- Si no hay problemas con las afectaciones docentes se actualiza el horario con el cambio. 9- El vicedecano docente envía un correo a los grupos o al grupo afectado por el cambio. 10- Informa al cliente del cambio realizado.
Prioridad:	**Alto**
Mejoras:	La automatización de las solicitudes de cambio de Actividades Docentes.

Cursos Alternos:	
Acción del actor	**Respuesta de Negocio**
8.1- Si el cambio es por un turno libre en vez de otra actividad docente nos remitimos al caso de uso Solicitar Local Libre.	

Descripción del caso de uso Consultar Información:

Nombre del Caso de Uso:	**Consultar Información**
Actores del Negocio:	**Estudiante, Profesor**
Propósito:	Consultar Información de los locales, del personal docente, además de información relacionada con Avisos y Noticias emitidas por el vicedecanato docente de la facultad.
Resumen: El caso de uso inicia cuando el planificador envía el horario docente por correo e Imprime este en el mural de Información junto con: avisos, urgente y noticias, entonces es cuando algún estudiante o profesor de la facultad necesita consultar algún tipo de Información relacionada con el horario, los locales con los que cuenta la facultad para las actividades docentes, también pueden consultar algún tipo de aviso urgente o alguna noticia relaciona con las actividades docentes que planifica el vice-decanato docente. Para esto los profesores acuden al horario que se envía por correo o acuden al mural de información que actualiza el vicedecanato. Ahora en caso de no encontrar la información que necesita se remite al planificador del vicedecanato y plantea sus inquietudes y problemáticas.	
Casos de Uso asociados	**-**
Flujo de trabajo	
Acción del actor	**Respuesta de Negocio**

1. El estudiante o el profesor solicita información ya sea relacionada con el horario docente o también pueden consultar algún tipo de aviso urgente o alguna noticia relaciona con las actividades docentes. 3. Para esto acude al horario docente o al mural que el vicedecanato imprime con informaciones. 4. Si la información que necesita no es encontrada se remita al planificador. 5 El estudiante se presenta en el vicedecanato docente para solicitar información. 7- El estudiante o el profesor es informado y se retira.	2 El planificador envía el horario (Por Correo) e imprime el horario junto a avisos urgentes y noticias en el mural de Información de la facultad. 6. El planificador atiende la solicitud de información y resuelve su problema. 8. El planificador consulta con el vice-decano docente si la información merece ser publicada para la facultad. 9. Si se decide publicar la información, el planificador envía un correo con la información tratada a toda la facultad.
Prioridad:	**Alta**
Mejoras:	Automatizar las consultas de información del personal docente así como la forma de contactar cualquier inquietud.
Cursos Alternos:	
Acción del actor	**Respuesta de Negocio**
	8.1- Si el vice- decano decide que no es necesario publicar la información se omite el paso 9.

Descripción del caso de uso Crear Horario:

Nombre del Caso de Uso:	**Crear Horario**
Actores del Negocio:	**Personal Docente**
Propósito:	Agilizar la elaboración de los horarios así como controlar el momento tope para su creación.
Resumen: El caso de uso se inicia cuando es ya tiempo de elaborar un nuevo horario con la llegada del nuevo semestre y es cuando el planificador comienza con la creación de este y de este modo organizar las demandas de los profesores en la planificación docente teniendo en cuenta cada uno de los principales documentos de información.	
Casos de Uso asociados	-
Flujo de trabajo	
Acción del actor	**Respuesta de Negocio**
1- El personal docente solicita la realización de la nueva planificación del horario docente debido a la llegada del nuevo semestre.	2- El planificador comienza a desarrollar la planificación de solicitud del horario. 3- El planificador teniendo en cuenta las afectaciones docentes de los profesores, los Grupos Docentes, los locales con los que cuenta y el Plan Calendario planifica el horario desarrollando una nueva versión para el semestre que recién comienza. 4- Luego consulta la aprobación de la versión realizada con el vicedecano docente. 5- Si es aprobada es elaborado el horario final. 6- Una vez elaborado el horario es cuando procede a enviar el horario por correo a toda la facultad.
Prioridad:	**Alta**

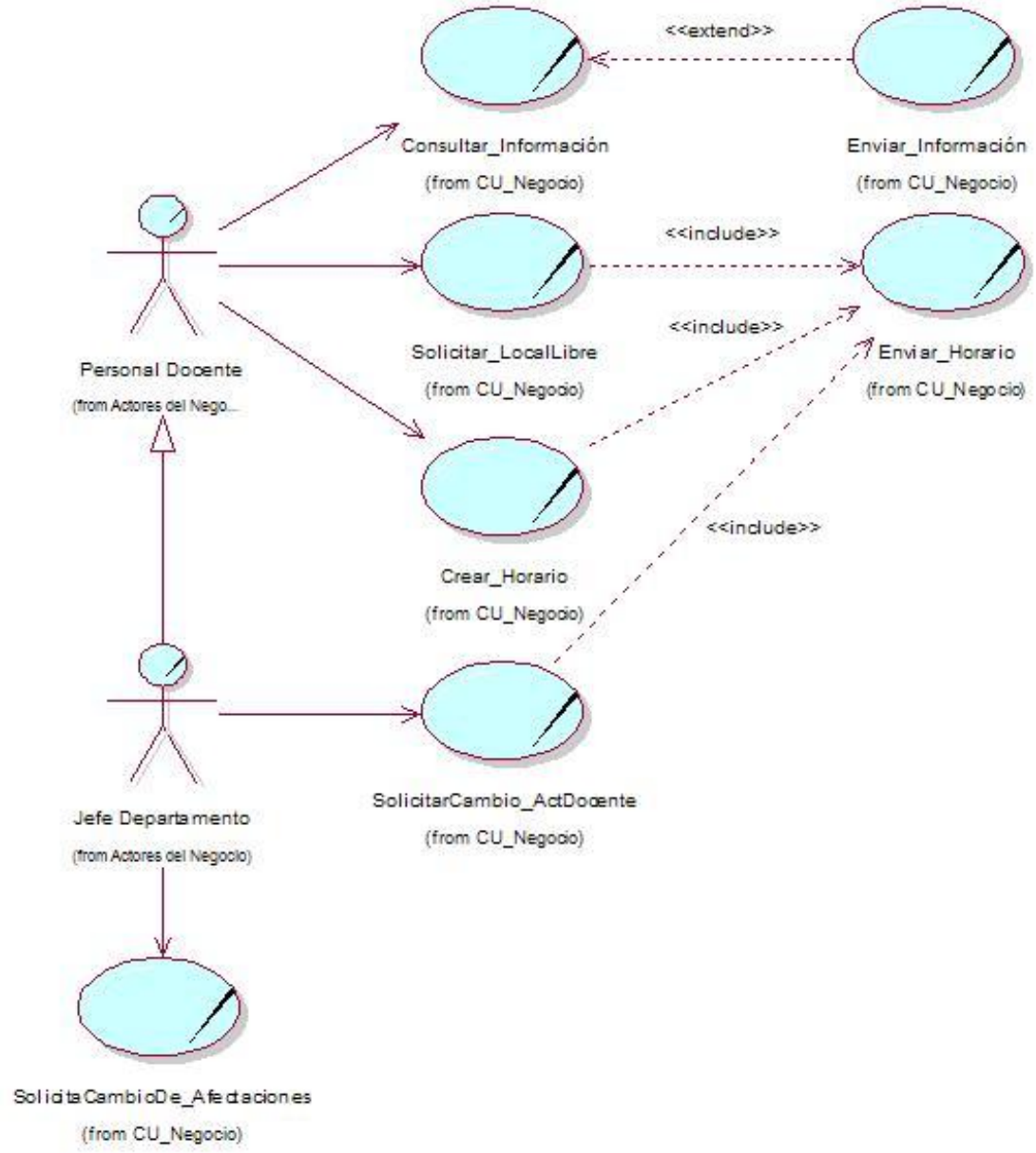
Mejoras:	La automatización de este proceso de negocio ayudará a el planificador en la velocidad de planificación del nuevo horario y de este forma el personal docente pueda contar lo más pronto posible con las nuevas orientaciones.
Cursos Alternos:	

Descripción del caso de uso Solicitar Cambios en las Afectaciones:

Nombre del Caso de Uso:	**Solicitar Cambios en las Afectaciones**
Actores del Negocio:	**Jefe Departamento**
Propósito:	Realizar una solicitud de cambio en las afectaciones docentes al vicedecanato de la facultad.
Resumen: El caso de uso inicia cuando algún profesor o alumno ayudante solicita un cambio de alguna de sus afectaciones docentes al Jefe Departamento, o el mismo Jefe Departamento teniendo conocimiento de alguna situación en particular realiza una solicitud de cambio en las afectaciones docentes de algún miembro del personal docente que trabaja en su departamento. Entonces estando de acuerdo con el cambio habla con el planificador para modificar la organización de estas afectaciones y controlar de esta forma la disponibilidad de personal de la facultad. Toda esta organización deberá ser consultada de antemano con el Vice-decano docente para aprobar las modificaciones efectuadas.	
Casos de Uso asociados	-
Flujo de trabajo	
Acción del actor	**Respuesta de Negocio**
1. Solicita el cambio de las afectaciones docentes.	2. Se atiendo o escucha esta solicitud de cambio del Jefe Departamento. 3. Se verifica si entregó las afectaciones docentes propias del departamento que dirige.
5. Se justifica la causa para el cambio.	4. Si las entregó se solicita la causa para el cambio

	en las afectaciones. 6. Se pone al tanto de la entrega y comprueba la necesidad de consultar al vice-decano docente. 7. Si se consulta al vice-decano docente este atiende la situación de duda y verifica la aceptación para el cambio en las afectaciones.
9. Se informa del cambio y se va.	8. Si se acepta el planificador realiza el cambio de en las afectaciones docentes.
Prioridad:	**Alta**
Mejoras:	La automatización de este proceso de negocio ayudará el la velocidad y comodidad de tramite con las afectaciones docentes de la facultad.
Cursos Alternos:	

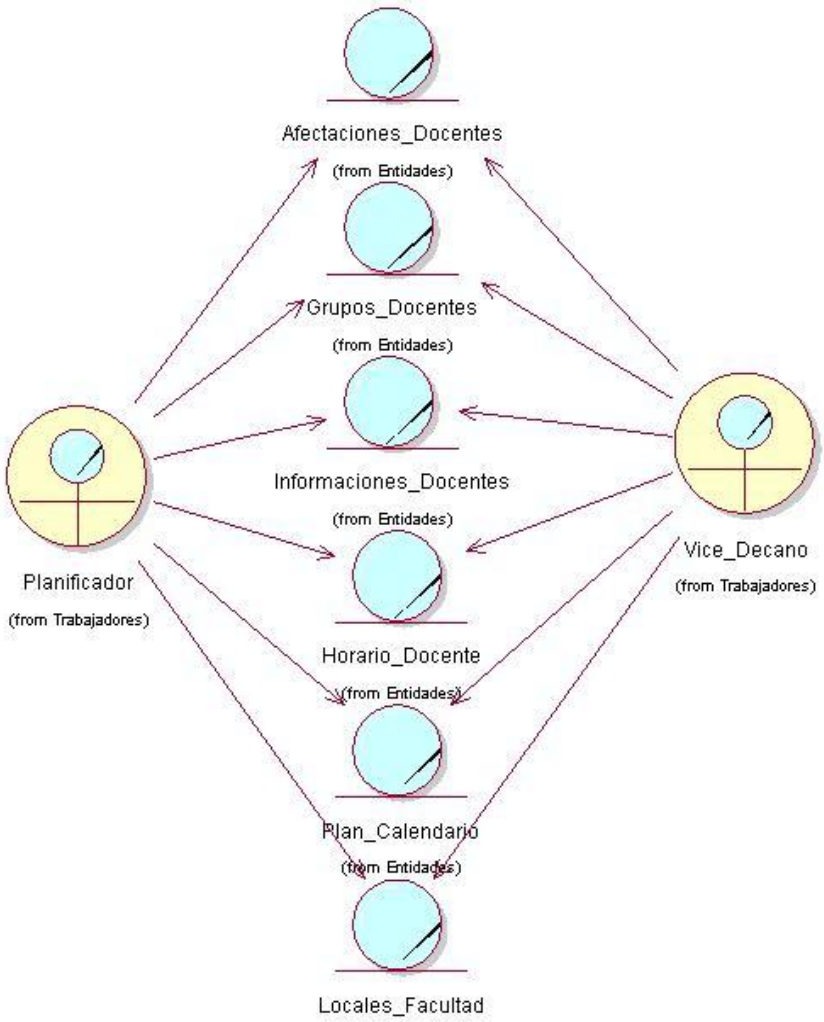
2.7.4. Diagrama de casos de uso del negocio.



2.7.5. Modelo de objetos del Negocio.

El diagrama de clases del modelo de objetos es el conjunto básico de objetos dentro de los que encontramos los trabajadores del negocio y las entidades que son tratadas así como sus relaciones.

2.7.6. Diagrama de clases del Modelo de objetos del Negocio.



2.8. Diagrama de Actividades

Un diagrama de actividades es un diagrama diseñado para mostrar una visión simplificada de lo que ocurre durante una operación o un proceso. Es una extensión de un diagrama de estados, pero el diagrama de actividades resalta precisamente a las actividades. A cada actividad se le representa con un rectángulo con las esquinas redondeadas, esta dividido en calles y muestra como se utilizan las entidades que apoyan la realización de los procesos de negocio [Ver Anexo 1].

2.9. Requisitos funcionales y no funcionales del Sistema.

Los requisitos son los encargados de desarrollar la solución que realmente necesita el cliente, son las funcionalidades que realmente hacen falta. Esto teniendo en cuenta las funcionalidades que se ofrecen a cada uno de los actores del sistema. Concretamente son una condición o capacidad que debe estar presente en un sistema o componente de sistema para satisfacer un contrato, estándar, especificación u otro documento formal.

2.9.1. Requisitos Funcionales.

Los requerimientos funcionales definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas.

Paquete Administrar Sistema

RF1 Autenticar Usuario

RF2 Administrar Sistema

Paquete Horario Docente

RF1 Crear Horario

RF1.1 Borrar todo el Horario

RF1.2 Guardar los datos

RF2 Buscar Horario

RF3 Mostar Horario

RF4 Mostrar locales libres

RF5 Registrar Usuario

RF6 Gestionar Horario

RF6.1 Adicionar actividad docente

RF6.2 Eliminar actividad docente

RF6.3 Cambiar actividad docente

RF7 Buscar local libre

RF8 Notificar Asignación del local

RF9 Atender Solicitud de local libre

RF10 Notificar Cambio de actividad docente

RF11 Atender Solicitud de Cambio de actividad docente

RF12 Solicitar local libre

RF13 Solicitar cambio de actividades

RF14 Consultar (responsable - local)

RF15 Gestionar Grupo

RF15.1 Adicionar grupo

RF15.2 Eliminar grupo

Paquete Afectaciones Docentes

RF1 Mostar afectaciones docentes

RF2 Gestionar Afectaciones

RF3 Modificar Afectaciones

RF3.1 Adicionar afectación

RF3.2 Modificar afectación

RF3.3 Eliminar afectación

RF4 Atender solicitud

RF5 Notificar solicitud

RF6 Buscar afectación

RF7 Realizar solicitud

Paquete avisos docentes

RF1 Mostar avisos docentes

RF2 Gestionar aviso

RF2.1 Adicionar aviso

RF2.2 Eliminar aviso

RF2.3 Modificar aviso

RF3 Buscar avisos docentes

Paquete locales - facultad

RF1 Gestionar Local

RF1.1 Adicionar Local

RF1.2 Eliminar Local

RF1.3 Bloquear local

RF2 Buscar Local

RF3 Mostrar Local

Paquete -Consultas

RF1 Dado un profesor en un grupo determinado:

RF1.1 Actividades docentes en un día

RF1.2 Actividades docentes en la semana

RF2 Fondo de tiempo del profesor

RF3 Dado un local:

RF3.1 Actividades docentes con su profesor por día

RF3.2 Actividades docentes con su profesor por semana

RF4 Dado un Departamento

RF4.1 Profesores con la asignatura que imparte y grupos que atiende.

RF4.2 Jefe Departamento.

Paquete -Profesores

RF1 Mostar profesores

RF2 Gestionar profesor

RF2.1 Adicionar profesor

RF2.2 Eliminar profesor

RF2.3 Modificar profesor

RF3 Buscar profesor

2.9.2. Requisitos no Funcionales.

Los requerimientos no funcionales tienen que ver con características que de una forma u otra puedan limitar al sistema, como el rendimiento ya sea en tiempo o espacio, interfaces de usuario, fiabilidad en lo que a robustez del sistema, disponibilidad del equipo se trata, además de esto se incluye el mantenimiento, seguridad, portabilidad, estándares entre otras propiedades o cualidades que el producto debe tener.

RNF1 **Apariencia**

RNF1.1 Utilizar distintas tonalidades del color que distingue a la facultad (verde).

RNF1.2 Interactivo con el usuario.

RNF2 **Rendimiento**

RNF2.1 El sistema de ayuda tratará toda la información relacionada con la planificación docente, solicitudes y consultas de todo tipo de una forma rápida.

RNF2.2 Debe lograr efectuar las gestiones de información y solicitudes sin realizar mucha navegación por el sitio Web, así como disponible todo el tiempo.

RNF3 **Usabilidad**

RNF3.1 El sistema de ayuda podrá ser usado por cualquier estudiante y profesor de la facultad, que como estudiantes de la UCI tendrá algún conocimiento básico sobre el trabajo en la Web.

RNF3.2 El tamaño del texto deberá tener un tamaño mediano para que pueda ser observado a una distancia no menor de un metro.

RNF3.3 Debe tener un diseño sencillo.

RNF4 **Soporte**

RNF4.1 Deberá ser fácil en su mantenimiento, asequible y de configuración sencilla para los usuarios de la facultad.

RNF4.2 Deberá permitir adicionar funcionalidades en forma de algoritmos que puede ayudar al sistema de ayuda en determinadas prácticas o ya sea a la hora de generar horarios.

RNF5 **Portabilidad**

RNF5.1 El sistema de ayuda deberá ser multiplataforma, se podrá montar tanto para Windows como en Linux. Podrá usar gestores como PostgreSQL, MySQL o Oracle entre otros, aunque se busca siempre una portabilidad hacia software libre de cada una de las parte del sistema de ayuda.

RNF6 **Confiabilidad**

RNF6.1 Deberá estar listo en todo momento debido a su alta importancia en la planificación docente de la facultad.

RNF6.2 Alta capacidad de restaurarse de la fallas lo más rápido posible.

RNF7 **Seguridad**

RNF7.1 Se limitará el acceso a la administración del sistema, solo tendrán acceso a este el vice-decano docente y el planificador docente.

RNF8 **Software**

RNF8.1 SGBD: PostgreSQL preferentemente.

RNF8.2 Servidor Web: Servidor Web Apache.

RNF8.3 En clientes: Navegador FireFox.

RNF9 **Hardware**

Para el desarrollo:

RNF9.1 Pentium IV.

RFF9.2 256 de memoria RAM o superior.

RNF9.3 Disco duro con capacidad de 60 GB o superior.

Para la explotación – Clientes:

Pentium IV.

256 de memoria RAM o superior.

RNF10 **Interfaz**

RNF10.1 La interfaz del sistema será legible.

RNF10.2 Fácil de usar.

2.10. Actores del sistema de ayuda para la docencia.

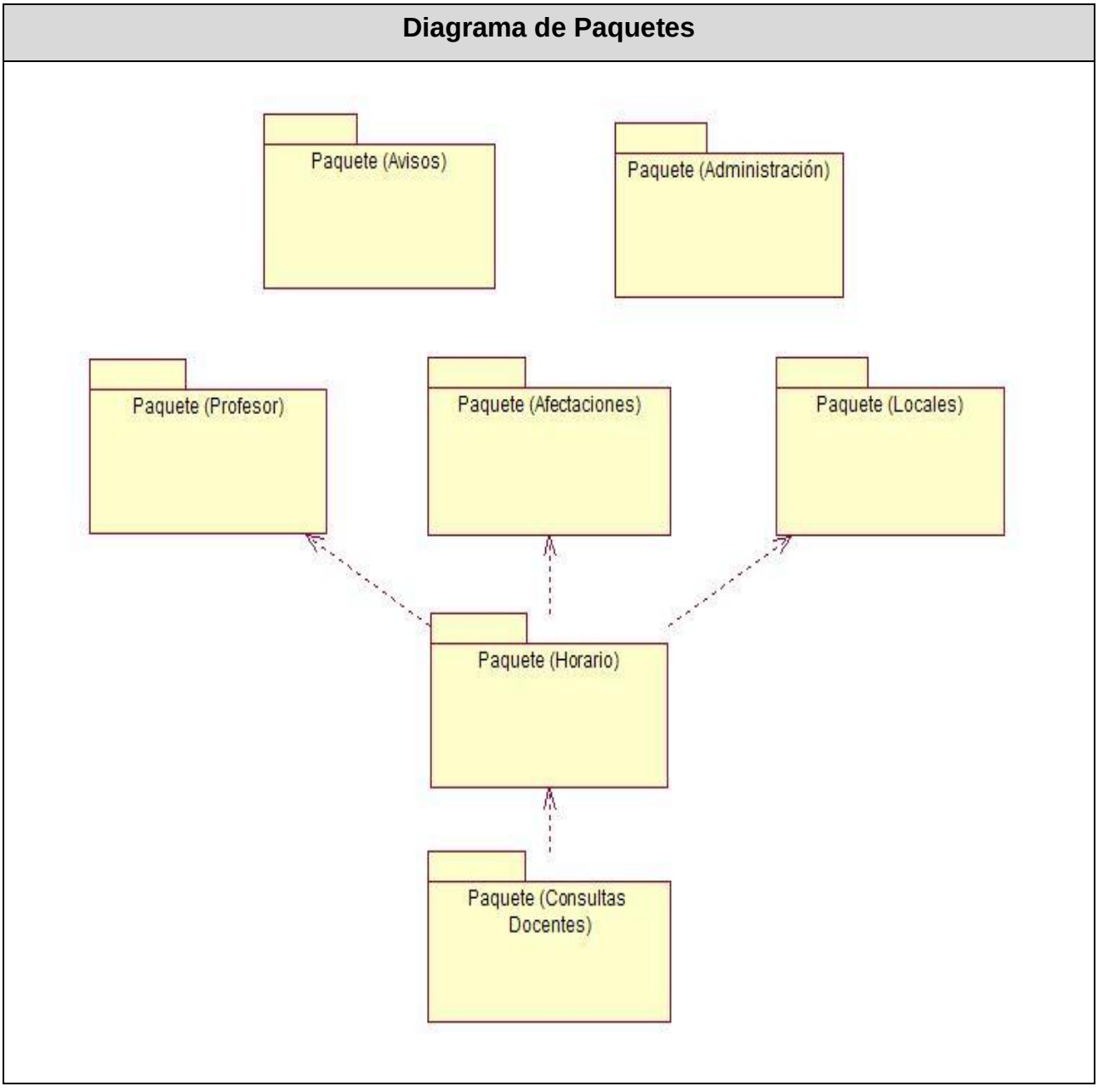
En el Modelo de casos de uso se describe lo que hace el sistema para cada tipo de usuario. Cada uno de estos se representa mediante uno o más actores. También se representa mediante uno o más actores cada sistema externo con el que interactúa el sistema, incluyendo dispositivos externos como temporizadores, que se consideran externos al sistema. Por tanto, los actores representan terceros fuera del sistema que colaboran con el sistema.

Actores del Sistema	Justificación
Planificador	El actor Planificador inicia todos los casos de uso que utiliza El planificador en su trabajo en el sistema junto a los que comparte con el actor del sistema vice-decano docente.
Vice-decano docente	El actor vice-decano docente inicia todos los casos de uso que supervisan el trabajo del usuario administrativo junto a otros casos de usos que solamente son iniciados por este con mayor peso.
Usuario Común (Personal Docente)	El actor personal docente inicia los casos de usos para buscar información y realizar solicitudes, además de poder registrase con su propia cuenta, y está compuesto por los profesores y estudiantes de la facultad.
Usuario	Todo humano que interactúa con el sistema.
Jefe Departamento	El actor Jefe Departamento inicia los casos de usos: solicitar cambios de actividades docentes y los casos de uso relacionados con las afectaciones docentes.
Administrador	Todo actor del sistema que participa en la administración de la información del sistema, es decir: El planificador y el vice-decano docente.

2.11. Diagrama de Casos de uso del Sistema.

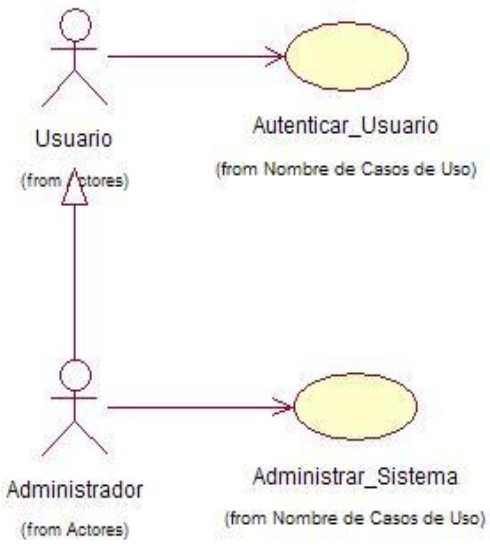
Diagrama de Casos de Uso del Sistema en Paquetes:

Diagrama de Paquetes



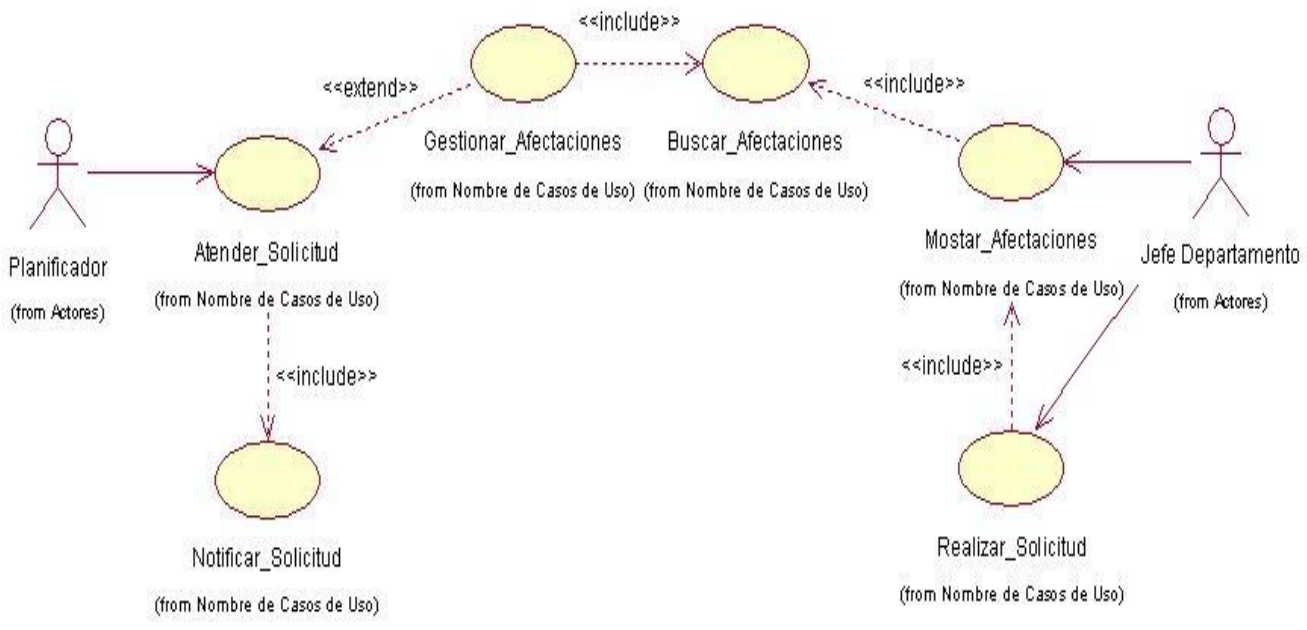
Paquete de Administración:

Diagrama de casos de uso del sistema



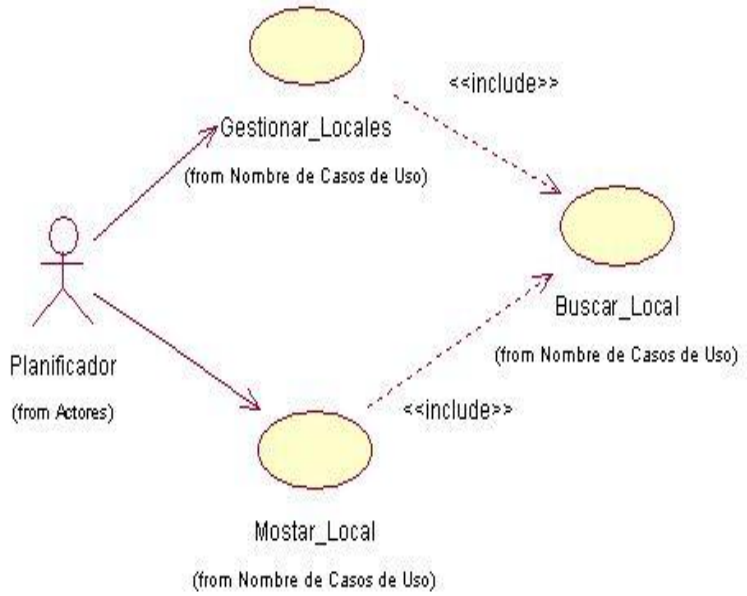
Paquete Afectaciones Docentes:

Diagrama de casos de uso del sistema



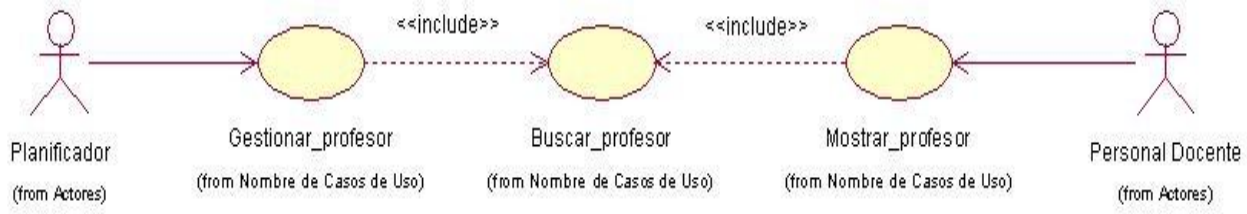
Paquetes Locales – Facultad:

Diagrama de casos de uso del sistema



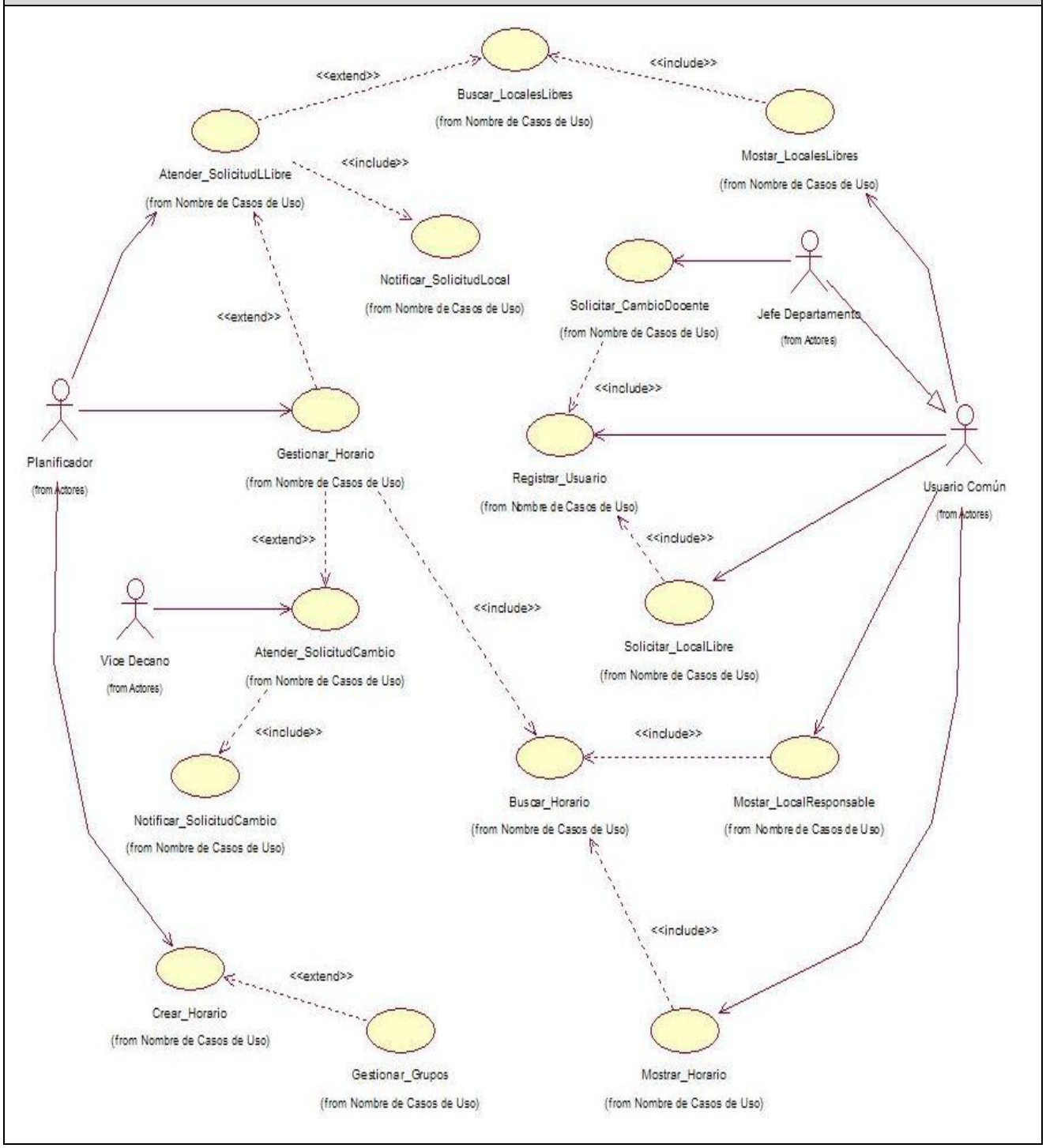
Paquetes Profesores – Facultad:

Diagrama de casos de uso del sistema

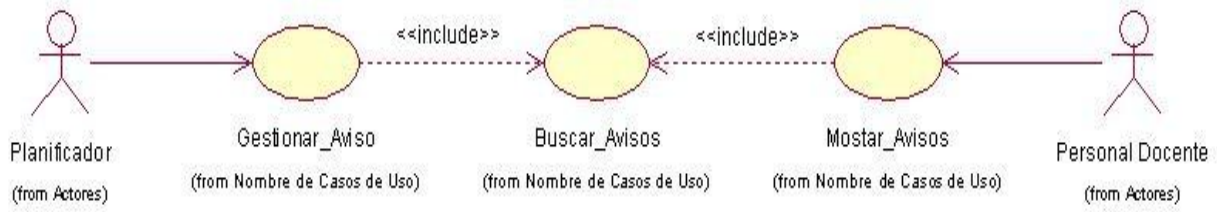


Paquete Horario Docente:

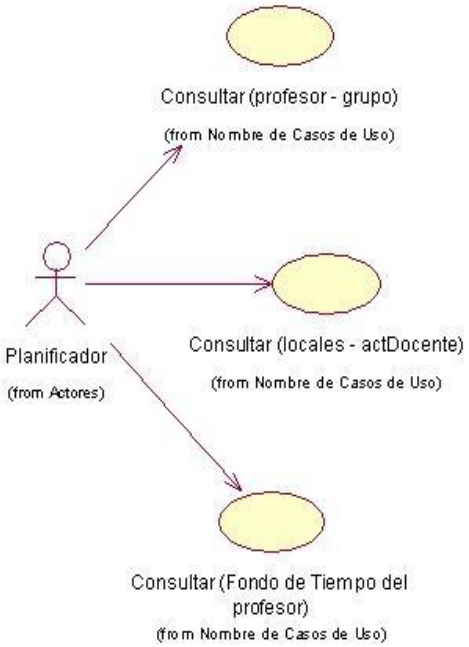
Diagrama de casos de uso del sistema



Paquetes Avisos Docentes:

Diagrama de casos de uso del sistema


Paquete – Consultas Docentes:

Diagrama de casos de uso del sistema


Para remitirse a las DCS [Anexo II].

2.12. Análisis de las características del Sistema de Ayuda.

En este epígrafe se presentará todo el análisis desarrollado una vez terminado el proceso de ingeniería de requerimientos para verificar la calidad con que se realizó cada actividad. Además se mostrarán los resultados obtenidos a partir del criterio de los desarrolladores.

2.12.1. Análisis de los requerimientos.

El principal propósito de este análisis es validar que todos los requerimientos fueron recogidos en al menos un caso de uso así como demostrar la ausencia de ambigüedades de los requisitos capturados.

El equipo de desarrollo, con las ideas tomadas del cliente principal hizo una revisión exhaustiva de cada requerimiento teniendo en cuenta todos los detalles y determinó que la especificación de requerimientos estaba totalmente terminada, por recoger todas las funcionalidades requeridas por el cliente ya sean funcionalidades primarias, secundarias, auxiliares y opcionales para el completo funcionamiento del sistema de ayuda. Y para garantizar que el proceso de análisis de los requerimientos se realizó con efectividad se utilizaron varias métricas para su demostración.

2.12.2. Métricas para determinar la ambigüedad.

$Q_1 = n_{ui} / n_r$ donde,

Q_1: es un valor que cuanto más cerca esté de 1 menor será la ambigüedad de la especificación.

n_{ui}: es el número de requisitos para los que todo el equipo de desarrollo tuvieron la misma interpretación.

n_r: es la cantidad de requisitos en una especificación, y se calcula usando la formula:

$n_r = n_f + n_{nf}$ donde,

n_f: es el número de requisitos funcionales.

n_{nf}: es el número de requisitos no funcionales.

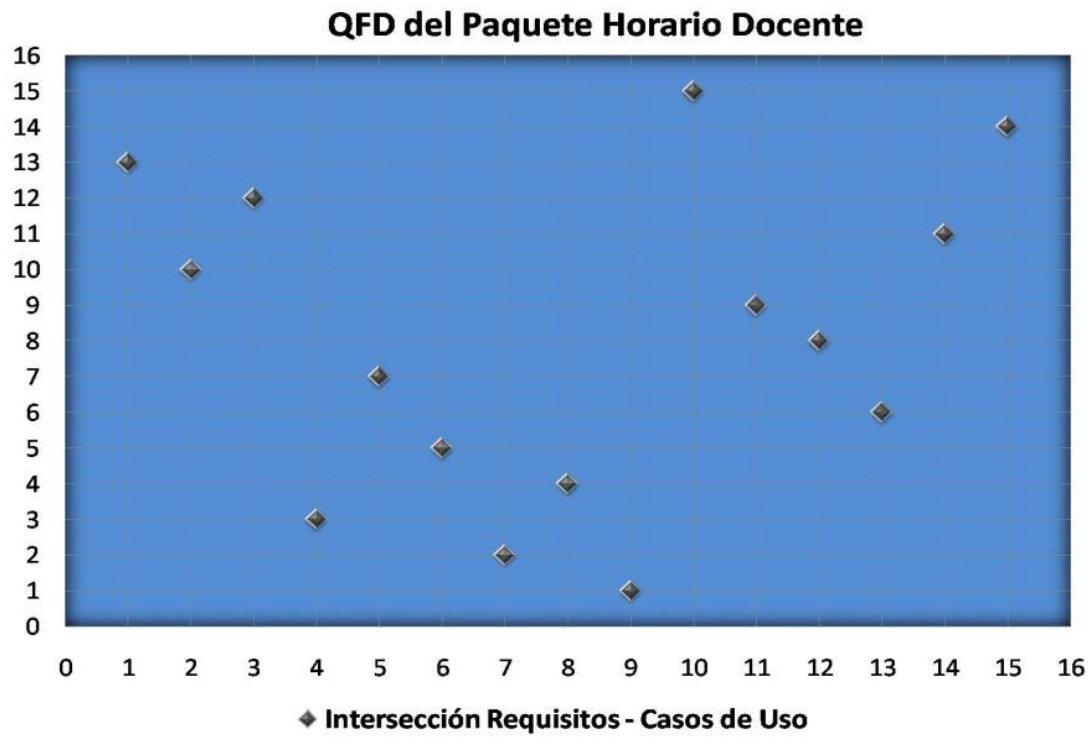
Una vez aplicada esta métrica se determinó que los requisitos no eran ambiguos, ya que el equipo de desarrollo siempre interpretó de igual forma el significado de los requisitos y por lo tanto el valor que tomaría Q1 en este caso es 1.

2.12.3. Métrica para determinar el número de requisitos que no son considerados en ningún CU.

El propósito de esta métrica es confirmar que cada uno de los requerimientos funcionales han sido registrados en al menos un caso de uso que representará su funcionalidad. Para esto se ha tomado la especificación de los requisitos y el modelo de casos de uso del sistema y utilizando la técnica conocida como QFD (Quality Function Deployment) se verificó que todo requerimiento podrá ser implementado a través de algún caso de uso, y que todo caso de uso satisface algún requerimiento.

2.12.4. Esquema QFD (Quality Function Deployment).

El esquema QFD (Quality Function Deployment) es una matriz que representa las casas de calidad, en las cuales las filas representan los “qué”, o sea, la lista de requerimientos, mientras que las columnas representan los “cómo”, es decir, como se llevan a cabo los requerimientos en casos de uso.



Al aplicar el esquema QFD (Quality Function Deployment) en cada paquete de casos de uso del sistema, se comprobó que todos requerimientos tienen al menos una cruz marcada, entonces se pudo comprobar que todos los requerimientos están incluidos en algún caso de uso. Además, después de construir la matriz se concluyó que el error calculado de acuerdo a la métrica y teniendo en cuenta la cantidad de requerimientos totales es del 0%.

2.14.5. Métricas para la evaluación de la calidad del diagrama de CU.

En este caso se usó un modelo de métricas que tiene el objetivo de medir la calidad de los productos intermedios generados en el sistema. El modelo utilizado usa cuatro atributos genéricos de propiedades de calidad, estos son:

- Completitud: Permite determinar el grado en que se ha incluido de forma clara y concisa todos los elementos necesarios para la descripción del aspecto.
- Complejidad: Permite medir el grado de claridad y rehúso del artefacto.
- Consistencia: Permite definir el grado en que los elementos del artefacto representan en forma única y no de forma contradictoria un aspecto de la situación planteada.
- Correctitud: Este atributo genérico permite el grado de adecuación del artefacto para satisfacer los requisitos capturados.

Dado que el diagrama de casos de uso es uno de los artefactos más importantes que se generan en el desarrollo de la ingeniería de requisitos de un producto de software se evaluará la calidad de la funcionalidad a través del diagrama de casos de uso, usando ocho métricas de calidad diferentes.

No	Atributo	Métricas	Umbral
1	Correctitud	Número de casos de uso que no tienen un usuario responsable.	15%
2		Número de casos de uso que deben ser modificados para adecuarlos a la funcionalidad del sistema.	10%
3	Complejidad	Número de elementos del diagrama que requieren reubicación.	30%
4	Completitud	Número de casos de uso que no poseen una descripción extendida.	20%
5		Número de casos de uso donde las acciones del flujo de eventos no están redactadas en función del responsable.	25%
6	Consistencia	Número de casos de uso que tienen un nombre incorrecto.	20%
7		Número de casos de uso donde la descripción del flujo de eventos no está redactada en el lenguaje del usuario.	15%
8		Número de elementos complejos que no tienen separación del flujo básico y de flujos alternos.	20%

Luego de aplicar las métricas anteriores al sistema desarrollado el equipo de desarrollo obtuvo los siguientes resultados:

Métrica #1: La totalidad de los casos de usos han sido inicializado por el actor que le corresponde de manera que cada uno tiene asignado su responsable directo, por tanto no hay ningún caso de uso a la deriva en el diagrama de casos de uso por paquetes del sistema. Por lo que el error es de 0%.

Métrica #2: Con una totalidad de 35 casos de uso especificados, 3 deben ser modificados con el propósito de ajustarlos a la necesidad del sistema, debido a que en algunos momentos las interacciones definidas se desviaban del contenido central que requería describir las funcionalidades del sistema. Por lo tanto el error es de 8.57%.

Métrica #3: El sistema objeto de estudio está dividido en 7 paquetes y cada paquete tiene un diagrama de casos de uso, después de analizar los diagramas se concluye que todos sus elementos están adecuadamente ubicados de manera que facilitan su interpretación. Por tanto no se necesita hacer ninguna reubicación después del análisis y el error es del 0%.

Métrica #4: Con un total de 35 casos de uso el sistema, todos están descritos de forma detallada, después de interactuar y buscar acuerdos con el cliente. Actualmente las descripciones no contienen no contienen ninguna deficiencia, por lo que el error es del 0%.

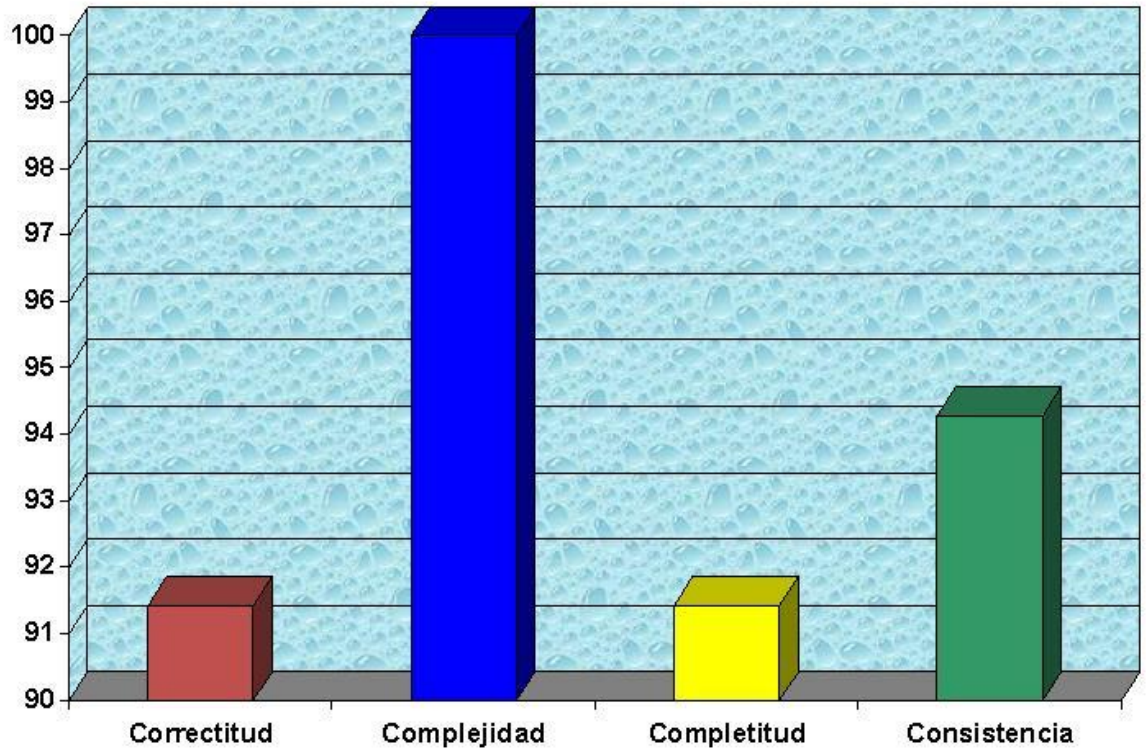
Métrica #5: Se detectaron 3 casos de uso en los cuales existían acciones del flujo de eventos que no que no habían sido redactadas en función del responsable. Se detectaron 2 acciones del flujo de eventos en cada uno de los tres casos de uso por lo que se tuvo un error de 8.57% de acuerdo con la totalidad de casos de uso y la cantidad de casos de uso donde existía problemas en el flujo de eventos.

Métrica #6: Con una totalidad de 35 casos de uso ninguno presentó problemas en cuanto al nombre dado, ya que todos los nombres propuestos a cada uno de los casos de usos representan una expresión verbal en infinitivo describiendo alguna funcionalidad relevante para el usuario, por lo que el error fue de 0%.

Métrica #7: De la totalidad de los casos de uso especificados, solo en 2 no se definía la descripción en el lenguaje del usuario y además no se establecía con claridad el final del caso de uso. Por lo que tuvo un error del 5.71%.

Métrica #8: De un total de 35 casos de usos especificados y después de haber realizado un minucioso trabajo con ellos principalmente para lograr una eficiente estructuración entre el flujo básico y los flujos alternos de manera que se lograra la separara de la funcionalidad básica de la funcionalidad alternativa. Debido a que no se encontró ningún caso de uso complejo que no tuviera separado correctamente el flujo básico de los flujos alternos se obtuvo un error del 0%.

En la siguiente gráfica se muestra el valor alcanzado por cada uno de los atributos después de efectuar el análisis del diagrama de casos de uso por paquetes del sistema.



Los resultados obtenidos demuestran que de forma general los atributos han sido cumplidos. La correctitud fue bien manejada pues todos los casos de uso presentan un responsable y solo fue necesario realizar modificaciones en tres casos de uso para adecuarlo a la funcionalidad del sistema, en cuanto a completitud se han definido todos los procesos que dan funcionalidad al sistema así como los roles de usuario, se han realizado todas las descripciones de los casos de uso en función del responsable considerando los requerimientos funcionales, y los errores fueron mínimos. Por otra parte, la consistencia los casos de uso fueron nombrados correctamente y sus descripciones fueron hechas en un lenguaje fácil de entender separando el flujo básico del flujo alternativo. Por último, la complejidad se comportó en un 100%, esto a pesar de ser grande el sistema y estar dividido por

paquetes el diagrama de casos de uso, todos los elementos de cada paquete se encontraban bien organizados y claramente entendibles.

2.15. Conclusiones Parciales.

En este capítulo se definieron las características de la propuesta de solución obteniéndose una lista de funcionalidades que debe tener el sistema, estas funcionalidades fueron representadas mediante un diagrama de casos de uso junto a la descripción de las acciones que realizan los actores y el sistema en general a través del análisis de los procesos del negocio. Se establecieron los diagramas de casos de uso del sistema y se describieron las acciones de los actores del sistema con los casos de uso que interactúan.

Capítulo III: Análisis y Diseño del Sistema.

3.1. Introducción.

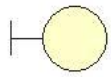
En este capítulo se abordará la propuesta del análisis y diseño del sistema de ayuda, se tratan estos temas un poco más esquemáticos que teóricos. Se utilizan herramientas y metodologías que incluyen principios y diagramas que muestran la forma en que los componentes de software interactúan y se comportan dentro del sistema de ayuda. Además se tratan temas específicos sobre los modelos de análisis y diseño y sobre su construcción como una propuesta RUP para el desarrollo del software; todo esto para lograr al final de la propuesta un eficiente diseño del software para nuestro problema.

3.2. Modelo de análisis.

En la construcción del modelo de análisis se tienen que identificar las clases que describen la realización de los casos de uso, los atributos y las relaciones entre ellas. Con esta información se construye el Diagrama de clases del análisis, que por lo general se descompone para agrupar las clases en paquetes. Esta descomposición tiene impacto por lo general en el diseño e implementación de la solución.

3.2.1. Clases de análisis.

Se centran en los requisitos funcionales y son evidentes en el dominio del problema porque representan conceptos y relaciones del dominio. Tienen atributos y entre ellas se establecen relaciones de asociación, agregación / composición, generalización / especialización y tipos asociativos. RUP propone clasificar a las clases en:

	**Clases Interfaz:** Modelan la interacción entre el sistema y sus actores.
	**Clases Entidad:** Modelan información que posee larga vida y que es a menudo persistente.
	**Clases Controladoras:** Coordinan la realización de uno o unos pocos casos de uso coordinando la actividades de los objetos que implementan la funcionalidad del caso de uso.

3.2.2. Modelo de clases del análisis.

Es un modelo de objetos que describe la realización de los Casos de Uso, y cuales sirven como abstracción del artefacto: Modelo de diseño. El modelo de análisis contiene los resultados del análisis de los casos de uso, instancias del artefacto: clases del análisis.

En adición a esto, en el modelo de análisis es donde se refinan los requisitos, no se toma en cuenta el lenguaje de programación a usar en la construcción, ni se define la plataforma en la que se ejecutará la aplicación ya que el objetivo principal del análisis es garantizar una total comprensión de los requisitos del software y no precisar cómo se implementará la solución propuesta.

Para la construcción del modelo de análisis se tienen que identificar las clases que describe la realización de los casos de uso y sus relaciones entre ellas y con esta información se desarrollan los Diagramas de Clases del Análisis para el sistema objeto de estudio.

3.2.3. Diagrama de clases del análisis.

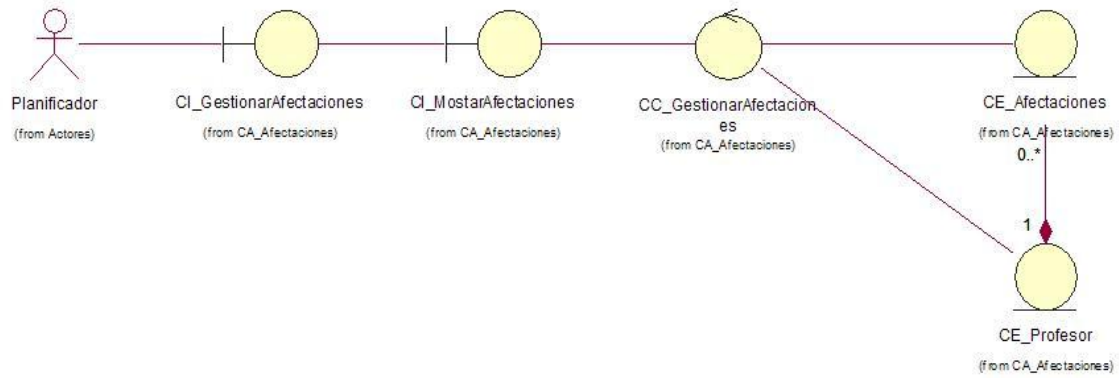
Un Diagrama de clases del análisis es un artefacto en el que se representan los conceptos en un dominio del problema. Representa las cosas del mundo real, no de la implementación automatizada de estas cosas. Son específicamente unos diagramas estáticos que muestran qué es lo que interactúa, pero no cómo interactúa.

A continuación se muestra algunos de los diagramas de clases del análisis, elaborados específicamente para desarrollar la realización de los casos de uso del sistema.

Diagrama de Clases del Análisis Solicitar Cambio – Afectaciones:

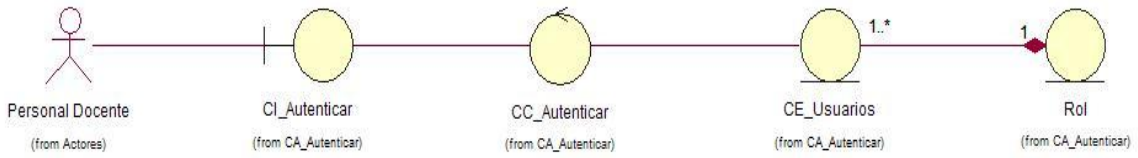
El diagrama muestra las clases que interactúan al solicitar el Jefe de Departamento un cambio de las afectaciones de un profesor al sistema.

Diagrama de Clases del Análisis Mostrar Afectaciones:



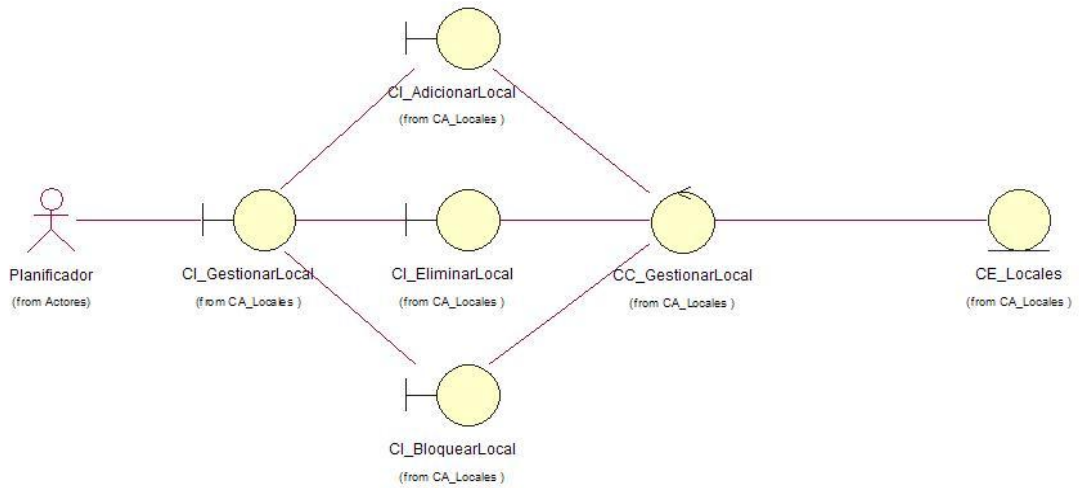
El diagrama muestra las clases que interactúan al solicitar el Planificador que se muestren las afectaciones de un profesor en el sistema.

Diagrama de Clases del Análisis Autenticar Usuario:



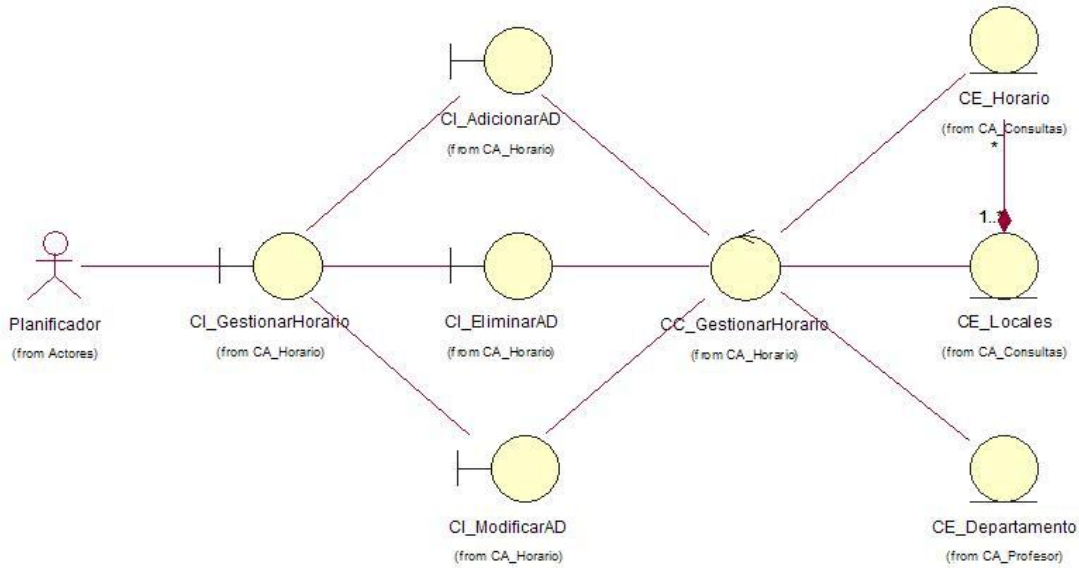
El diagrama muestra las clases que interactúan al autenticarse un usuario en el sistema.

Diagrama de Clases del Análisis Gestionar Locales Docentes:



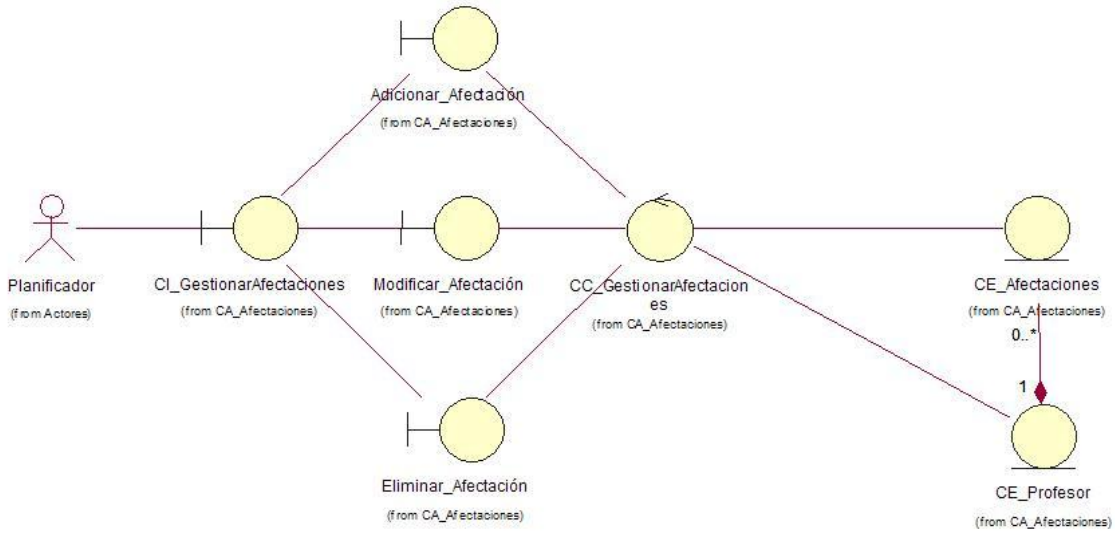
El diagrama muestra las clases que interactúan al procederse a Gestionar Locales Docentes por parte del Planificador en el sistema.

Diagrama de Clases del Análisis Gestionar Horario Docente:



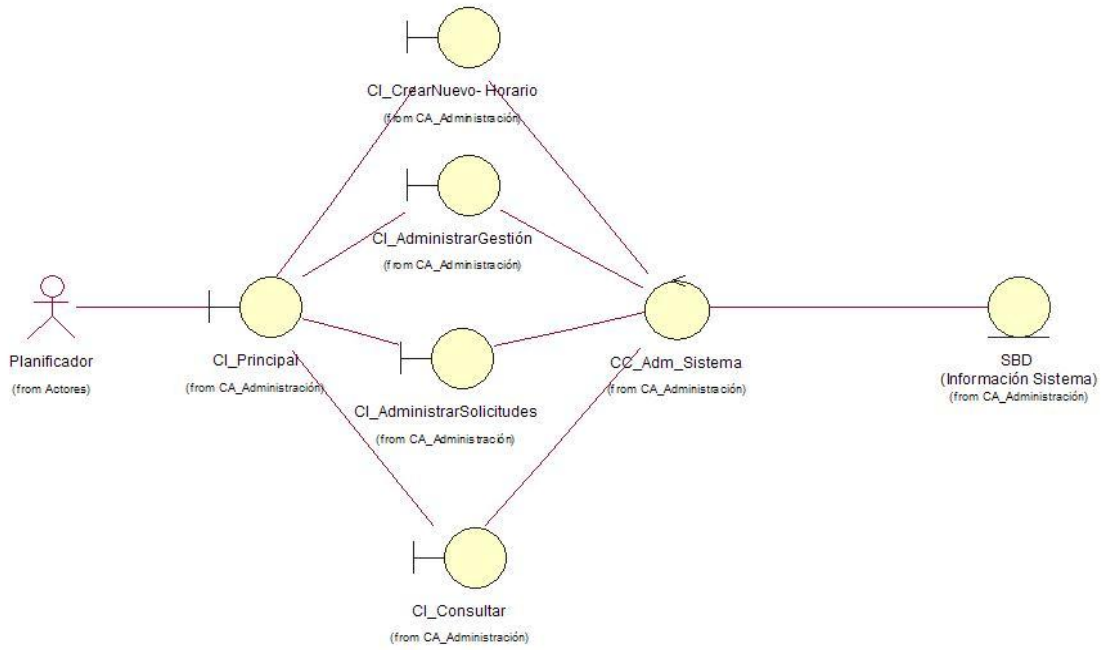
El diagrama muestra las clases que interactúan al procederse a Gestionar el Horario Docente por parte del Planificador en el sistema.

Diagrama de Clases del Análisis Gestionar Afectaciones:



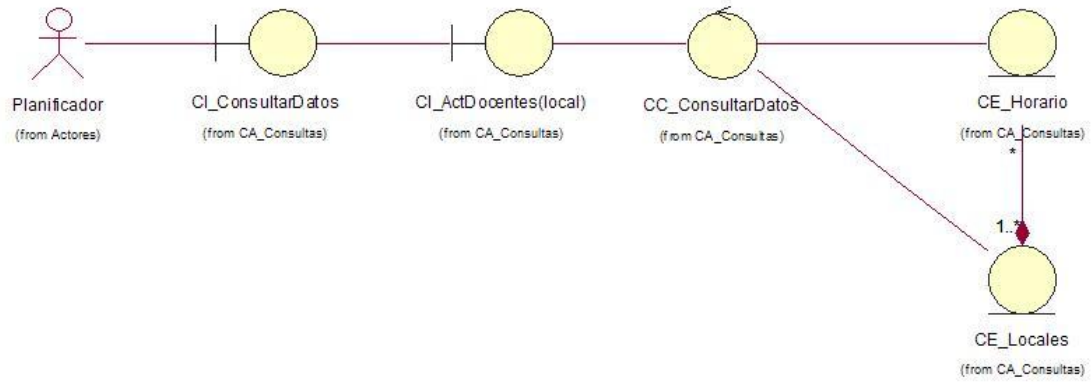
El diagrama muestra las clases que interactúan al proceder el Planificador a la Gestión de las Afectaciones Docentes.

Diagrama de Clases del Análisis – Administrar Sistema:



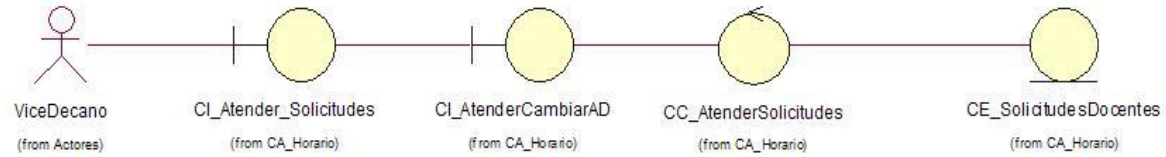
El diagrama muestra las clases que interactúan al proceder el Planificador a Administrar el Sistema.

Diagrama de Clases del Análisis Consultar Actividades Docentes (Local):



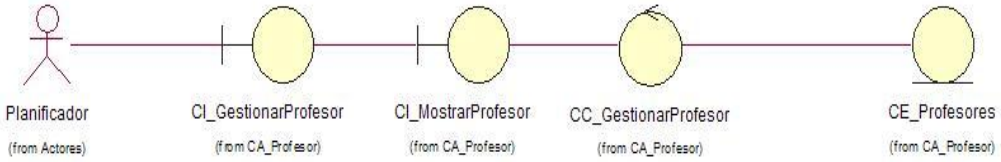
El diagrama muestra las clases que interactúan al proceder el Planificador a Consultar Actividades Docentes dado un Local determinado.

Diagrama de Clases del Análisis Atender Solicitud - Cambiar AD:



El diagrama muestra las clases que interactúan al proceder el vice -decano para Atender una Solicitud de Cambio de Actividad Docente.

Diagrama de Clases del Análisis Mostrar Profesores:



El diagrama muestra las clases que interactúan al solicitar el Planificador que se muestren los Profesores en el sistema.

Para remitirse el resto de los diagramas de clases del análisis ver: [Anexo III].

3.3. Diagramas de Interacción.

Los diagramas de interacción muestran una interacción concreta: un conjunto de objetos y sus relaciones, junto con los mensajes que se envían entre ellos. Modelan el comportamiento dinámico del sistema; el flujo de control en una operación. Describe la interacción entre objetos, los objetos interactúan a través de mensajes para cumplir ciertas tareas. Las interacciones proveen un comportamiento y típicamente implementan un Caso de Uso.

Existen dos tipos de diagramas de interacción en UML:

1. Diagramas de Secuencia (dimensión temporal).
2. Diagrama de Colaboración (dimensión estructural).

3.3.1. Diagramas de Secuencia.

Un diagrama de secuencia muestra las interacciones entre objetos, ordenadas en secuencia temporal durante un escenario concreto. Si los casos de uso tienen varios flujos o subflujos distintos, suele ser útil crear un diagrama de secuencia para cada uno de ellos.

Los Diagramas de Secuencia que describen los distintos flujos de los casos de uso del sistema se encuentran en: [Ver Anexo IV].

3.4. Modelo de Diseño.

El modelo de diseño es un modelo de objetos que adquiere una comprensión en profundidad de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución concurrencias y, tecnologías de interfaz de usuario. Además ayuda a descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo, teniendo en cuenta la posible concurrencia. (Ivar Jacobson, 2000)

3.4.1. Diagrama de clases del Diseño.

Un diagrama de clases de diseño es un diagrama que muestra un conjunto de interfaces, colaboraciones y sus relaciones. Los diagramas de clases de diseño se utilizan para modelar principalmente la vista de diseño estática de un sistema. Esto incluye modelar el vocabulario del sistema, las colaboraciones o esquemas. Los diagramas de clases, son importantes, no sólo para visualizar, especificar y documentar modelos estructurales, sino también para construir sistemas ejecutables, aplicando ingeniería directa e inversa. [Anexo V].

	**Server Page:** Representa la página Web que tiene código que se ejecuta en el servidor. Este código interactúa con recursos en el servidor. Las operaciones representan las funciones del código y los atributos las variables visibles dentro del alcance de la página. Esta clase sólo puede tener relaciones con objetos en el servidor.
	**Client Page:** Una instancia de Página Cliente es una página Web, con formato HTML. Mezcla de datos, presentación y lógica. Son interpretadas por el navegador. Sus atributos son las variables declaradas dentro del script que son accesibles para páginas cualquier función dentro de la página. Cada página cliente es construida por una sola página de servidor.

	**Form:** Colección de elementos de entrada que son parte de una página cliente. Se relaciona directamente con la etiqueta de igual nombre del HTML. Sus atributos son los elementos de entrada del formulario (Text Field, Text Area, Button, Label, Radio Button, Radio Group, Select, Check Box y Hidden Fields).
	**Build:** Representa una asociación especial que relaciona las páginas cliente con las páginas servidor, de forma general se expresa como que las páginas que se encuentran en el servidor construyen las páginas en el cliente. Debe ser una relación direccional, donde una página servidor puede construir una o más páginas cliente.
	**Submit:** Es la relación que se crea siempre entre una página servidor y un formulario, a través de esta relación el formulario manda los valores de sus campos al servidor, para ser procesados por la página servidor.

Contiene normalmente los siguientes elementos:

- Clases
- Interfaces
- Colaboraciones
- Relaciones de dependencia, generalización y asociación.

3.5. Extensiones UML para Web utilizadas en el sistema.

UML posee una extensión para el modelado de aplicaciones Web, usada para el diseño de las clases, dicha extensión utiliza diferentes estereotipos que permiten definir un nuevo significado de la semántica para el elemento a modelar, los estereotipos más usados son:

3.6. Patrones aplicados en el Sistema de Ayuda.

Dentro de los patrones utilizados tenemos los GRASP (General Responsability Asignment Software Patterns) permiten asignar correctamente las responsabilidades a cada una de las clases que intervienen en el modelo. En el sistema se utilizaron cuatro de los cinco patrones GRASP fundamentales:

- **Experto:** Se asignaron responsabilidades a las clases con la información necesaria para cumplirla, lo cual permitió que se conservara el encapsulamiento en las clases del sistema

debido a que los objetos lograron valerse de su propia información para hacer lo se les pedía, esto favoreció el logro del diseño de un sistema más robusto y fácil de mantener .

- **Creador:** Se asignaron responsabilidades a las clases de crear instancias de otras conociendo que las primeras son las que contienen la información para ello lográndose un bajo acoplamiento lo cual se tradujo en menos dependencias respecto al mantenimiento así como en mejores oportunidades de reutilización en el sistema.
- **Alta cohesión:** Se asignaron responsabilidades a las clases de manera que todos sus métodos tuvieran un comportamiento bien definido logrando mantener la complejidad dentro de límites manejables obteniéndose de esta manera clases más fáciles de entender, cambiar y reutilizar.
- **Bajo acoplamiento:** Cada clase está acoplada a las clases estrictamente necesarias por lo que si ocurriera algún cambio en alguna parte del sistema no se afectarían el resto de los componentes, además se pueden entender las clases por separado.

De los patrones GoF que se aplicaron en el diseño de clases del sistema de ayuda en específico están los siguientes:

Facade: Proporciona un diseño adecuado que nos llevará a una factorización en las clases que son estrictamente necesarias para el sistema donde se esta aplicando. En ese caso, es posible construir una clase facade (fachada), que tenga la interfaz y sea la que realmente se comunique con la estructura de las clases reales cuyas interfaces pueden variar. La clase fachada se interpone entre el cliente y el subsistema que proporciona la funcionalidad deseada, en resumen fachada simplifica la interfaz del subsistema.

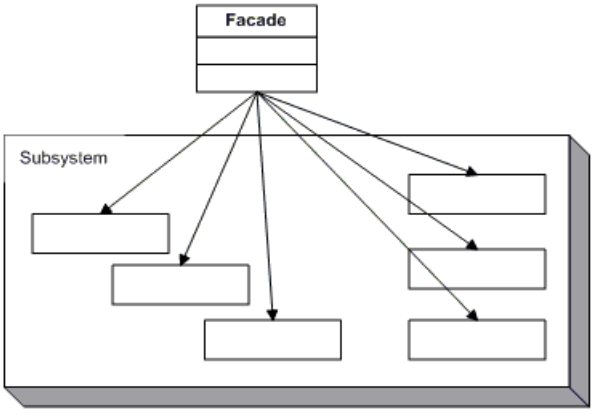


Figura 3.1 Patrón Facade.

Singleton: Este patrón asegura que una clase tendrá solo una instancia, y provee un mundo global de acceso a la misma. El constructor es privado y el método instance() es el que devuelve la única instancia de esta clase o la crea si no existe.

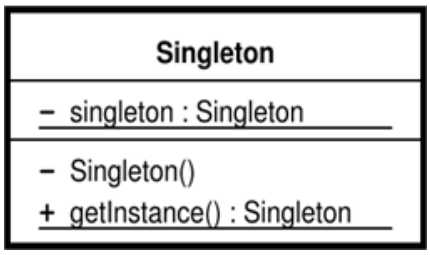


Figura 3.2 Patrón Singleton

El siguiente diagrama de clases del diseño es un ejemplo de cómo son aplicados los patrones Singleton y Facade en el caso del primero asegurándose de que la clase tenga una sola instancia y a la vez proporcionando un punto de acceso global a la misma, en caso del Facade o Fachada (como lo denominamos en la figura) este es utilizado como punto de entrada para que accedan a la base de datos los componentes redireccionando las peticiones al objeto correspondiente.

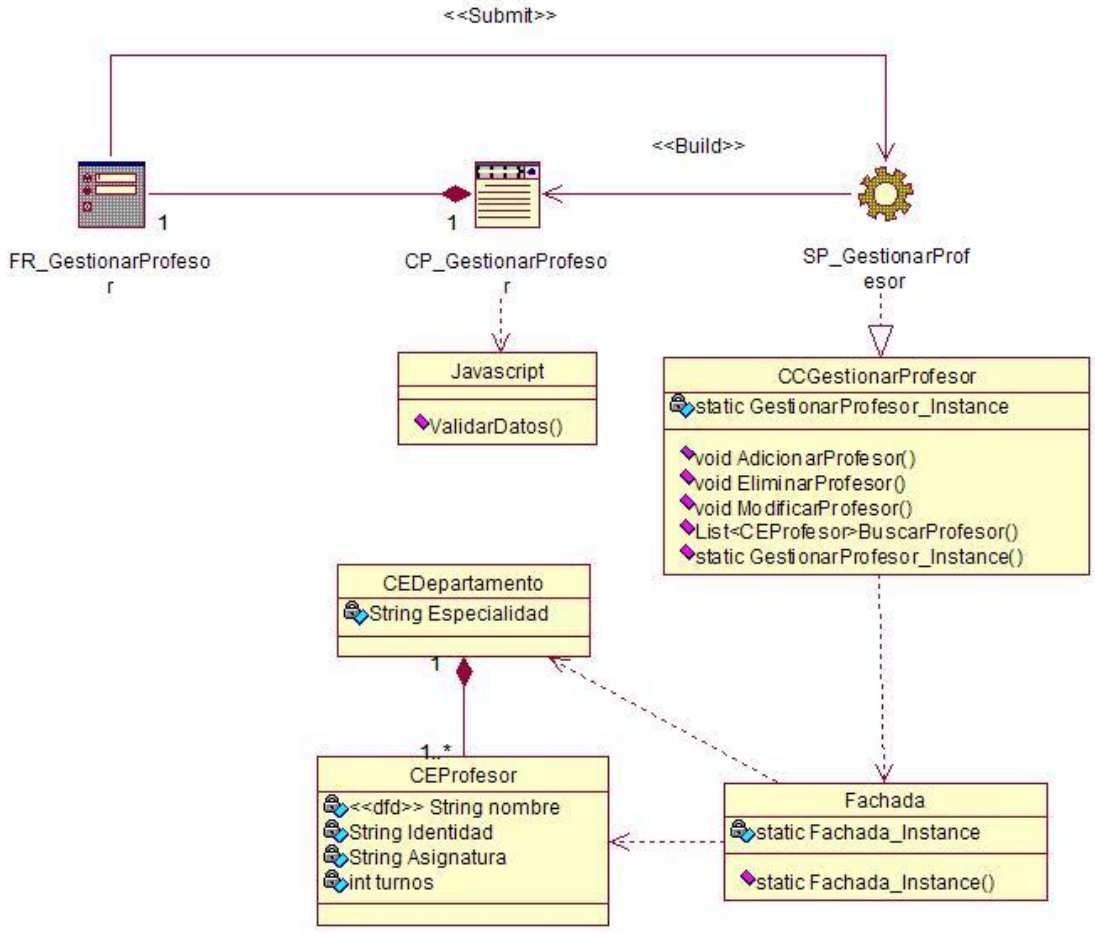


Figura 3.3 Ejemplo de DCD.

El patrón tres capas se utiliza en la arquitectura de tres capas ya que ofrece entre sus principales ventajas las que se listan a continuación:

- Aísla la lógica de la aplicación y la convierte en una capa intermedia bien definida y lógica del software.
- En la capa de presentación se realiza relativamente poco procesamiento de la aplicación.
- Simplifica la comprensión y organización del desarrollo del sistema, reduciendo las dependencias de forma que las capas más bajas no son conscientes de ningún detalle de las superiores.
- Facilita la reutilización.(LARMAN, 2003)

3.7. Capas utilizadas en el Sistema.

Capa de presentación o de interfaz de usuario:

Esta formada por los formularios .HTML y sus controles o atributos, las client page y las server page. Está definida como la capa con la que interactúa el usuario.

Capa de negocio:

Está formada por las entidades que se definen en el sistema, que representan objetos que van a ser manejados o consumidos por toda la aplicación y por las clases .php que se encuentran dentro del negocio de la aplicación.

Capa de acceso a datos:

Contiene clases que interactúan con la base de datos, estas clases altamente especializadas permiten, utilizando los procedimientos almacenados generados, realizar todas las operaciones con la base de datos de forma transparente para la capa de negocio. Están definidas por las clases DAO del sistema.

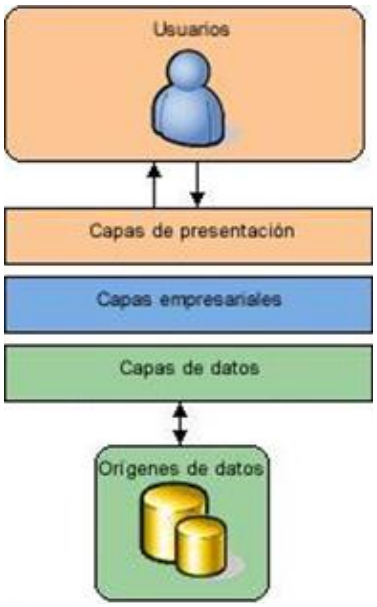
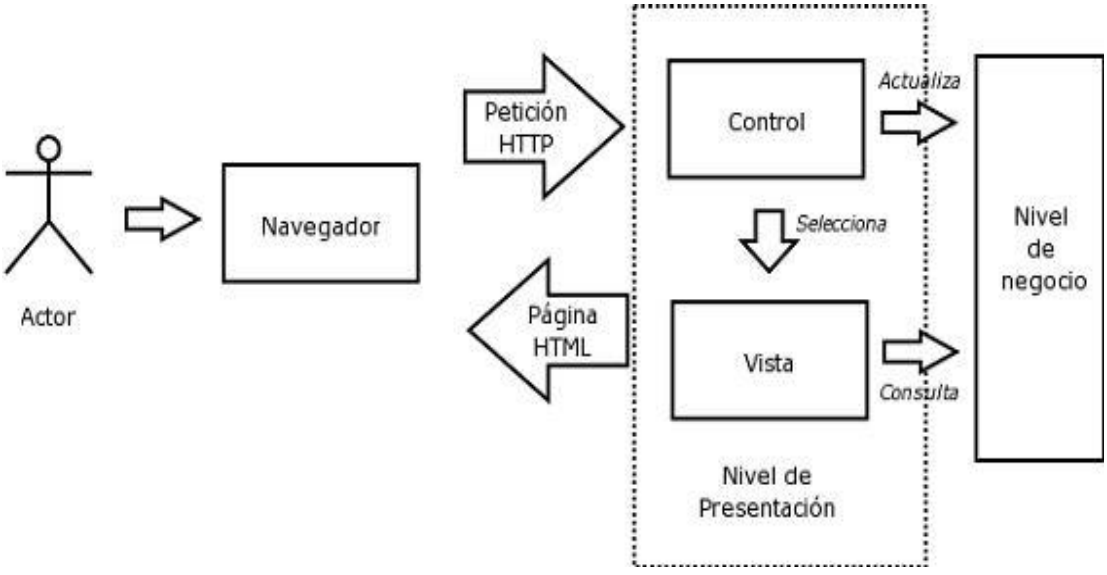


Figura 3.4 Organización de las capas en el sistema

3.8. Descripción de la arquitectura propuesta.

Debido en gran manera, a que la estrategia tradicional de utilizar aplicaciones compactas causan gran cantidad de problemas de integración en sistemas de software complejos, es que se propone para nuestro sistema utilizar una arquitectura dividida en capas para eliminar importantes problemas de escalabilidad, disponibilidad, seguridad e integración entre otros. Y este es el caso de una arquitectura MVC (Model/View/Controller), donde las donde las piezas de un programa se pueden construir por separado y luego unirlas en tiempo de ejecución. Si uno de los componentes comienza a funcionar mal, puede reemplazarse sin que las otras piezas se vean afectadas y como el Modelo, las Vistas y los Controladores se tratan como entidades separadas, esto hace que cualquier cambio producido en el Modelo se refleje automáticamente en cada una de las Vistas. Unido a esto, incorpora ventajas importantes tales como:

- Al separar de manera clara la lógica de negocio (modelo) de la vista permite la reusabilidad del modelo, de modo que la misma implementación de la lógica de negocio que maneja una aplicación pueda ser usado en otras aplicaciones, sean éstas web o no.
- Permite una sencilla división de roles, dejando que sean diseñadores gráficos sin conocimientos de programación o desarrollo de aplicaciones los que se encarguen de la realización de la capa vista, sin necesidad de mezclar código php entre el código visual que desarrollen.

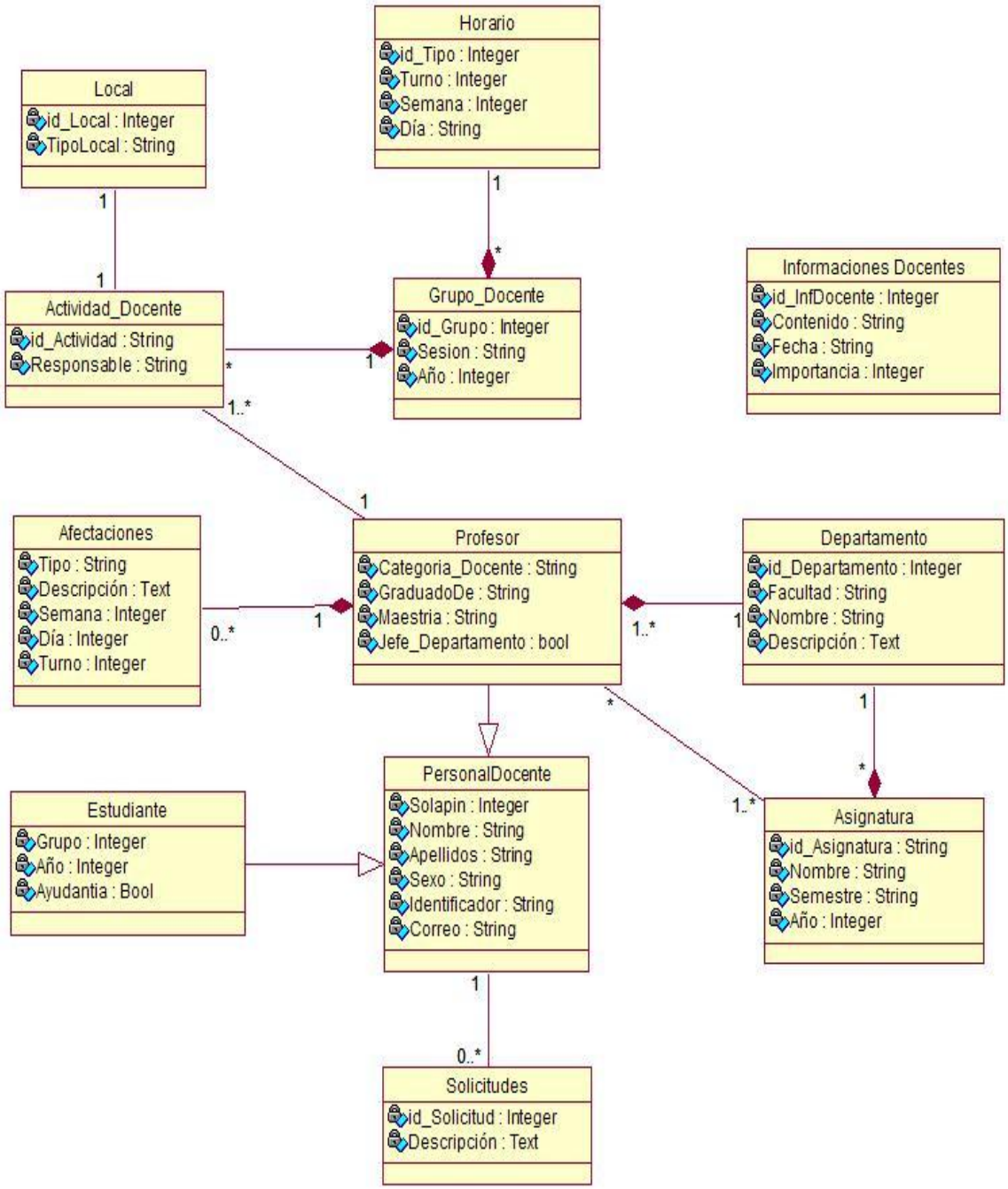


El diseño debe ser específico al problema que se tiene entre manos, pero suficientemente general para adaptarse a problemas y requerimientos futuros. Los diseñadores experimentados en OO dicen que un diseño reusable y flexible es difícil, si no imposible de obtener bien, la primera vez. Antes de

que un diseño sea terminado, usualmente tratan de reutilizarlo varias veces, modificándolo cada vez. (GAMMA, 1995).

3.9. Diagrama de clases Persistente.

Una clase persistente es una clase entidad que tiene la capacidad de mantener su valor en el espacio y en el tiempo, y el diagrama de clases persistentes no es más que dichas clases y las relaciones que se establece entre ellas (asociación, agregación/composición).



3.10. Conclusiones Parciales.

En el presente capítulo tomando como base las características que se habían definido en el capítulo anterior se construyeron los diagramas de clases del análisis y del diseño así como los diagramas de interacción y el diagrama de clases persistentes generándose los artefactos definidos en el flujo de trabajo que propone RUP. Se hace uso de patrones de diseño los cuales mejoran considerablemente la calidad del software y agilizan en gran medida el trabajo de los programadores, con lo que finalmente quedaron sentadas las bases para seguir trabajando en la futura implementación del sistema propuesto.

Conclusiones.

Luego de la investigación y estudio realizado se arribaron a las siguientes conclusiones:

- Se estableció la metodología de desarrollo, tecnologías y herramientas necesarias para realizar el análisis, diseño y posterior desarrollo del sistema.
- Se definieron y describieron los servicios y funcionalidades que debe tener el sistema, teniendo en cuenta las características intrínsecas de estos procesos y peculiaridades de nuestra universidad.
- Se definieron requisitos funcionales y no funcionales ilustrándose mediante los diferentes modelos y diagramas de análisis y diseño la estructura final que tendrá la solución propuesta.
- Se obtuvieron los artefactos necesarios, según la metodología de desarrollo de software seleccionada (RUP), para implementar la propuesta de solución.
- Se realizó un el análisis y diseño de un sistema informático para la planificación docente en la Facultad 3, que permitirá el posterior desarrollo por los programadores.

Recomendaciones.

Al concluir la presente tesis se realizan una serie de recomendaciones que podrán tenerse en cuenta para el desarrollo futuro del sistema:

- Proseguir con el estudio realizado con el propósito de añadir nuevas funcionalidades al sistema.
- Incorporar funcionalidades que involucren técnicas de Inteligencia Artificial y generación automática de horarios.
- Se propone recoger las diversas opiniones de estudiantes y profesores con el objetivo de enriquecer y perfeccionar la aplicación.

Bibliografía.

Ivar Jacobson, G. B. (2000). *El Proceso Unificado de Desarrollo de Software.* Madrid: Addison Wesley.

Molina, M. O. (1995). *Higiene de la actividad docente.* La Habana: Editorial Pueblo y Educación.

Pressman, Roger. (2005). *Ingeniería del software: Un enfoque práctico*. Félix Varela.

LARMAN, C. (2003). UML y Patrones. Introducción al análisis y diseño orientado a objetos. Prentice Hall.

GAMMA, E. (1995). Design Pattern. Addison-Wesley

Övergaard, Gunnar y Palmkvist, Karin: *Use Cases Patterns and Blueprints*, Addison Wesley Professional, 2004.

SCHMULLER, J. *Aprendiendo UML en 24 horas*. Mexico, 2000.

Dra. Lourdes García Ávila Ing. Leidy Fernández Sánchez. *Procedimiento para el desarrollo del proceso de ingeniería de requisitos en un proyecto software* (PROCIR). Departamento de Sistemas y Computación. Instituto Tecnológico de Morelia.

Baker, F.T., *Structured programming in a production programming environment.* 1975: ACM Press.

Jackson, M.A., *Principles of Program Design.* 1975: Academic Press.

Buschmann, F., *Pattern-Oriented Software Architecture.* 1996: John Wiley & Sons.

Davis, A., *Principles of Software Development.* 1995: McGraw-Hill.

Booch, G., *Object-Oriented Design with Applications. The Benjamin/Cummings Publishing Company.* Vol. 1. 1991.

Albin, S., The *Art of Software Architecture: Design methods and techniques.* 2003, Nueva York: Adison Wiley.

Perry, D., *Software Architecture and its relevance for Software Engineering.* 1997: Coord.

Rising, L., *Pattern Almanac 2000.* 2000: Addison-Wesley.

Referencias Bibliográficas.

Molina, M. O. (1995). *Higiene de la actividad docente.* La Habana: Editorial Pueblo y Educación.

Ivar Jacobson, G. B. (2000). *El Proceso Unificado de Desarrollo de Software.* Madrid: Addison Wesley.

Pressman, Roger. (2005). Ingeniería del software: Un enfoque práctico. Félix Varela.

LARMAN, C. (2003). UML y Patrones. Introducción al análisis y diseño orientado a objetos. Prentice Hall.

GAMMA, E. (1995). Design Pattern. Addison-Wesley

www.w3schools.com. (s.f.). Recuperado el 4 de 5 de 2008, de www.w3schools.com: http://www.w3schools.com/dhtml/dhtml_intro.asp

www.w3c.es. (s.f.). Recuperado el 4 de 5 de 2008, de www.w3c.es: http://www.w3c.es/divulgacion/guiasbreves/HojasEstilo

Garrett, J. J. (18 de Febrero de 2005). *www.adaptivepath.com*. Recuperado el 4 de 5 de 2008, de www.adaptivepath.com: http://www.adaptivepath.com/ideas/essays/archives/000385.php

www.netcraft.com. (s.f.). Recuperado el 7 de 5 de 2008, de www.netcraft.com.

Ricardo J. Vargas Del Valle, J. P. (s.f.). *www.di-mare.com.* Recuperado el 7 de 5 de 2008, de www.di-mare.com: http://www.di-mare.com/adolfo/cursos/2007-2/pp-3capas.pdf

Lago, R. (Abril de 2007). *www.proactiva-calidad.com.* Recuperado el 8 de 5 de 2008, de www.proactiva-calidad.com: http://www.proactiva-calidad.com/java/patrones/mvc

Gracia, J. (27 de Mayo de 2005). *http://www.ingenierosoftware.com.* Recuperado el 12 de 5 de 2008, de http://www.ingenierosoftware.com: http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php

Glosario de Términos.

A

Artefacto de software: Cualquier cosa que resulte del proceso de desarrollo de software; por ejemplo: especificaciones, diseños, software, etc.

AD: Término usado para referirse a las Actividades Docentes en un horario determinado.

C

CU: Caso de Uso

CUN: Cado de Uso del Negocio

CUS: Caso de Uso del Sistema

D

DAO: Se referiere a Data Access Object, Diseño Asistido por Ordenadores.

DCS: Diagrama de Casos de Uso del Sistema.

DCD: Diagrama de Clases de Diseño.

E

Enlace, vínculo, hipervínculo: Un hiperenlace es un URL publicado dentro de una página web. Los hipervínculos o enlaces normalmente aparecen en azul y si se les pincha le llevan al usuario a otra página.

EF: Educación Física (asignatura).

H

Herramientas CASE: Grupo de herramientas utilizadas para el desarrollo de de todo tipo de proyectos de ingeniería de software.

HTTP: Hypertext Transfer Protocol. Protocolo de nivel de aplicación usado extensivamente en Internet para el acceso a documentos.

HTML: Hypertext Markup Language. Lenguage de tags estandarizado para la creación de documentos para la web.

R

RUP: Metodología para el desarrollo de software, que en español sería Proceso Unificado de Desarrollo.

Roles: Para la administración y habilitar distintas funcionalidades a los usuarios surgen los roles, como Jefe Departamento, Planificador, Personal Docente, Vice-decano docente.

T

TIC: Tecnologías de la Información y de las Comunicaciones.

U

UCI: Universidad de las Ciencias Informáticas.

UML: Unified Modeling Language. Lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software.

URL: las siglas URL en inglés quieren decir "Uniform Resource Locator," y se refiere al texto que identifica a una página web. Normalmente empieza por "http:/...".

S

Software: Sistemas o Aplicaciones expresadas en un lenguaje de máquina.

SGBD: Sistema de gestión de Base de Datos, es básicamente el software que permite la utilización y la actualización de los datos almacenados en una o varias bases de datos por los usuarios.

P

PHP: Hypertext Preprocessor. Ambiente script del lado del servidor que permite crear y ejecutar aplicaciones Web dinámicas e interactivas, con PHP se pueden combinar páginas HTML y scripts con el objetivo de crear aplicaciones más potentes.

Personal Docente: En este documento cuando se utiliza este término se refiere a estudiantes y profesores, ambos.

X

XP: eXtreme Programming. Metodología de desarrollo de software basada en valores como simplicidad, comunicación, retroalimentación.