UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA

PROYECTO DE GRADO
"SISTEMA DE PLANIFICACION Y SEGUIMIENTO ACADÉMICO CASO: UNIDAD EDUCATIVA RVDO. P. WALTER STRUB"

PARA OPTAR AL TITULO DE LICENCIATURA EN INFORMÁTICA
MENCIÓN: INGENIERÍA DE SISTEMAS INFORMÁTICOS

POSTULANTE: WILMER DANIEL LEONARDINI APARICIO
TUTOR: Lic. ROBERTO VARGAS BLACUTT
REVISOR: Lic. RENÉ CASILLA GUTIÉRREZ MSc.

LA PAZ – BOLIVIA
2009

UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA

LA CARRERA DE INFORMÁTICA DE LA FACULTAD DE CIENCIAS PURAS Y NATURALES PERTENECIENTE A LA UNIVERSIDAD MAYOR DE SAN ANDRÉS AUTORIZA EL USO DE LA INFORMACIÓN CONTENIDA EN ESTE DOCUMENTO SI LOS PROPÓSITOS SON ESTRICTAMENTE ACADÉMICOS.

LICENCIA DE USO

El usuario está autorizado a:

a) visualizar el documento mediante el uso de un ordenador o dispositivo móvil.
b) copiar, almacenar o imprimir si ha de ser de uso exclusivamente personal y privado.
c) copiar textualmente parte(s) de su contenido mencionando la fuente y/o haciendo la referencia correspondiente respetando normas de redacción e investigación.

El usuario no puede publicar, distribuir o realizar emisión o exhibición alguna de este material, sin la autorización correspondiente.

DEDICATORIA

A mi madre por su cariño y comprensión incondicional que me dio a lo largo de todos estos años de estudio y a lo largo de mi vida.

A mis hermanas por su comprensión y apoyo brindado, y por todas las experiencias que pasamos juntos.

AGRADECIMIENTOS

Al Licenciado Roberto Vargas Blacutt, docente Tutor, por sus acertadas observaciones, recomendaciones y exigencias realizadas, que ayudaron a concluir con el proyecto.

Al Licenciado René Casilla Gutiérrez, docente Revisor, por su paciencia en todas las revisiones y por las valiosas sugerencias que me guiaron a lo largo del desarrollo del proyecto.

Al Licenciado Franz Cuevas Quiroz, por sus contribuciones y sugerencias realizadas al inicio del proyecto.

Al Director de la Unidad Educativa Rvdo. P. Walter Strub, Profesor Juan Carlos Torrico, por su colaboración y la confianza depositada en mi persona.

A todo el personal de la Unidad Educativa Rvdo. P. Walter Strub, por la colaboración y tiempo brindado en el desarrollo del proyecto.

A todas aquellas personas que de una u otra forma, colaboraron o participaron en mi formación como persona y profesional, un sincero agradecimiento

RESUMEN

El aumento de Instituciones Educativas y el crecimiento de la población estudiantil dentro de dichos centros, genera un gran volumen de información, que debe ser organizada y clasificada para su utilización. Para realizar estas tareas muchos de los centros educativos hacen uso de las computadoras y tecnología informática actual, para coadyuvar con el trabajo.

La Unidad Educativa Rvdo. P. Walter Strub es una institución con crecimiento de población estudiantil, donde se tiene un gran volumen de información, la forma de almacenamiento y organización de esta producía retardo en los procesos realizados por la institución, es así que se vio necesario desarrollar el presente proyecto para organizar y clasificar de mejor manera la información académica generada por el colegio.

El proyecto ha sido desarrollado con conceptos de programación Orientada a Objetos, utilizando el Lenguaje de Modelado Unificado (UML) para modelar el sistema, siguiendo el método de Análisis y Diseño sugerido por Pressman, que se basa en el método de Diseño Dirigido por Responsabilidades (DDR) de Wirf-Brock. Para el diseño de la parte navegacional de la aplicación se apoyo con la metodología OOHDM. La implementación del sistema se la realizo con el lenguaje PHP que es orientado a la web, junto con el gestor de base de datos Postgres y como apoyo para la interfaz de usuario Java script que se ejecuta en el lado del cliente.

Dentro el Sistema de Planificación y Seguimiento Académico fueron implementados los módulos de kardex estudiantil, inscripción de estudiantes, kardex del personal, planificador de horarios, actualización de calificaciones, control de asistencia de personal y estudiantes. La implementación de estos módulos le brinda a la Unidad Educativa un mejor acceso, organización y control de la información académica y la posibilidad de obtener informes o reportes sobre: horarios escolares por curso y por profesor, emisión de boletines de calificaciones, informe sobre asistencia del personal, listados de estudiantes entre otros.

INDICE

Capitulo 1 MARCO REFERENCIAL **Pag**

1.1 Introducción 1
1.2 Antecedentes 2
1.2.1 Antecedentes de la Institución 2
1.2.2 Trabajos Similares 4
1.3 Planteamiento del Problema 5
1.3.1 Análisis del Problema 5
1.3.2 Problema Principal 6
1.4 Objetivos 6
1.4.1. Objetivo General 6
1.4.2. Objetivos Específicos 6
1.5 Justificación 7
1.6 Alcances 7

Capitulo 2 MARCO TEÓRICO

2.1 Marco Conceptual 9
2.2 Proceso de Desarrollo de Software 12
2.2.1 Modelo Evolutivo de Proceso de Software 13
2.2.2 Ingeniería de Requisitos 14
2.3 Análisis 16
2.3.1 El Paradigma Orientado a Objetos 16
2.3.2 Lenguaje de Modelado Unificado 18
2.3.3 Proceso de Análisis Orientado a Objetos 18
2.3.4 Especificación de Casos de Uso 19
2.3.5 Modelado de Clases Responsabilidad Colaboracion (CRC) 22
2.3.6 Definición de Estructuras y Jerarquías 23
2.3.7 Modelo Objeto Relación 23
2.3.8 Modelo Objeto Comportamiento 24

2.4 Modelado de Datos 27
2.5 Diseño 28
2.5.1 Proceso de diseño del Sistema 29
2.5.2 Proceso de diseño de Objetos 30
2.6 Metodología OOHDM 31
2.6.1 Modelo Conceptual 32
2.6.2 Diseño Navegacional 33
2.6.3 Diseño de la Interfaz Abstracta 35
2.6.4 Implementación 37
2.7 Pruebas Orientadas a Objetos 37
2.8 Mantenimiento de Software 38
2.9 Métricas Orientadas a la Función 39

Capitulo 3 MARCO APLICATIVO

3.1 Ingeniería de Requisitos 42
3.2 Análisis y Diseño 46
3.2.1 Casos de Uso 46
3.2.2 Tarjetas CRC (Clases Responsabilidad Colaboracion) 51
3.2.3 Modelo Objeto-Relacion 53
3.2.4 Modelo Objeto-Comportamiento 54
3.2.4.1 Diagrama de Secuencia 54
3.2.4.2 Diagrama de Colaboracion 57
3.2.5 Modelo Entidad Relacion 59
3.2.6 Diseño del Sistema 60
3.2.7 Diseño de Objetos 61
3.2.8 Diagrama de Contexto Navegacional 62
3.2.9 Diseño de la Interfaz Abstracta 67
3.3 Implementación 70
3.4 Pruebas de Integración basadas en el uso 74
3.5 Calidad de Software 76
3.5.1 Funcionalidad 76
3.5.2 Usabilidad 80
3.5.3 Portabilidad 81

Capitulo 4 CONCLUSIONES

4 Conclusiones y Recomendaciones 83
4.1 Conclusiones 83
4.2 Recomendaciones 84

Bibliografía 85

Anexos 87

INDICE DE FIGURAS

Pag.

Figura 1.1 Organigrama de la Institución..... 2
Figura 2.1 Modelo Incremental..... 13
Figura 2.2 Representación de una Clase..... 17
Figura 2.3 Clase con atributos y operaciones..... 17
Figura 2.4 Diagrama de casos de uso de alto nivel..... 20
Figura 2.5 Diagrama de casos de uso detallado..... 20
Figura 2.6 Tarjeta Clase-Responsabilidad-Colaboracion..... 22
Figura 2.7 Relaciones entre Clases..... 24
Figura 2.8 Diagrama de Secuencia..... 25
Figura 2.9 Diagrama de Colaboración..... 27
Figura 2.10 Diagrama Entidad Relación..... 28
Figura 2.11 Transformación de Modelo de AOO al Modelo de DOO..... 29
Figura 2.12 Etapas de la Metodología OOHDM..... 32
Figura 2.13 Esquema conceptual..... 33
Figura 2.14 Diagrama de Contexto Navegacional..... 35
Figura 2.15 Representación de ADV's..... 36
Figura 2.16 Calculo de Puntos de Función..... 40
Figura 3.1 Modelo de casos de uso del Sistema..... 46
Figura 3.2 Diagrama de casos de uso detallado..... 47
Figura 3.3 Relaciones entre clases..... 53
Figura 3.4 Diagrama de secuencia Registra inscripción..... 54
Figura 3.5 Diagrama de secuencia Planifica Horarios..... 55
Figura 3.6 Diagrama de secuencia actualiza calificación..... 56
Figura 3.7 Diagrama de secuencia Registro faltas o atrasos..... 56
Figura 3.8 Diagrama de colaboración Registra Inscripción..... 57
Figura 3.9 Diagrama de colaboración Planifica Horarios..... 57
Figura 3.10 Diagrama de colaboración Actualiza Calificación..... 58
Figura 3.11 Diagrama de colaboración Registro de faltas o atrasos..... 58
Figura 3.12 Diagrama Entidad Relación..... 59
Figura 3.13 Arquitectura Propuesta para el Sistema..... 60
Figura 3.14 Diagrama de Clases Sistema de Planificación y Seguimiento Académico..... 63
Figura 3.15 Contexto Navegacional Secretaria..... 64

Figura 3.16 Contexto Navegacional Docente 65
Figura 3.17 Contexto Navegacional Director 66
Figura 3.18 Contexto Navegacional Regente 67
Figura 3.19 ADV ingreso de usuarios 67
Figura 3.20 ADV Menú Director 68
Figura 3.21 ADV Planificar horario 68
Figura 3.22 ADV Asignar Cursos 69
Figura 3.23 ADV Tiempo Libre 69
Figura 3.24 ADV Calificaciones por curso 70
Figura 3.25 Interfaz de registro de docentes y administrativos 72
Figura 3.26 Interfaz para asignación de docentes a cursos 73
Figura 3.27 Interfaz ingreso de calificaciones Primaria 73

INDICE DE TABLAS

Pag.

Tabla 2.1 Actividades de la Ingeniería de Requisitos..... 14
Tabla 2.2 Descripción de un caso de uso expandido..... 21
Tabla 2.3 Curso Normal de Eventos..... 22
Tabla 2.4 Actividades de la fase del Diseño Navegacional..... 35
Tabla 2.5 Actividades de la fase del Diseño de la Interfaz Abstracta..... 36
Tabla 2.6 Actividades de la fase de Implementación..... 37
Tabla 3.1 Tareas en la obtención de Requisitos..... 42
Tabla 3.2 Listado de Requisitos..... 43
Tabla 3.3 Actores del Sistema..... 44
Tabla 3.4 Descripción del caso de uso Planifica horarios..... 48
Tabla 3.5 Descripción del caso de uso Asigna materias..... 49
Tabla 3.6 Descripción del caso de uso Registra inscripción..... 49
Tabla 3.7 Descripción del caso de uso Registra faltas y atrasos de estudiantes..... 50
Tabla 3.8 Tarjeta CRC para la clase Docente..... 51
Tabla 3.9 Tarjeta CRC para la clase Curso..... 52
Tabla 3.10 Tarjeta CRC para la clase Inscripción..... 52
Tabla 3.11 Tarjeta CRC para la clase Horario..... 52
Tabla 3.12 Caso de Prueba Materias asignadas a un Docente..... 74
Tabla 3.13 Caso de Prueba Docentes disponibles..... 74
Tabla 3.14 Caso de Prueba Inscritos en gestión..... 75
Tabla 3.15 Caso de Prueba anotar falta estudiante..... 75
Tabla 3.16 Calculo de puntos de función no ajustado..... 78
Tabla 3.17 Calculo de ajuste de complejidad..... 79
Tabla 3.18 Rangos para evaluar la usabilidad..... 80
Tabla 3.19 Valoración para el cuestionario de Usabilidad..... 81
Tabla 3.20 Rangos para evaluar la portabilidad..... 82
Tabla 3.21 Valoración para la portabilidad..... 82

1.1 Introducción

Uno de los recursos más importantes para una institución es la información que genera con las actividades que realiza, siendo esta de gran importancia para cumplir con los objetivos trazados por la institución.

Las Instituciones Educativas generan información académica de sus estudiantes año tras año, en este sentido se ve la necesidad de organizar y clasificar la información generada, para lo cual el uso de las computadoras y tecnología actual tiene gran relevancia. Muchos de los procesos para generar esta información son repetitivos ocasionando perdida de tiempo si se los realiza de forma manual.

En este sentido desarrollar un sistema que ayude a automatizar los procesos repetitivos y que también administre y organice de mejor manera la información generada por una Institución Educativa, es el principal motivo del presente proyecto. El desarrollo del proyecto pretende ser una herramienta útil para una Unidad Educativa, ayudando en tareas como la inscripción de estudiantes, emisión de boletines de calificaciones y una mejor planificación de horarios.

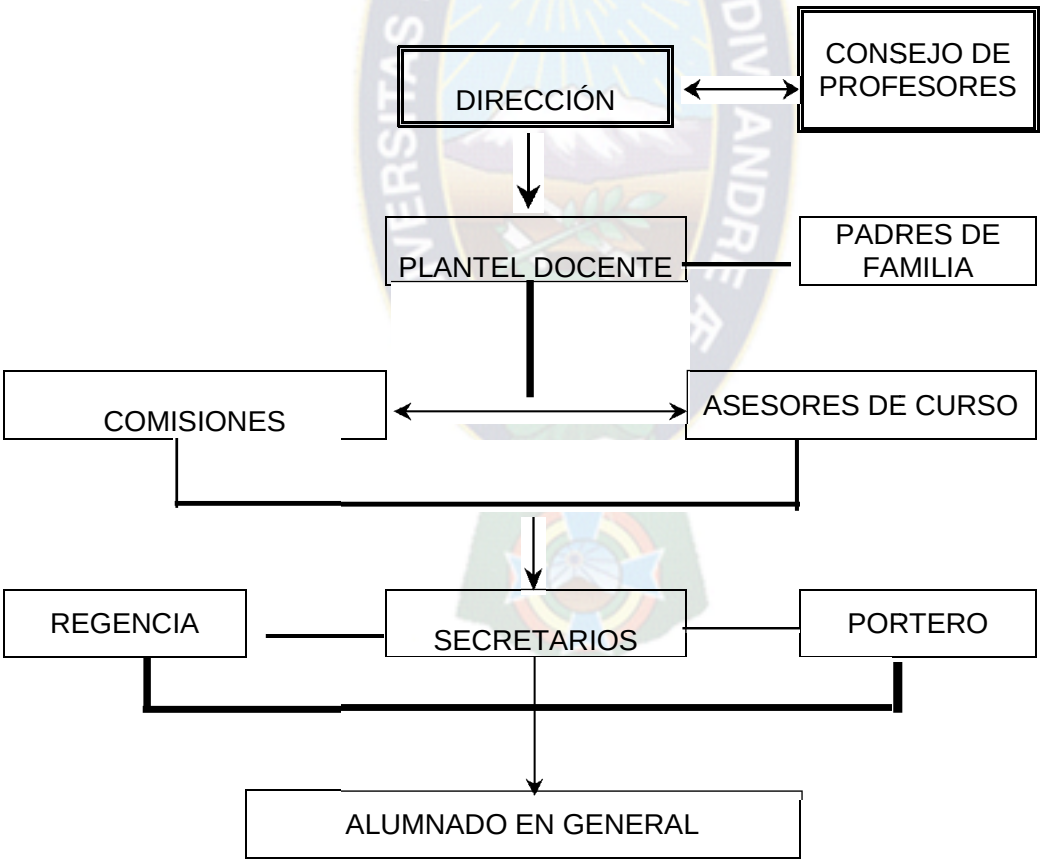
La Unidad Educativa Rvdo. Padre Walter Strub es una institución educativa de la zona sur de La Paz, en la cual la información académica generada no se encuentra organizada de una manera óptima, para la cual se implementara el sistema propuesto, para el desarrollo del mismo se empleara el método de Análisis y Diseño Orientado a Objetos recomendado por [Pressman, 2003], que se basa en el método DDR (Diseño Dirigido por Responsabilidades) de Wirf-Brock, para complementar el diseño se utilizara la metodología OOHDM, con la cual se modelará la parte navegacional del sistema.

1.2 Antecedentes

1.2.1 Antecedentes de la Institución

La Unidad Educativa R.P. Walter Strub lleva a cabo sus actividades académicas, en la zona de Cota Cota, con el principal objetivo de formar a estudiantes de primaria y secundaria, para cumplir con sus tareas y tener un buen desempeño y funcionamiento del colegio, se encuentra organizada de acuerdo al siguiente organigrama (Ver Figura 1.1), donde cada cargo tiene una función especifica para realizar de forma correcta los objetivos de la institución.

Figura 1.1 Organigrama de la Institución



Fuente: [Dirección U.E.R.P. Walter Strub, 2009]

La “Unidad Educativa R.P. Walter Strub A”, actualmente desarrolla sus actividades en el turno tarde, con estudiantes de sexto a octavo de Primaria y de primero a cuarto grado de Secundaria, teniendo un total de diecinueve cursos, con un alumnado aproximado de seiscientos cincuenta estudiantes, siendo de vital importancia la información académica del alumnado, para cumplir con los objetivos de la institución, cada miembro tiene una tarea especifica.

Entre las funciones que debe realizar el Director, se puede destacar las siguientes:

- Planificar, organizar, dirigir y supervisar los procesos pedagógicos;
- Supervisar y evaluar el desempeño del personal a su cargo;
- Elaborar un parte mensual de asistencia del personal docente y administrativo.

Los profesores tienen muchas tareas a su cargo entre estas podemos mencionar:

- Elaborar los informes de aprendizaje y llenar las libretas escolares de los alumnos;
- Tendrá que estar asignado a un curso, y ser un consejero del curso, elegido por el Director de la Unidad Educativa y el Consejo de Profesores.

Las tareas de la secretaria que se pueden mencionar son:

- Llenar y centralizar los libros de inscripción y de notas, formularios y kardex de los alumnos y profesores;
- Recibir, registrar, distribuir, archivar y custodiar toda la documentación de la Unidad Educativa.

La parte de Regencia esta encargada principalmente en apoyar al secretario en el cumplimiento de sus funciones, y el control riguroso de asistencia de los alumnos, así como del comportamiento disciplinado de los mismos.

La Unidad Educativa R. P. Walter Strub realiza los procesos académicos y administrativos de forma manual y en algunos casos con la ayuda de Excel y por estos motivos se ve la necesidad de implementar un sistema de Planificación y Seguimiento Académico y que ayude a minimizar los tiempos en determinadas tareas, y también tener la información almacenada de manera mas organizada.

1.2.2 Trabajos Similares

En la actualidad la gestión académica en los colegios del ámbito local y así como del exterior tiene gran importancia para las instituciones educativas, pudiéndose mencionar los siguientes proyectos o Sistemas desarrollados dentro del ámbito académico y con características comunes, algunos de estos desarrollados por estudiantes de la carrera de informática, otros implementados en colegios de países extranjeros, y otros genéricos desarrollados por compañías especializadas que se encuentran en la Red de Internet y que necesitan pagar una licencia para su uso.

- El Colegio Americano de Guayaquil es una entidad privada sin fines de lucro, no sectaria, bilingüe y bicultural, cuyo interés primordial es cumplir con las necesidades educativas de sus estudiantes. Este colegio Ecuatoriano tiene su sistema académico funcionando en su página web desarrollado en PHP [Guayaquil, 2009];
- DILOGROS es un Software Colombiano especializado en el sector Educativo, que lleva el control de matriculación, elabora listados, permite ver el proceso de calificaciones, elaboración de documentos certificados de notas y diplomas , permite establecer parámetros generales de la institución como ser numero de cursos, aulas, etc. Para usar este software se debe pagar licencia [Dilogros,2009];
- "Sistema de Gestión Académica para la (Unidad Educativa Piloto Intervida) UEPI" que fue desarrollado como proyecto de grado por Edgar Vircochea, utilizando metodología RUP y UML, este sistema realiza la evaluación socioeconómica para los estudiantes que postulan a la Unidad Educativa [Viracochea, 2006];
- "Gestión de Información Académica y administrativa del Colegio Particular Luz a las Naciones" este Sistema fue realizado con metodologías OOHDM y Web Site QEM, y desarrollado por Guido Cutipa, dicho sistema realiza un control de el ingreso monetario por concepto de pensiones de los estudiantes [Cutipa, 2006].

Viendo los sistemas desarrollados, se puede observar la necesidad de implementar una parte de la planificación de horarios, y la parte de Regencia que es la encargada de realizar el control de asistencia de los estudiantes.

1.3 Planteamiento del Problema

1.3.1 Análisis del Problema

La Unidad Educativa Rvdo. Padre Walter Strub, tiene como misión formar a los alumnos pertenecientes a la institución, en este proceso realiza algunas de sus actividades todavía de forma manual provocando perdida de tiempo, tiende a no tener un control pleno sobre algunas de las tareas que realiza, al observar las actividades que el colegio desarrolla se observaron los siguientes problemas (Se pueden ver los problemas encontrados gráficamente en el Anexo A):

- La elaboración de horarios se realiza manualmente, tomando un tiempo de dos semanas en el mejor de los casos**,** esto debido a que se debe tomar en cuenta ciertos parámetros de decisión, como ser la disponibilidad de tiempo de los diferentes docentes, puesto que muchos de ellos trabajan en otros colegios, y tienen cierta cantidad de horas repartidas en los establecimientos a los que pertenecen, también se debe tomar en cuenta que dos docentes no pueden estar asignados a un mismo curso en la misma hora o viceversa, que un docente este asignado a dos cursos en la misma hora, es importante tomar en cuenta todos los factores mencionados, para así poder acomodar a todos los profesores en las diferentes aulas sin tener mayores inconvenientes;
- La entrega de notas por parte de los docentes se hace de forma manual mediante el llenado de una planilla, ocasionando perdida de tiempo en la transcripción;
- Los procesos de emisión de boletines de calificación demoran al no tener un centralizador de notas, actualmente se los elabora copiando materia por materia para cada alumno;
- No se tiene un historial por alumno a la mano;
- El control de asistencia de los estudiantes por parte de los regentes se realiza de manera manual, por lo cual no se tiene un historial eficiente al final de una gestión;
- La elaboración de planillas de asistencia del personal docente y administrativo se realiza de forma manual, demorando bastante tiempo al localizar los horarios de los docentes.

1.3.2 Problema Principal

Tomando en cuenta las observaciones que se detallaron anteriormente, se propone:

¿El Sistema de Planificación y Seguimiento Académico, para la U.E.R.P. Walter Strub permitirá automatizar los procesos manuales de la parte académica, así como una mejor asignación y organización de los horarios para docentes y alumnado?

1.4 Objetivos

1.4.1. Objetivo General

Implementar un sistema que ayude a la Planificación y Seguimiento de los procesos Académicos dentro de la U.E. R.P. Walter Strub, de tal forma que minimice los tiempos que se toman al realizarlos de forma manual, teniendo información accesible y completa en todo momento.

1.4.2. Objetivos Específicos

- Diseñar una interfaz web amigable mediante la cual los usuarios puedan acceder a la información académica;
- Implementar mecanismos de seguridad para que solo usuarios autorizados tengan acceso a datos y programas.
- Implementar la consulta del historial académico por alumno, teniendo información actual en el momento que se requiera;
- Implementar un kardex por alumno y también del personal docente y administrativo que trabaja en la U.E. R.P. Walter Strub;
- Agilizar los procesos de inscripción y generación de boletines de calificaciones;
- Ayudar en el proceso de elaboración de horarios por docente y por curso, ayudando al Director Académico en la toma de decisiones;
- Implementar un modulo de control de asistencia para los docentes y alumnos.

Los objetivos a desarrollar en el presente proyecto se pueden ver gráficamente en el Anexo B.

1.5 Justificación

La responsabilidad de toda institución es tener toda la información que respecta a la misma de manera ordenada y organizada de tal manera que sea fácil encontrar ciertos requerimientos cuando sea necesario, el automatizar dichos procesos para la Unidad Educativa R.P. Walter Strub, eliminara la información redundante que se genera al realizar los procesos manualmente.

En lo Económico la implementación del sistema no incurrirá en gastos adicionales para la institución, puesto que se utilizara herramientas de desarrollo con licencia gratuita y por otra parte la institución cuenta con los equipos necesarios para implementar el sistema.

El desarrollo del presente proyecto tiene como función social ayudar al personal docente y administrativo con las tareas cotidianas que realizan, reduciendo la probabilidad de errores que pueden presentarse haciéndolas manualmente, minimizar los tiempos de los procesos mecánicos y repetitivos que se realizan, teniendo de esta manera la información almacenada de manera mas ordenada.

Desde el punto de vista tecnológico, se empleara herramientas actuales para el desarrollo, como ser PHP 5 y Postgres, las cuales permiten el acceso y manipulación de la información, de forma fácil y segura, siendo así una innovación tecnológica para la Institución Educativa automatizando sus procesos que realizan actualmente.

1.6 Alcances

El proyecto propuesto contempla los siguientes alcances:

- Un seguimiento académico a cada estudiante en los diferentes cursos del establecimiento, como ser, inscripción de estudiantes, control de asistencia, registro y control de calificaciones ;
- Asignación de cursos a los docentes, controlando la doble asignación o choque de horarios que pueda existir.
- Se implementara una Interfaz Web amigable, con la cual los usuarios se familiaricen de forma natural;

- Los usuarios autorizados que deseen consultar información académica tendrán que identificarse con un nombre de usuario y contraseña, para ingresar al sistema. La actualización de información, será realizada por los usuarios autorizados a una respectiva área;

- Elaboración de reportes, para cada área utilizando la información que corresponda a la misma, como ser, nominas de alumnado, listados de docentes, horarios por curso, horarios por profesor, emisión de libretas de calificaciones entre otros;

Al desarrollar este sistema se realizaran los módulos mencionados con anterioridad de manera general para el uso de cualquier unidad educativa y con limitaciones propias de las herramientas que se pretenden utilizar. Los cuales son PHP como lenguaje de programación, Postgres para gestionar la base de datos y Javascript para interactuar con el usuario, todas estas son de carácter gratuito.

2.1 Marco Conceptual

En este apartado se darán a conocer los conceptos y definiciones relevantes con respecto al desarrollo del proyecto, las mismas que llevaran a la realización del mismo.

Aplicaciones Web

Una aplicación basada en web (WebApps), es muy diferente a un software tradicional, puesto que debe tomar en cuenta muchos atributos que en los sistemas convencionales se pueden pasar por alto o no son de gran importancia. Las antiguas aplicaciones distribuidas en cd's dieron lugar a aplicaciones dinámicas, de constante actualización e incluso personalizables, capaces de adaptarse a los tipos de usuarios y en casos avanzados, a cada usuario en particular. Estas características encuentran el medio ideal en la *web*, ya que de otra forma sería costoso su mantenimiento y evolución.

El desarrollo de aplicaciones web involucra decisiones no triviales de diseño e implementación que inevitablemente influyen en todo el proceso de desarrollo, afectando la división de tareas. Los problemas involucrados, como el diseño del modelo del dominio y la construcción de la interfaz de usuario, tienen requerimientos disjuntos que deben ser tratados por separado. [Silva, 2002].

Los sistemas basados en Web implican una mezcla de publicación impresa y desarrollo de software, de marketing e informática, de comunicaciones internas y relaciones externas, y de arte y tecnología. Reside en una red y debe dar servicio a las necesidades de muchos clientes. Una WebApp puede residir en Internet (haciendo posible su acceso desde cualquier lugar del mundo). De forma alternativa, una aplicación se puede ubicar en una Intranet (implementado en la red interna de una organización) o una Extranet (comunicación entre redes) [Pressman, 2003]. Las Aplicaciones Web pueden tener contenido hipermedial, textos, gráficos, animaciones, etc. A diferencia de una aplicación tradicional, una Web App se encuentra en constante evolución.

Dado que las WebApps están disponibles a través del acceso por red, es difícil, si no imposible, limitar la población de usuarios finales que pueden acceder a la aplicación. Con

objeto de proteger el contenido confidencial y de proporcionar formas seguras de transmisión de datos, deberán implementarse fuertes medidas de seguridad en toda la infraestructura que apoya una WebApp y dentro de la misma aplicación [Pressman, 2003].

Tecnologías Web

El desarrollo de Aplicaciones web se debe basar en estándares de Internet, en la actualidad el estándar para construcción de WebApps, es HTML que posibilita la inserción de texto, gráficos, sonido, video, etc. El metalenguaje HTML es estático, por lo que es necesario utilizar lenguajes de programación web que sirvan de ayuda para poder interactuar con contenido dinámico. Entre estos podemos encontrar a JavaScript que se ejecuta del lado del cliente, PHP que es un lenguaje del lado del servidor, el cual es capaz de manipular múltiples bases de datos.

A lo anteriormente mencionado se puede agregar que una Web App necesita de un gestor de base de datos, para poder almacenar los datos que se requieren manipular, se puede mencionar a Postgres que tiene licencia gratuita. Para poder correr los lenguajes que se usaran en la WebApp es necesario contar con un servidor de peticiones HTTP, Apache es uno de los servidores mas conocidos. Todos estos elementos los veremos a continuación.

Apache

El servidor Web Apache es un programa que corre sobre el servidor que escucha las peticiones HTTP que le llegan y las satisface. Dependiendo del tipo de la petición, el servidor Web buscara una pagina Web o bien ejecutara un programa en el servidor. De cualquier modo, siempre devolverá algún tipo de resultado HTML al cliente o navegador que realizo la petición.

Tener un servidor Web instalado en el servidor es fundamental para el desarrollo de las aplicaciones Web que se vayan a construir, ya que se ejecutaran en el. Apache es uno de los mejores servidores de Webs utilizados en la red Internet desde hace mucho tiempo, únicamente le hace competencia el servidor de Microsoft, el IIS.

Por lo que este servidor es uno de los mayores triunfos del software libre. Es un servidor de Web flexible, gratuito pero profesional, rápido y eficiente, continuamente actualizado y adaptado a los nuevos protocolos [García, 2000].

Lenguaje PHP

Preprocessed Hypertext Pages (PHP), es lenguaje de scripting que permite generar paginas HTML. El lenguaje PHP es multiplataforma y de licencia libre. A diferencia de las paginas estáticas de HTML que son útiles para presentar documentos estáticos, es decir que no son modificables, PHP permite generar un pagina HTML en forma dinámica, por ejemplo como resultado de una consulta a una base de datos., o generar gráficos, o cualquier otra cosa que necesite ser generada en base ciertos datos que pueden cambiar en el tiempo [Morales, 2009].

Una de las grandes limitaciones del HTML es la forma en que trabaja. En particular, si debe presentar al usuario un grupo de opciones para que este elija, uno debe conocer esas opciones de antemano al momento del diseño del formulario.

Con PHP es fácil generar un formulario con las opciones tomadas desde alguna base de datos, de forma que el programador no tiene que modificar su código HTML, ya que este es generado automáticamente mediante un simple script escrito en PHP, que accede a esa base de datos. Entre las muchas bases de datos que se pueden usar tenemos a Postgres, MySQL, Sybase, Oracle, Informix entre otras.

Postgres

Postgres o PostgreSQL, es un sistema de gestión de base de datos relacional orientada a objetos de software libre, ha sido desarrollado de varias formas desde la década de 1980. Entre las principales características podemos mencionar:

- Multiplataforma, puede ser usado bajo cualquier plataforma Unix y Windows;
- Conectividad TCP/IP, JDBC y ODBC;
- Alta concurrencia, mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. para conseguir una mejor respuesta en ambientes de grandes volúmenes;
- Disparadores (*triggers*), Un disparador o *trigger* se define en una acción específica basada en algo ocurrente dentro de la base de datos. En PostgreSQL esto significa la ejecución de un procedimiento almacenado basado en una determinada acción sobre una tabla específica [WikPos, 2009].

JavaScript

JavaScript, es un lenguaje del lado del cliente, se trata de un lenguaje interpretado y los programas escritos con estos lenguajes son conocidos como scripts o guiones. Pese a su nombre no tiene nada que ver con Java, este último es un lenguaje algo más complejo con el que se pueden construir programas de propósito general. La única razón de ser de JavaScript son las páginas web. Con JavaScript no pueden construirse programas independientes, sólo pueden escribirse scripts que funcionarán en el entorno de una página Web, interpretado por un explorador [Rodríguez, 2002].

Con JavaScript es posible manipular las variables de los formularios de forma dinámica, la parte mas interesante que puede destacarse del JavaScript es el manejo de eventos con los que el usuario interactúa, estos pueden ser hacer un clic con el mouse, presionar un botón, etc.

2.2 Proceso de Desarrollo de Software

Un proceso del desarrollo de software es un conjunto de actividades y resultados asociados que producen un producto de software, estas actividades son llevadas acabo por los ingenieros de software. Existen cuatro actividades fundamentales que son comunes para todos los procesos de software, las cuales son [Sommer, 2005]:

- **Especificación de Software,** en esta actividad los clientes e ingenieros definen el software a producir y las restricciones sobre su operación. Es decir, durante la definición, el que desarrolla el software intenta identificar qué información ha de ser procesada, qué función y rendimiento se desea, qué comportamiento del sistema, qué interfaces van a ser establecidas, qué restricciones de diseño existen, y qué criterios de validación se necesitan para definir un sistema correcto [Pressman, 2003].

- **Desarrollo de Software,** es la actividad donde el software se diseña y se programa. Es decir, durante el desarrollo un ingeniero del software intenta definir cómo han de diseñarse las estructuras de datos, cómo ha de implementarse la función dentro de una arquitectura de software, cómo han de implementarse los detalles procedimentales, cómo han de caracterizarse interfaces, cómo ha de traducirse el diseño en un lenguaje de programación y cómo ha de realizarse la prueba [Pressman, 2003].

- **Validación de Software,** en esta etapa el software se valida para asegurar que los requerimientos del cliente hayan sido satisfechos. Se realizan pruebas al sistema acerca de las funciones que el cliente necesita.

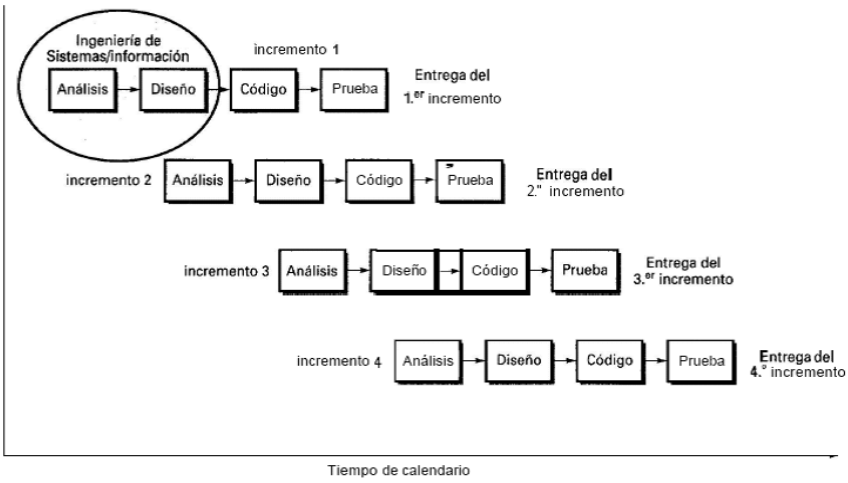
- **Evolución de Software,** en esta actividad se realizan las tareas de corrección, adaptación, mejora y prevención. Las adaptaciones requeridas serán a medida que evoluciona el entorno del software y a cambios debidos a las mejoras producidas por los requisitos cambiantes del cliente.

2.2.1 Modelo Evolutivo de Proceso de Software

Los modelos evolutivos son iterativos. Se caracterizan por la forma en que permiten a los ingenieros del software desarrollar versiones cada vez más completas del software. Existen dos tipos de desarrollo evolutivo, el Desarrollo exploratorio y el de Prototipos desechables, donde el objetivo de este ultimo es el de comprender los requerimientos del cliente y desarrollar prototipos para los requerimientos que no se comprenden con mucha precisión.

El modelo Incremental es un proceso evolutivo es mucho más efectivo que un enfoque cascada, puesto que se adapta a los requerimientos cambiantes del cliente. La Figura 2.1 nos muestra el desarrollo de software desde un enfoque incremental.

Figura 2.1 Modelo Incremental



Fuente: [Pressman, 2003]

El modelo incremental se centra en la entrega de un producto operacional con cada incremento, donde el primer incremento a menudo es un producto esencial. Según [Pressman, 2003], el modelo incremental es útil cuando la dotación de personal no esta disponible para una implementación completa.

2.2.2 Ingeniería de Requisitos

Los Requisitos tienen un papel importante en el desarrollo de cualquier sistema, de ellos depende el fracaso o éxito del mismo. La ingeniería de Requisitos llamada a veces Ingeniería de Requerimientos, trata de estandarizar las actividades que deben realizarse para poder recolectar información, analizarla, documentarla y clasificarla, todo esto para poder definir de forma clara y precisa las funciones que el sistema debe implementar.

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, 2003]. Las Actividades y tareas que se deben realizar en la Ingeniería de Requisitos se muestran en la Tabla 2.1.

Tabla 2.1 Actividades de la Ingeniería de Requisitos

Actividad	Tareas
Obtención de Requisitos	• Realizar entrevistas a usuarios • Observación al entorno • Encuestas a usuarios • Documentar requisitos encontrados
Análisis de Requisitos	• Clasificar requisitos encontrados. • Identificar requisitos imposibles. • Eliminar requisitos inconsistentes
Validación de Requisitos	• Revisión de los requisitos por ingenieros y usuarios. • Aclarar e interpretar términos.

Fuente: Elaboración Propia

Desde un punto de vista conceptual, las actividades que se deben desarrollar en la Ingeniería de Requisitos son de 3 tipos, las cuales detallaremos a continuación.

a) Obtención de requisitos

La identificación de requisitos puede parecer una tarea simple, talves se piense que preguntando a los usuarios sobre lo que desean que el sistema realice sea suficiente, por el contrario la identificación de requisitos es algo mas complicado. La obtención de requisitos puede presentar los siguientes problemas:

- Problemas de alcance, se da cuando el límite del sistema está mal definido, se suele confundir los objetivos del sistema;
- Problemas de comprensión, sucede cuando clientes/usuarios no están completamente seguros de lo que necesitan, a menudo no tienen compresión de las capacidades y limitaciones de una computadora, se reservan cierta información que a su vista puede resultar obvia;
- Problemas de volatilidad, los requisitos cambian con el tiempo.

La Obtención de Requisitos debe realizarse mediante entrevistas a cada usuario, grupos de discusión, realizando encuestas, etc. En toda actividad realizada debe tomarse en cuenta los problemas mencionados anteriormente.

b) Análisis de requisitos

Una vez recopilados los requisitos, el producto obtenido configura la base del análisis de requisitos. Los requisitos se agrupan por categorías y se organizan en subconjuntos, se estudia cada requisito en relación con el resto, se examinan los requisitos en su consistencia, completitud y ambigüedad, y se clasifican en base a las necesidades de los clientes/usuarios [Pressman, 2003].

Al realizar el análisis de Requisitos se debe detectar y corregir las falencias comunicativas, transformando los requisitos obtenidos de entrevistas , en condiciones apropiadas para ser tratados por el diseño, utilizando un procedimiento iterativo, se irán eliminando requisitos, se irán combinando y/o modificando para conseguir satisfacer los objetivos planteados.

c) Validación de requisitos

La validación de requisitos se encarga de la revisión técnica formal de los requisitos encontrados, se debe identificar cuales requisitos no serán abordados por el sistema. El equipo de revisión debe incluir a ingenieros del sistema, clientes, usuarios, y otros

intervinientes que examinan la especificación del sistema buscando errores en el contenido o en la interpretación, áreas donde se necesitan aclaraciones, información incompleta, inconsistencias, requisitos contradictorios, o requisitos imposibles o inalcanzables.

2.3 Análisis

En todo desarrollo de software, la fase de análisis trata de definir de manera formal las tareas que el software debe realizar, las respuestas que se esperan por parte del sistema, que dependerán de las acciones del usuario. En la fase de análisis se debe definir de manera clara todos los módulos que formaran todo el sistema.

Para crear una aplicación software hay que describir el problema y las necesidades o requerimientos, en que consiste el conflicto y que debe hacerse. El análisis se centra en la investigación del problema, no en la manera de definir la solución [Larman, 1999].

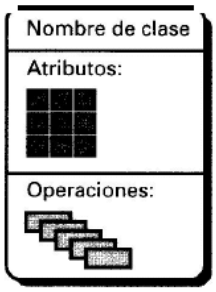
La abstracción de un sistema real y representarlo de tal forma que pueda ser integrado en un sistema computarizado, se denomina análisis orientado a objetos. La tarea de Análisis Orientado a Objetos es definir todos los objetos que participan en un sistema, cuales de estos son relevantes, como se comportan los objetos en el contexto del sistema, todo esto para poder construir modelos que servirán para el desarrollo del sistema.

2.3.1 El Paradigma Orientado a Objetos

Durante muchos años el término orientado a objetos (OO) se usó para referirse a un enfoque de desarrollo de software que usaba uno de los lenguajes orientados a objetos (Ada 95, C++, Eiffel, Smalltalk, etc.). Hoy en día el paradigma OO encierra una completa visión de la ingeniería del software [Pressman, 2003].

Dos conceptos básicos de un enfoque OO son el de clase y objeto, una clase es un concepto OO que encapsula las abstracciones de datos y procedimientos que se requieren para describir el contenido y comportamiento de alguna entidad del mundo real, la Figura 2.2 representa una clase, un objeto sera la instancia de una clase, este objeto tendrá todos los atributos y operaciones de la clase del que fue instanciado.

Figura 2.2 Representación de una Clase



Fuente: [Pressman, 2003]

La Figura 2.3 define la clase Auto, que tiene como atributos marca, modelo, numero de placa y color, estos son los atributos que se pueden encontrar en cualquier auto. Las operaciones asociadas a la clase son ingresar datos, cambiar color y cambiar placa, estas operaciones pueden estar definidas dentro de un contexto de un sistema de compra y venta de automóviles.

La ingeniería del software OO hace hincapié en la reutilización. Por lo tanto, las clases se buscan en una biblioteca (de clases OO existentes) antes de construirse.

Figura 2.3 Clase con atributos y operaciones

Fuente: Elaboración propia

Cuando una clase no puede encontrarse en la biblioteca, el desarrollador debe crear la clase y los objetos derivados de la clase. La nueva clase se pone en la biblioteca de tal manera que pueda ser reutilizada en el futuro, será extremadamente difícil definir las clases necesarias para un gran sistema o producto en una sola iteración.

2.3.2 Lenguaje de Modelado Unificado

El Lenguaje Unificado de Modelado o UML *(Unified Modellng Language),* prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Mientras que ha habido muchas notaciones y métodos usados para el diseño orientado a objetos, ahora los modeladores sólo tienen que aprender una única notación [Popkin, 2002].

UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real. UML es un lenguaje de modelado, y no un método.

El lenguaje de modelado es la notación (principalmente gráfica) de que se valen los métodos para expresar los diseños. Así pues, en gran medida el lenguaje de modelado es la parte más importante del método, ciertamente es la clave para la comunicación. Si se desea analizar el diseño con alguien, lo que ambos necesitan comprender es el lenguaje de modelado [Fowler, 1999].

UML ofrece nueve diagramas en los cuales modelar sistemas, son los siguientes:

- Diagramas de Casos de Uso para modelar los procesos;
- Diagramas de Secuencia para modelar el paso de mensajes entre objetos;
- Diagramas de Colaboración para modelar interacciones entre objetos;
- Diagramas de Estado para modelar el comportamiento de los objetos en el sistema;
- Diagramas de Actividad para modelar el comportamiento de los Casos de Uso, objetos u operaciones;
- Diagramas de Clases para modelar la estructura estática de las clases en el sistema;
- Diagramas de Objetos para modelar la estructura estática de los objetos en el sistema;
- Diagramas de Componentes para modelar componentes;
- Diagramas de Implementación para modelar la distribución del sistema.

2.3.3 Proceso de Análisis Orientado a Objetos

El proceso de Análisis Orientado a Objetos, no comienza con una preocupación por los objetos. Más bien comienza con una comprensión de la manera en la que se usará el

sistema: por las personas, si el sistema es de interacción con el hombre; por otras máquinas, si el sistema está envuelto en un control de procesos; o por otros programas; si el sistema coordina y controla otras aplicaciones. Una vez que se ha definido el escenario, comienza el modelado del software [Pressman, 2003]. El proceso OO comienza con la comunicación con el usuario, es aquí donde se define el dominio del problema y se identifican las clases básicas del problema.

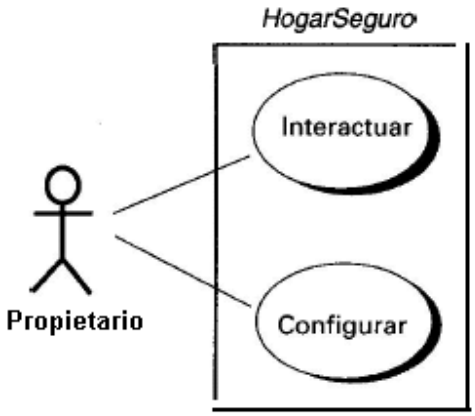
2.3.4 Especificación de Casos de Uso

El primer paso según [Pressman, 2003] para el análisis OO, es el modelado de Casos de Uso, es la técnica más efectiva y a la vez la más simple para modelar los requisitos del sistema desde la perspectiva del usuario. Los Casos de Uso se utilizan para modelar cómo un sistema o negocio que funciona actualmente, o cómo los usuarios desean que funcione. No es realmente una aproximación a la orientación a objetos sino una forma de modelar procesos. Es, sin embargo, una manera muy buena de dirigirse hacia el análisis de sistemas orientado a objetos. Los casos de uso son generalmente el punto de partida del análisis orientado a objetos con UML [Popkin, 2002].

Diagramas de Casos de Uso

El modelo de casos de uso consiste en actores y casos de uso. Los actores representan usuarios y otros sistemas que interaccionan con el sistema. Actualmente representan el tipo de usuario, no una instancia de usuario. Los casos de uso representan el comportamiento del sistema, los escenarios que el sistema atraviesa en respuesta a un estímulo desde un actor, la Figura 2.4 representa un diagrama de caso de uso a alto nivel, la cual representa dos casos de uso referentes a un sistema de seguridad el actor, propietario, está relacionado con los casos de uso expuestos, interactuar y configurar que hacen referencia al sistema de seguridad en si.

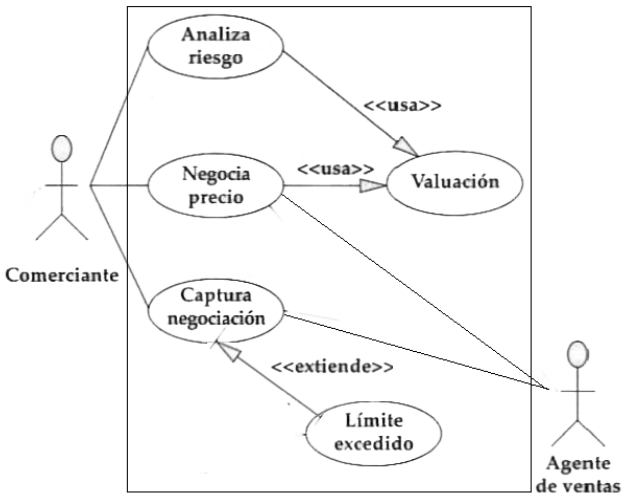
Figura 2.4 Diagrama de casos de uso de alto nivel



Fuente: [Pressman, 2003].

Cada caso de uso de alto nivel puede detallarse mediante diagramas de casos de uso de nivel inferior, como se puede ver en la Figura 2.5 representa un diagrama de casos de uso detallado.

Figura 2.5 Diagrama de casos de uso detallado



Fuente: [Fowler, 1999].

Además de los vínculos entre los actores y los casos de uso, hay otros dos tipos de vínculos en la Figura 2.5 estos representan las relaciones de usa (*uses*) y extiende (*extends*) entre los casos de uso.

Se usa la relación **extiende** cuando se tiene un caso de uso que es similar a otro, pero que hace un poco más. Las relaciones **usa** ocurren cuando se tiene una porción de comportamiento que es similar en más de un caso de uso y no se quiere copiar la descripción de tal conducta [Fowler, 1999].

Tratándose de la relación extiende, los actores tienen que ver con los casos de uso que se están extendiendo. Se supone que un actor dado se encargará tanto del caso de uso base como de todas las extensiones. En cuanto a las relaciones uses, es frecuente que no haya un actor asociado con el caso de uso común. Incluso si lo hay. no se considera que esté llevando a cabo los demás casos de uso [Fowler, 1999].

Casos de Uso en formato expandido

El caso de uso es un documento narrativo que describe la secuencia de eventos de un actor, que utiliza un sistema para completar un proceso. Los casos de uso son historias o casos de utilización de un sistema [Larman, 1999]. Un caso de uso en formato expandido contiene la descripción del caso de uso como se observa en la Tabla 2.2.

Tabla 2.2 Descripción de un caso de uso expandido

Caso de uso	Nombre de caso de uso
Actores	Lista de actores, en la cual se indica quien inicia el caso de uso
Propósito	Intención del caso de uso
Resumen	Repetición del caso de uso de alto nivel o alguna síntesis similar
Tipo	Primario, secundario, esencial o real
Referencias cruzadas	Casos relacionados de uso y funciones también relacionadas del sistema.

Fuente: [Larman, 1999]

Otro componente que es la parte medular de un caso de uso expandido, es el curso normal de eventos, un aspecto esencial es que explica la secuencia mas común de eventos, la historia normal de las actividades y la terminación exitosa de un proceso. La Tabla 2.3 muestra como se representa un curso normal de eventos.

Tabla 2.3 Curso Normal de Eventos

Acción del Actor	Respuesta del Sistema
Acciones numeradas de los actores.	Descripciones numeradas de las respuestas del sistema.

Fuente: [Larman, 1999]

La Última sección, el curso alterno de eventos, describe importantes opciones o excepciones que pueden presentarse en relación con el curso normal. Si son complejas podemos expandirlas y convertirlas en nuestros casos de uso.

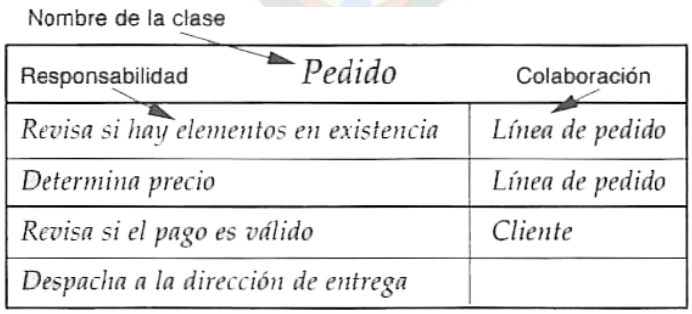
El curso alterno de eventos se sitúa debajo del curso normal de eventos, bajo rotulo **Cursos Alternos**, seguidamente debajo del rotulo debe describirse la excepción y el número de línea en la que puede ocurrir dicha excepción.

2.3.5 Modelado de Clases Responsabilidad Colaboracion (CRC)

Después de definir los casos de uso básicos para el sistema, es el momento de identificar las clases candidatas e indicar sus responsabilidades y colaboraciones. El modelado de clases-responsabilidades-colaboraciones (CRC) es un medio sencillo de identificar y organizar las clases que resulten relevantes al sistema [Pressman, 2003].

Una tarjeta CRC es una tabla dividida en tres partes, en la cabecera se debe poner el nombre de la clase, en la parte inferior izquierda las responsabilidades y en la parte derecha las clases colaboradoras que ayudan a cumplir las responsabilidades la Figura 2.6 Ilustra una tarjeta CRC.

Figura 2.6 Tarjeta Clase-Responsabilidad-Colaboracion



Fuente: [Fowler, 1999]

Las **responsabilidades** son en realidad es una descripción de alto nivel del propósito de una clase. La idea es tratar de eliminar la descripción de pedazos de datos y procesos y, en cambio, captar el propósito de la clase en unas cuantas frases. Con cada responsabilidad, están los **colaboradores,** que indican cuáles son las otras clases con las que se tiene que trabajar para cumplirla. Esto da cierta idea sobre los vínculos entre las clases, siempre a alto nivel. En ciertos casos puede que una responsabilidad sea manejada por la misma clase utilizando sus propias operaciones para cumplir con la misma.

Según [Pressman, 2003], las responsabilidades deben distribuirse de manera equitativa en todas las clases de la aplicación, de esta forma será mas fácil una modificación posterior, puesto que la inteligencia del sistema estará descompuesta en objetos.

2.3.6 Definición de Estructuras y Jerarquías

Una vez que se han identificado las clases y objetos usando el modelo CRC, el analista comienza a centrarse en la estructura del modelo de clases y las jerarquías. Las jerarquías pueden representar generalización-especialización, composición de clases, agregación de clases, las mismas que luego serán representadas en el diagrama de clases que se realizara al definir el Modelo Objeto Relación.

2.3.7 Modelo Objeto Relación

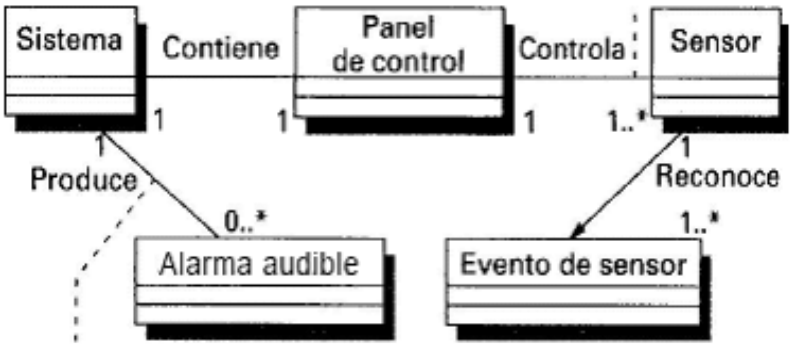
El enfoque del modelado CRC establece los primeros elementos de las relaciones de clases y objetos, esto comprendiendo las responsabilidades de cada clase y las clases colaboradoras que ayudan en la realización de cada responsabilidad. Esto establece la conexión entre las clases.

Entre dos clases cualesquiera que estén conectadas existe una relación. El tipo de relación más común es la binaria (existe una relación entre dos clases). Cuando se analiza dentro del contexto de un sistema OO, una relación binaria posee una dirección específica que se define a partir de qué clase desempeña el papel del cliente y cuál actúa como servidor. Una relación binaria se puede observar en la Figura 2.7 entre las clases panel de control y sistema.

El modelo objeto-relación puede obtenerse en tres pasos o etapas:

1) Usando las tarjetas índice CRC, puede dibujarse una red de objetos colaboradores, utilizando las conexiones entre clases;
2) Revisando el modelo de tarjetas CRC, se evalúan responsabilidades y colaboradores y cada línea de conexión sin etiquetar recibe un nombre. Para evitar ambigüedades en casos particulares , una punta de flecha indica la dirección de la relación;
3) Una vez que se han establecido y nombrado las relaciones, se evalúa cada extremo para determinar cardinalidad. Existen cuatro opciones: cero a uno (0 a 1), uno a uno (1 a 1), cero a muchos (0 a *), ó uno a muchos (1 a *).

Figura 2.7 Relaciones entre Clases



Fuente: [Pressman, 2003]

2.3.8 Modelo Objeto Comportamiento

El modelo CRC y el de objeto-relación representan elementos estáticos del modelo de análisis OO. También se debe representar el comportamiento dinámico del sistema o producto OO. Para ejecutar este paso debemos representar el comportamiento del sistema como una función de sucesos específicos y tiempo [Pressman, 2003].

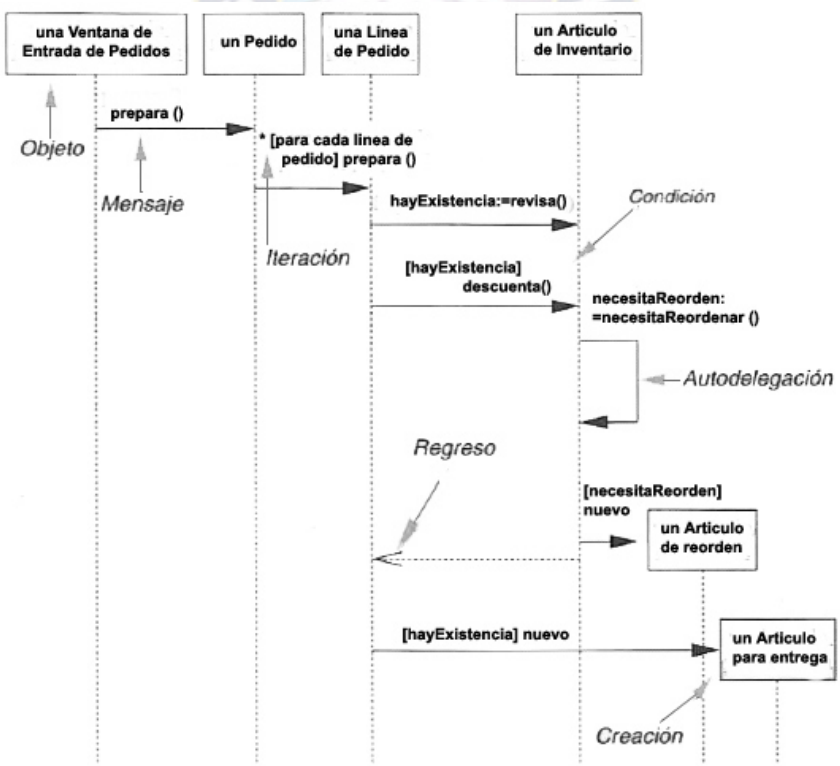
Se evalúan cada caso de uso y se identifican los sucesos que se relacionan con objetos, se trazan los suceso para cada caso de uso, en este contexto se pueden utilizar los diagramas de interacción que ofrece UML.

Diagrama de Secuencia

El Diagrama de Secuencia es uno de los diagramas más efectivos para modelar interacción entre objetos en un sistema. Un diagrama de secuencia se modela para cada caso de uso. Mientras que el diagrama de caso de uso permite el modelado de una vista mas general del escenario, el diagrama de secuencia contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes pasados entre los objetos.

Un diagrama de secuencia muestra los objetos que intervienen en el escenario con líneas discontinuas verticales, y los mensajes pasados entre los objetos como vectores horizontales (ver Figura 2.8). Los mensajes se dibujan cronológicamente desde la parte superior del diagrama a la parte inferior; la distribución horizontal de los objetos es arbitraria.

Figura 2.8 Diagrama de Secuencia



Fuente: [Fowler, 1999]

La línea vertical se llama línea de vida del objeto. La línea de vida representa la vida del objeto durante la interacción. Esta forma fue popularizada inicialmente por Jacobson. Cada mensaje es etiquetado por lo menos con el nombre del mensaje; pueden incluirse también los argumentos y alguna información de control. y se puede mostrar la autodelegación, que es un mensaje que un objeto se envía a sí mismo, regresando la flecha de mensaje de vuelta a la misma línea de vida como se ve en la Figura 2.8.

Dos partes de la información de control son valiosas. Primero, hay una condición, que indica cuándo se envía un mensaje (por ejemplo, [necesitaReorden] ver Figura 2.8*).* El mensaje se envía sólo si la condición es verdadera. El segundo marcador de control útil es el marcador de iteración, que muestra que un mensaje se envía muchas veces a varios objetos receptores, como sucedería cuando se itera sobre una colección. La base de la iteración se puede mostrar entre corchetes (como en *[para cada línea de pedido] ver Figura 2.8).

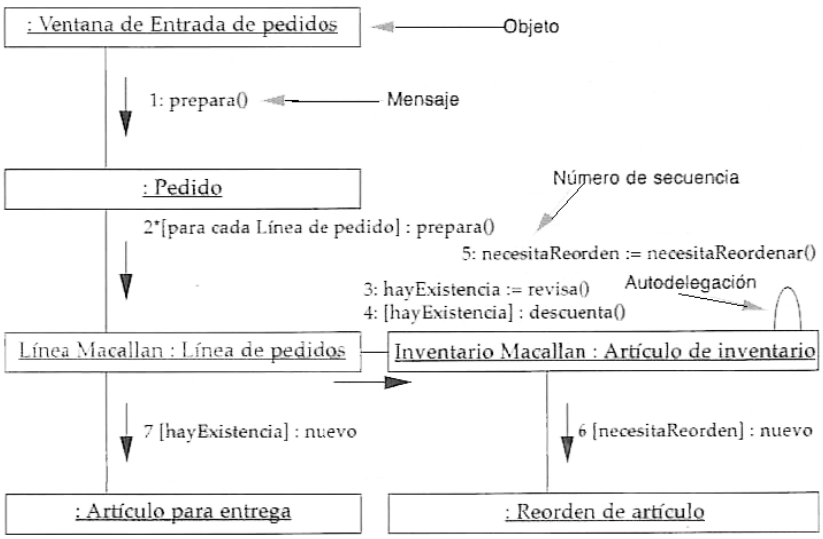
El diagrama incluye un regreso, el cual indica el regreso de un mensaje, no un nuevo mensaje. Los regresos difieren de los mensajes normales en que la línea es punteada, estos pueden graficarse o no, pues en el caso de no estar presente en el diagrama, igual se consideran implícitamente.

Diagrama de Colaboración

Un diagrama de colaboraciones es otra forma de presentar la información de un diagrama de secuencias. Ambos tipos de diagramas son semánticamente equivalentes y se recomienda usar ambos cuando se construye el modelo de un sistema. El diagrama de secuencias se organiza de acuerdo al tiempo, y el de colaboración de acuerdo al espacio [Schmuller, 1999].
Los objetos se conectan por medio de enlaces, cada enlace representa una instancia de una asociación entre las clases implicadas, los mensajes entre clases se envían por los enlaces.

En la Figura 2.9 se pueden apreciar las distintas formas del esquema de nombrado de objetos en UML. Se puede usar alguno de los diversos esquemas de numeración para los diagramas de colaboración. El más simple se ilustra en la Figura.2.9, se debe poner la cifra seguida de dos puntos (:) y el mensaje.

Figura 2.9 Diagrama de Colaboración



Fuente: [Fowler, 1999]

2.4 Modelado de Datos

La mayoría de los sistemas software grandes utilizan bases de datos de gran tamaño. Una parte importante del modelado de sistemas es la definición de la forma lógica de los datos procesados por el sistema. La técnica de modelado de datos mas ampliamente utilizada es el modelado Entidad- Relación (ER), que muestra las entidades de datos sus atributos asociados y las relaciones entre estas entidades [Sommer, 2005].

Puesto que en el modelo ER se pueden definir, supertipos y subtipos, lo que se denomina generalización, es fácil de implementar una base de datos OO utilizando el modelado ER.

UML no incluye una notación específica para el modelado de base de datos, pero se puede usar para representar un modelo semántico de datos. Se puede pensar en las entidades de un modelo ER, como clases de objetos simplificadas (sin operaciones), los atributos como atributos de la clase y las denominadas asociaciones entre clases como relaciones [Sommer, 2005].

El diagrama de clase de UML se puede usar para modelar algunos aspectos del diseño de bases de datos relacionales, pero no cubre toda la semántica involucrada en el modelado relacional, mayoritariamente la noción de atributos clave que relacionan entre sí

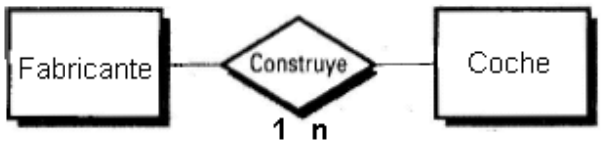
las tablas unas con otras. Para capturar esta información, un Diagrama Entidad Relación se recomienda como extensión a UML [Popkin, 2002].

Diagrama Entidad Relación

El Diagrama Entidad Relación (DER), fue propuesto originalmente por Peter Chen para el diseño de sistemas de bases de datos relacionales y ha sido ampliado por otros. Se identifica un conjunto de componentes primarios para el DER: objetos de datos, atributos, relaciones y varios indicadores tipo. El propósito primario del DER es representar objetos de datos y sus relaciones.

Los objetos de datos son representados por un rectángulo etiquetado. Las relaciones se indican mediante una línea etiquetada conectando objetos. En algunas variaciones del DER, la línea de conexión contiene un rombo que se etiqueta con la relación (Ver Figura 2.10). Las conexiones entre objetos de datos y relaciones se establecen mediante una variedad de símbolos especiales que indican cardinalidad.

Figura 2.10 Diagrama Entidad Relación



Fuente: [Pressman, 2003]

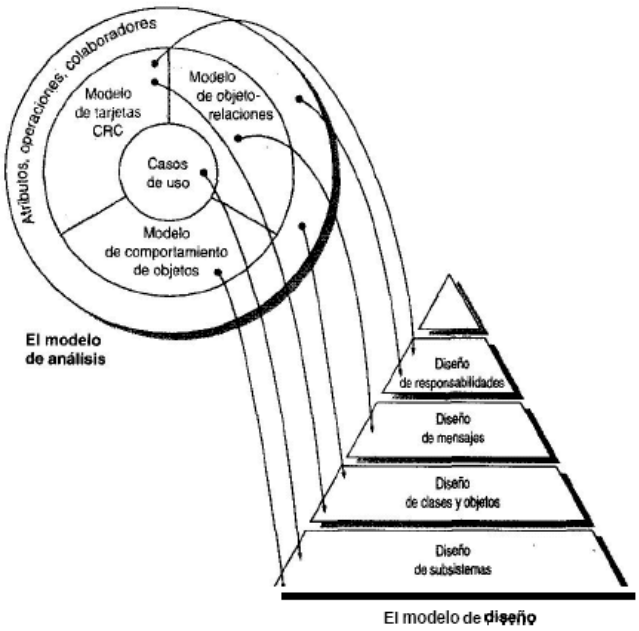
La Figura 2.10 muestra la relación Construye entre la entidad fabricante y la entidad coche, la cual tiene una cardinalidad de 1 a N, que indica que un fabricante puede construir N coches. La cardinalidad puede ser tomada de la multiplicidad de un modelo objeto-relación.

2.5 Diseño

Lo fundamental en la etapa de diseño es traducir la tarea de análisis, a un modelo de implementación, debe centrarse en el diseño de subsistemas, que implementan funciones principales de sistema. Según [Pressman, 2003], el diseño Orientado a Objetos, se divide en dos grandes actividades, el diseño del sistema y el diseño de objetos. La Figura 2.11 representa la relación existente entre el Análisis y Diseño la pirámide del diseño consta de

cuatro capas, la capa de subsistema, capa de clases y objetos, capa de mensajes y la capa de responsabilidades.

Figura 2.11 Transformación de Modelo de AOO al Modelo de DOO



Fuente: [Pressman, 2003],

2.5.1 Proceso de diseño del Sistema

El diseño de sistema comprende la arquitectura básica, en esta fase se tomaran decisiones estratégicas. El proceso de diseño del sistema abarca las siguientes actividades:

- Partición del modelo de análisis en subsistemas.
- Identificar la concurrencia dictada por el problema.
- Asignar subsistemas a procesadores y tareas.
- Desarrollar un diseño para la interfaz de usuario.
- Elegir una estrategia básica para implementar la administración (gestión) de datos.
- Identificar recursos globales y los mecanismos de control requeridos para su acceso.
- Diseñar un mecanismo de control apropiado para el sistema, incluyendo administración de tareas.
- Considerar cómo deben manejarse las condiciones de frontera.

2.5.2 Proceso de diseño de Objetos

El diseño de objetos tiene que ver con el diseño detallado de los objetos y sus interacciones. Se completa dentro de la arquitectura global, definida durante el diseño del sistema y de acuerdo con las reglas y protocolos de diseño aceptados. El diseño del objeto está relacionado en particular con la especificación de los tipos de atributos, cómo funcionan las operaciones y cómo los objetos se enlazan con otros objetos [Pressman, 2003].

Se definen las estructuras de datos locales (para atributos), y se diseñan los algoritmos (para operaciones), para esto se debe hacer una descripción de objetos y un diseño de algoritmos y estructuras de datos.

Descripción de Objetos

Una descripción del diseño de un objeto puede ser de dos formas: (1) una descripción de protocolo que establece la interfaz de un objeto, definiendo cada mensaje que el objeto puede recibir y las operaciones que el objeto lleva a cabo cuando recibe un mensaje, o (2) una descripción de implementación que muestra detalles de implementación para cada operación implicada por un mensaje pasado a un objeto.

La descripción del protocolo no es nada más que un conjunto de mensajes y un comentario correspondiente para cada mensaje, como ejemplo podría ser, **MENSAJE(sensor.movimiento) -> leer: DEVUELVE sensor.ID, sensor. estado; // mensaje requerido para leer un sensor** [Pressman, 2003].

Una descripción de la implementación de un objeto, proporciona los detalles ocultos, que se requieren para la implementación. Una descripción de la implementación se compone de la siguiente información: (1) una especificación del nombre del objeto y referencia a la clase; (2) una especificación de las estructuras de datos privadas, con indicación de los datos y sus respectivos tipos; (3) una descripción de procedimientos de cada operación o, alternativamente, indicadores a dichas descripciones de procedimientos.

En un entorno Orientado a Objetos solo se debería usar la descripción de protocolo, puesto que la descripción de implementación contiene detalles que deberían estar ocultos (encapsulados).

Diseño de algoritmos y estructuras de datos

Las representaciones contenidas en el modelo de análisis y el diseño de sistema, proveen una especificación para todas las operaciones y atributos necesarios para la especificación de algoritmos y estructuras de datos.

Se deben crear algoritmos para implementar la especificación para cada operación. En muchas ocasiones, el algoritmo es una simple secuencia computacional o procedural. Sin embargo, si la especificación de la operación es compleja, será necesario modularizar la operación.

Las estructuras de datos se diseñan al mismo tiempo que los algoritmos. Ya que las operaciones manipulan los atributos de una clase, el diseño de estructuras de datos, que reflejan mejor los atributos, tendrán un fuerte sentido en el diseño algorítmico de las operaciones correspondientes.

Aunque existen muchos tipos diferentes de operaciones, normalmente se pueden dividir en tres grandes categorías: operaciones que manipulan los datos de alguna manera (por ejemplo, agregando, eliminando, reformateando, seleccionando), operaciones que ejecutan cálculos, y operaciones que monitorizan o supervisan (por ejemplo, operaciones que devuelven valores tipo booleano) al objeto para la ocurrencia de un suceso controlado.

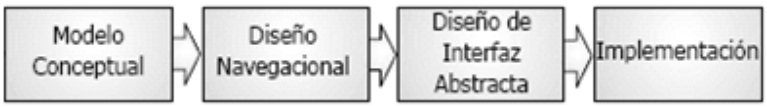
2.6 Metodología OOHDM

Las metodologías tradicionales de ingeniería de software, o las metodologías para desarrollo de sistemas de información, no contienen una buena abstracción capaz de facilitar la tarea de especificar aplicaciones hipermedia. Producir aplicaciones en las cuales el usuario pueda aprovechar el potencial del paradigma de la navegación de sitios *web*, mientras ejecuta transacciones sobre bases de información, es una tarea muy difícil de lograr. Si el usuario entiende dónde puede ir y cómo llegar al lugar deseado, es una buena señal de que la aplicación ha sido bien diseñada. Construir la interfaz de una aplicación web es también una tarea compleja, no sólo se necesita especificar cuáles son los objetos de la interfaz que deberían ser implementados, sino también la manera en la cual estos objetos interactuarán con el resto de la aplicación.

La metodología OOHDM, para diseño de aplicaciones hipermedia y para la Web, fue diseñado por Schwabe y Rossi, es una extensión de HDM con orientación a objetos, que se está convirtiendo en una de las metodologías más utilizadas. Ha sido usada para diseñar diferentes tipos de aplicaciones y sitios web [Lamarca, 2009].

OOHDM. es una metodología orientada a objetos que propone un proceso de desarrollo de cuatro fases donde se combinan notaciones gráficas UML con otras propias de la metodología. En una primera instancia debido al poco auge que tenía Internet, OOHDM era sólo para aplicaciones que incluían hipertexto y algo de multimedia (CD-ROM promocionales, enciclopedias, museos virtuales, etc.). Pero el gran desarrollo de Internet obligó su adaptación para el desarrollo de aplicaciones hipermedia en Internet, tales como comercio electrónico, motores de búsqueda, sitios educacionales y de entretención [Palma, 2004]. En la Figura 2.12 se grafican las cuatro etapas de OOHDM.

Figura 2.12 Etapas de la Metodología OOHDM



Fuente: [Palma, 2004]

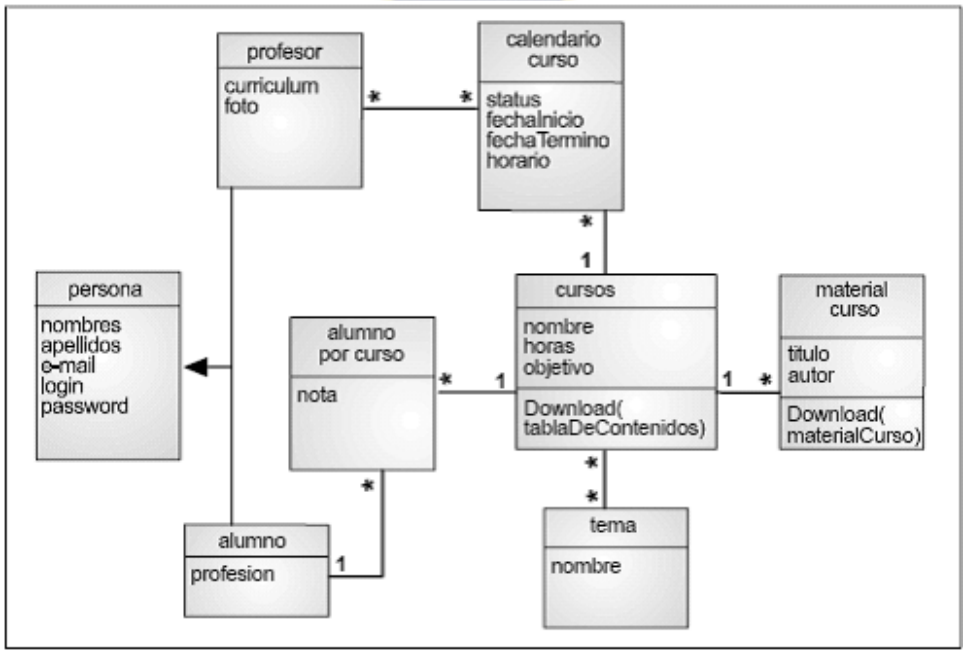
A continuación se describirá cada etapa de la metodología con más detalle, se define cómo el diseñador procede desde el Diseño Conceptual a través del Diseño de la Navegación al Diseño de la Interfaz y la Implementación.

2.6.1 Modelo Conceptual

Durante esta actividad se construye un esquema conceptual representado por los objetos del dominio, las relaciones y colaboraciones existentes establecidas entre ellos. En las aplicaciones hipermedia convencionales, cuyos componentes de hipermedia no son modificados durante la ejecución, se podría usar un modelo de datos semántico estructural (como el modelo de entidades y relaciones). De este modo, en los casos en que la información base pueda cambiar dinámicamente o se intenten ejecutar cálculos complejos, se necesitará enriquecer el comportamiento del modelo de objetos.

En OOHDM, el esquema conceptual está construido por clases, relaciones y subsistemas. Las clases son descritas como en los modelos orientados a objetos tradicionales. Sin embargo, los atributos pueden ser de múltiples tipos para representar perspectivas diferentes de las mismas entidades del mundo real [Silva, 2002]. La Figura 2.13 Ilustra el esquema conceptual, el modelo conceptual en OOHDM incluye el modelo de clases utilizado en métodos orientados a objeto tradicionales.

Figura 2.13 Esquema conceptual



Fuente: [Palma, 2004]

Siendo basado en UML, puede ser complementado obviamente con otros modelos del mismo usando casos de uso, diagramas de secuencia y demás artefactos UML, puesto que OOHDM no provee un método particular para esta fase. La Tabla 2.4 muestra las actividades que deben realizarse en esta etapa.

2.6.2 Diseño Navegacional

En OOHDM, la navegación es considerada un paso crítico en el diseño aplicaciones. Un modelo navegacional es construido como una vista sobre un diseño conceptual,

admitiendo la construcción de modelos diferentes de acuerdo con los diferentes perfiles de usuarios. Cada modelo navegacional provee una vista subjetiva del diseño conceptual [Silva, 2002].

El diseño de navegación es expresado en dos esquemas: el esquema de clases navegacionales y el esquema de contextos navegacionales. En OOHDM existe un conjunto de tipos predefinidos de clases navegacionales, nodos, enlaces y estructuras de acceso.

a) Esquema de Clases Navegacionales

Establece las posibles vistas del hiperdocumento a través de unos tipos predefinidos de clases, llamadas navegacionales como son los nodos, los enlaces y otras clases que representan estructuras o formas alternativas de acceso a los nodos, como los índices y los recorridos guiados.

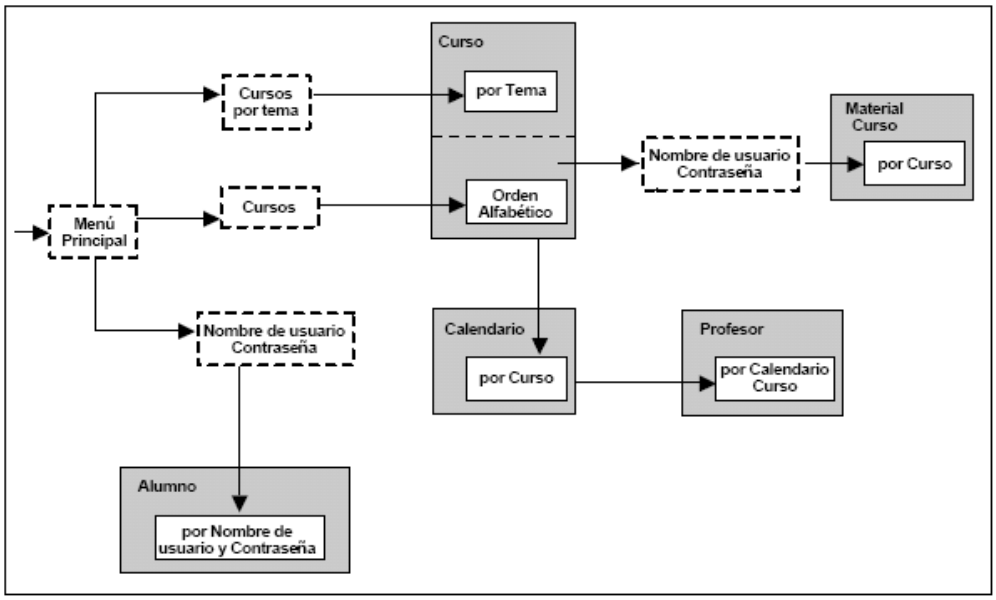
Los **Nodos** son contenedores de información, éstos se definen como vistas orientadas a objetos de las clases conceptuales. Los nodos se pueden definir combinando atributos de clases relacionadas en el esquema conceptual

Los **Enlaces** reflejan la relación de navegación que puede explorar el usuario, ellos serán imprescindibles para poder crear vistas diferentes de esquemas navegacionales. Las clases de los enlaces especifican sus atributos, comportamiento y los objetos fuentes del mismo, estos representan las posibles formas de comenzar la navegación. En cualquier caso, el enlace puede actuar como un objeto intermedio en un proceso de navegación o como un puente de conexión entre dos nodos.

b) Esquema de Contexto Navegacional

Es el que permite la estructuración del hiperespacio de navegación en sub-espacios para los que se indica la información que será mostrada al usuario y los enlaces que estarán disponibles cuando se accede a un objeto (nodo) en un contexto determinado. La Figura 2.14 ilustra un diagrama de Contexto Navegacional.

Figura 2.14 Diagrama de Contexto Navegacional



Fuente: [Palma, 2004]

Como se puede ver en la Figura 2.14 un contexto navegacional es un conjunto de nodos, enlaces, clases de contextos y otros contextos navegacionales (contextos anidados). Las actividades a tomarse en cuenta en esta etapa se encuentran en la Tabla 2.4.

Tabla 2.4 Actividades de la fase del Diseño Navegacional

DISEÑO NAVEGACIONAL	
PRODUCTOS	Nodos, enlaces, estructuras de acceso, contextos navegacionales y transformaciones navegacionales.
HERRAMIENTAS	Técnicas de modelado OO, patrones de diseño, diagramas de estados, escenarios.
MECANISMOS	Clasificación, agregación, generalización y especialización.
OBJETIVOS DEL DISEÑO	Diseñar los recorridos que el usuario puede seguir por la aplicación.

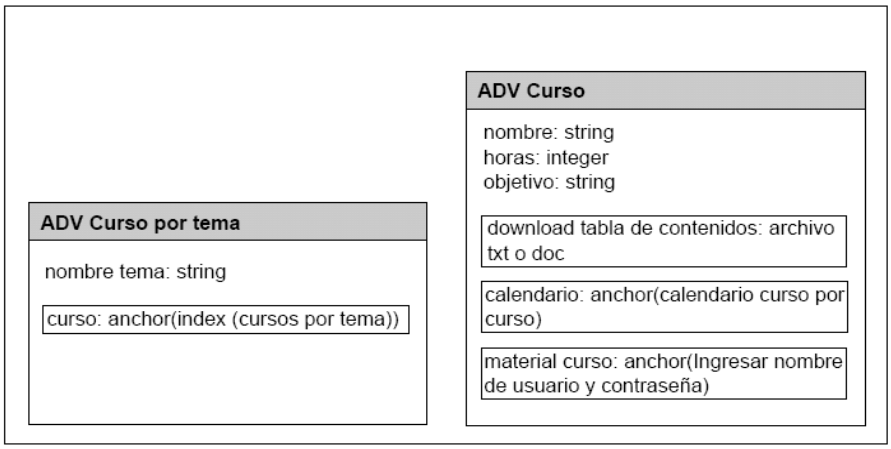
Fuente: [SCHRO, 1998]

2.6.3 Diseño de la Interfaz Abstracta

Una vez finalizado el diseño navegacional, será necesario especificar las diferentes interfaces de la aplicación. Esto significa definir de que manera aparecerán los objetos navegacionales en la interfaz y cuales objetos activarán la navegación. Para lograr esto

se utilizarán ADVs(Vista de Datos Abstracta), modelos abstractos que especifican la organización y el comportamiento de la interfaz, es necesario aclarar que las ADVs representan estados o interfaces y no la implementación propiamente tal (ver Figura 2.15).

Figura 2.15 Representación de ADV's



Fuente: [Palma, 2004]

El modelo de interfaz ADVs (Vista de Datos Abstracta) especifica la organización y comportamiento de la interfaz, pero la apariencia física real o de los atributos, y la disposición de las propiedades de las ADVs en la pantalla real son hechas en la fase de implementación.

Las actividades comunes en la etapa de Diseño de la interfaz Abstracta se pueden observar en la Tabla 2.5.

Tabla 2.5 Actividades de la fase del Diseño de la Interfaz Abstracta

DISEÑO DE LA INTERFAZ ABSTRACTA	
PRODUCTOS	Objetos de Interfaz abstracta, respuestas a eventos externos, y transformaciones de la interfaz.
HERRAMIENTAS	Vistas, datos abstractos, patrones de diseño.
MECANISMOS	Mapeo entre navegación y objetos perceptibles.
OBJETIVOS DEL DISEÑO	Modelado de los objetos perceptibles por el usuario, descripción de interface navegacional.

Fuente: [SCHRO, 1998]

Un ADV usado en el diseño de aplicaciones Web puede verse como un objeto de interfaz. Comprende un conjunto de atributos (y objetos de interfaz anidado) qué define sus propiedades de percepción, y el conjunto de eventos que puede manejar, como eventos generados por el usuario.

2.6.4 Implementación

En esta fase, se debe implementar el diseño. Hasta ahora, todos los modelos fueron construidos en forma independiente de la plataforma de implementación; en esta fase es tenido en cuenta el entorno particular en el cual se va a correr la aplicación.

Al llegar a esta fase, el primer paso que debe realizar el diseñador es definir los ítems de información que son parte del dominio del problema. Debe identificar también, cómo son organizados los ítems de acuerdo con el perfil del usuario y su tarea; decidir qué interfaz debería ver y cómo debería comportarse. A fin de implementar todo en un entorno web, el diseñador debe decidir además qué información debe ser almacenada. La Tabla 2.6 contiene las actividades a ser tomadas en cuenta en esta etapa.

Tabla 2.6 Actividades de la fase de Implementación

IMPLEMENTACION	
PRODUCTOS	Aplicación ejecutable
HERRAMIENTAS	El lenguaje de programación soportado por entorno.
MECANISMOS	Los ofrecidos por el propio lenguaje.
OBJETIVOS DEL DISEÑO	Aplicación ejecutable, rendimiento.

Fuente: [SCHRO, 1998]

2.7 Pruebas Orientadas a Objetos

La estrategia clásica para la prueba de software de ordenador, comienza con probar lo pequeño y funcional hacia fuera haciendo probar lo grande. Siguiendo los pasos de la prueba de software tradicional**,** se comienza con las pruebas de unidad, después se progresa hacia las pruebas de integración y se culmina con las pruebas de validación del sistema [Pressman, 2003]. A diferencia del software convencional, en las pruebas para software OO se toma a las clases para realizar las distintas pruebas.

Pruebas de Unidad

En las pruebas de unidad no se toman en cuenta los módulos como en el software convencional, la unidad más pequeña comprobable es la clase u objeto encapsulado, ya que esta tiene atributos y operaciones en su interior. La visión de prueba de unidad en un ambiente OO cambia drásticamente, puesto que la prueba de clases para el software OO se conduce mediante las operaciones encapsuladas por la clase y el comportamiento de la clase.

Pruebas de Integración

Existen dos estrategias diferentes para las pruebas de integración de los sistemas OO. El primero, las **pruebas basadas en hilos**, integran el conjunto de clases requeridas, para responder una entrada o suceso al sistema. La segunda aproximación de integración, la **prueba basada en el uso**, comienza la construcción del sistema probando aquellas clases (llamadas clases independientes), que utilizan muy pocas (o ninguna) clases servidoras. Después de que las clases independientes se prueban, esta secuencia de pruebas por capas de clases dependientes continúa hasta que se construye el sistema completo.

Pruebas de Validación

Así como la validación convencional, la validación del software OO se centra en las acciones visibles al usuario y salidas reconocibles desde el sistema. Para ayudar en la construcción de las pruebas de validación, el probador debe utilizar los casos de uso, que son parte del modelo de análisis. Los casos de uso proporcionan un escenario, que tiene una gran similitud de errores con los revelados en los requisitos de interacción del usuario.

Los métodos de prueba convencionales de caja negra pueden usarse para realizar pruebas de validación. Además, los casos de prueba deben derivarse del modelo de comportamiento del objeto y del diagrama de flujo de sucesos, creado como parte del AOO.

2.8 Mantenimiento de Software

El mantenimiento de Software se centra en el cambio que va asociado a la corrección de errores, a las adaptaciones requeridas a medida que evoluciona el entorno del software y

a cambios debidos a las mejoras producidas por los requisitos cambiantes del cliente. Durante la fase de mantenimiento se encuentran cuatro tipos de cambios [Pressman, 2003]:

- **Corrección,** el mantenimiento correctivo se llevara a cabo cuando un usuario encuentre alguna función errónea, que retorne datos que no son coherentes con lo que el usuario espera.
- **Adaptación,** el mantenimiento adaptativo produce modificación en el software para acomodarlo a los cambios de su entorno externo, se realizará un mantenimiento adaptativo, en el caso de que las funciones de los usuarios cambien, o en el caso de asignarles mas funciones debido a un cambio en el reglamento interno de la institución.
- **Mejora,** conforme se utilice el software, el usuario puede descubrir funciones adicionales que van a producir beneficios. Estas funciones encontradas por los usuarios se implementaran siempre y cuando no lleve a un cambio drástico en toda la estructura del sistema implementado.
- **Prevención.** El software de computadora se deteriora debido al cambio, y por esto el mantenimiento preventivo también llamado reingeniería del software, se debe conducir a permitir que el software sirva para las necesidades de los usuarios finales. En esencia, el mantenimiento preventivo hace cambios en programas de computadora a fin de que se puedan corregir, adaptar y mejorar más fácilmente, estos cambios se realizaran en los módulos mas complejos del sistema.

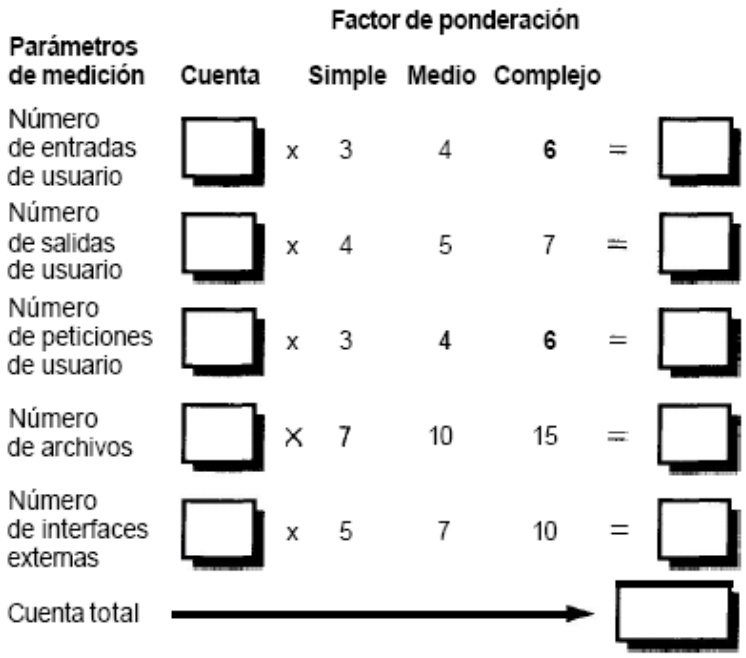
2.9 Métricas Orientadas a la Función

Las métricas del software orientadas a la función utilizan una medida de la funcionalidad entregada por la aplicación como un valor de normalización. Las métricas orientadas a la función fueron propuestas por primera vez por Albretch, quien sugirió una medida llamada **punto de función***.* Los puntos de función se derivan con una relación empírica según las medidas contables (directas) del dominio de información del software y las evaluaciones de la complejidad del software [Pressman, 2003].

Los puntos de función se calculan completando la tabla de la Figura 2.16.
Los valores de los dominios de información se definen de cómo sigue:

- Número de entradas de usuario, se cuenta cada entrada de usuario que proporciona diferentes datos orientados a la aplicación;
- Número de salidas de usuario, se cuenta cada salida que proporciona al usuario información orientada a la aplicación. En este contexto la salida se refiere a informes, pantallas, mensajes de error, etc;
- Número de peticiones de usuario, una petición se define como una entrada interactiva que produce la generación de alguna respuesta del software inmediata en forma de salida interactiva;
- Número de archivos. Se cuenta cada archivo maestro lógico (esto es, un grupo lógico de datos que puede ser una parte de una gran base de datos o un archivo independiente);
- Número de interfaces externas. Se cuentan todas las interfaces legibles por la máquina (por ejemplo: archivos de datos de cinta o disco) que se utilizan para transmitir información a otro sistema.

Figura 2.16 Calculo de Puntos de Función



Fuente: [Pressman, 2003]

Para calcular puntos de función (PF), se utiliza la relación siguiente:

$$PF = \text{cuenta-total} \times [0{,}65 + 0{,}01 \times 6(F_i)]$$

En donde cuenta-total es la suma de todas las entradas PF obtenidas de la Figura 2.18. F_i (i = 1 a 14) son valores de ajuste de la complejidad según las respuestas a las siguientes preguntas:

1 ¿Requiere el sistema copias de seguridad y de recuperación fiables?

2 ¿Se requiere comunicación de datos?

3 ¿Existen funciones de procesamiento distribuido?

4 ¿Es crítico el rendimiento?

5 ¿Se ejecutan el sistema en un entorno operativo existente y fuertemente utilizado?

6 ¿Requiere el sistema entrada de datos interactiva?

7 ¿Requiere la entrada de datos interactiva que las transacciones de entrada se lleven a cabo sobre múltiples pantallas u operaciones?

8 ¿Se actualizan los archivos maestros de forma interactiva?

9 ¿Son complejas las entradas, las salidas, los archivos o las peticiones?

10 ¿Es complejo el procesamiento interno?

11 ¿Se ha diseñado el código para ser reutilizable?

12 ¿Están incluidas en el diseño la conversión y la instalación'?

13 ¿Se ha diseñado el sistema para soportar múltiples instalaciones en diferentes organizaciones?

14 ¿Se ha diseñado la aplicación para facilitar los cambios y para ser fácilmente utilizada por el usuario?

Cada una de las preguntas anteriores es respondida usando una escala con rangos desde 0 (no importante o aplicable) hasta *5* (absolutamente esencial). Los valores constantes de la ecuación y los factores de peso que se aplican a las cuentas de los dominios de información se determinan empíricamente.

3.1 Ingeniería de Requisitos

La tarea de Ingeniería de requisitos es fundamental para que un sistema sea exitoso, como se menciono en la sección 2.2.2 , en este sentido para la realización del presente proyecto se realizaron las 3 actividades como se indica en la Tabla 2.1, ahora se describirán cada una de ellas.

Obtención de Requisitos

Las tareas que se realizaron en esta fase son 5 las cuales se puede observar de forma resumida en la Tabla 3.1.

Tabla 3.1 Tareas en la obtención de Requisitos

Entrevistas personales	Se hicieron entrevistas personales a posibles usuarios potenciales, entrevistas frecuentes con el Director Académico de la U.E.R. P. Walter Strub, entrevista con la Secretaria, Regentes y algunos miembros del Plantel docente.
Observación	Se observaron los procesos que realizan los diferentes miembros que están directamente relacionados con la Información Académica del establecimiento, como ser la elaboración de boletines, horarios, kardex del personal, e inscripciones.
Papel de aprendiz	Se tomo el rol de aprendiz, en las actividades del establecimiento como ser: inscripción, elaboración de boletines, administración de los horarios.
Documentación	Fue posible tener copias de respaldo de la documentación que tiene actualmente la U.E.R.P. Walter Strub(ver Anexo C).

Análisis de Requisitos

Una vez que se realizo la obtención de requisitos con las tareas descritas en la tabla 3.1, se hace un análisis de los requisitos candidatos que se encontraron, los cuales se detallan a continuación:

- Registrar todos los datos del personal que trabaja dentro de la U.E.;
- Registrar los datos de estudiantes y apoderados que pertenecen a la U.E.;
- Registro de inscripciones de los estudiantes;
- Registro de calificaciones de estudiantes;
- Emisión de boletines de calificaciones de primaria y secundaria;
- Estructurar los horarios de los diferentes docentes y cursos;
- Llevar un control de asistencia y retrasos del personal;
- Llevar un control de asistencia y retrasos de estudiantes;
- Obtener listados de docentes y estudiantes, según se requiera.

En la Tabla 3.2 se detallan los requisitos funcionales que se detectaron a partir del análisis de requisitos

Tabla 3.2 Listado de Requisitos

Ref.	**Requisitos Funcionales**
R1	Guardar y actualizar los datos de estudiantes
R2	Guardar los datos de kardex del personal
R3	Registro y actualización de datos de apoderados
R4	Registro de materias por nivel y orden de libreta
R5	Registro de cursos por nivel y paralelo
R6	Inscripción de estudiantes
R7	Registrar faltas y atrasos de estudiantes
R8	Registrar faltas y atrasos del personal

R9	Asignación de materias a docentes
R10	Guardar datos días y periodos disponibles de un docente
R11	Designar cursos a todos los docentes
R12	Controlar la asignación de docentes a cursos
R13	Emisión de horarios por curso
R14	Emisión de horarios por docente
R15	Registro y actualización de calificaciones de estudiantes
R16	Emisión de boletines de calificación de los estudiantes
R17	Generar listados de estudiantes por curso
R18	Generar listados de docentes
R19	Contar con un mecanismo de seguridad de acceso al sistema
R20	Crear y eliminar usuarios

En la Tabla 3.3 se describen a los actores detectados en la obtención de requisitos, los mismos que interactuaran de alguna forma con el sistema.

Tabla 3.3 Actores del Sistema

Actor	**Objetivo**
Director	Planifica Horarios Controla la asistencia del personal Tiene acceso a toda la información académica Elabora los horarios para cursos y profesores
Docente	Registra calificaciones de los estudiantes Evalúa a los estudiantes de los cursos que le fueron asignados
Secretaria	Registra datos en kardex, de docentes como de estudiantes Registra las inscripciones de estudiantes

Regente	Registra asistencia de los estudiantes, registra retrasos, faltas con y sin licencia. Realiza una estadística con los datos de los estudiantes
Estudiante	Realiza la inscripción junto al apoderado Pertenece a un curso en una determinada gestión.
Apoderado	Realiza la inscripción junto al estudiante, tiene un cierto grado de parentesco con el estudiante
Personal Adm.	Pertenece a la U.E. tiene un rol especifico dentro de ella, o puede ser personal de apoyo

Validación de Requisitos

Para la validación de requisitos se aclararon ciertos requisitos con el cliente en este caso el Director Académico de la U.E.R.P. Walter Strub, los requisitos que se trataron son los siguientes:

- El proceso de generación de horarios se realizará previa selección de docentes por parte del Director Académico, para esto el sistema brindara bastante información acerca de materias, disponibilidad docente y posibles choques que pudiera existir al asignar un docente a un curso específico;
- El proceso de importación de calificaciones de los estudiantes por parte de los docentes, desde un archivo externo en formato Excel, se analizara en la implementación;
- Como requisito de interfaz el sistema deberá presentar una interfaz web amigable para cada tipo de usuario;
- El sistema operativo a utilizar será Windows XP SP2 con el cual cuenta la U.E., se utilizara el gestor de base de datos Postgres para el almacenamiento de la información, servidor Web Apache para correr el sistema, lenguaje de programación PHP, e Internet Explorer con Javascript habilitado para interactuar con el usuario.

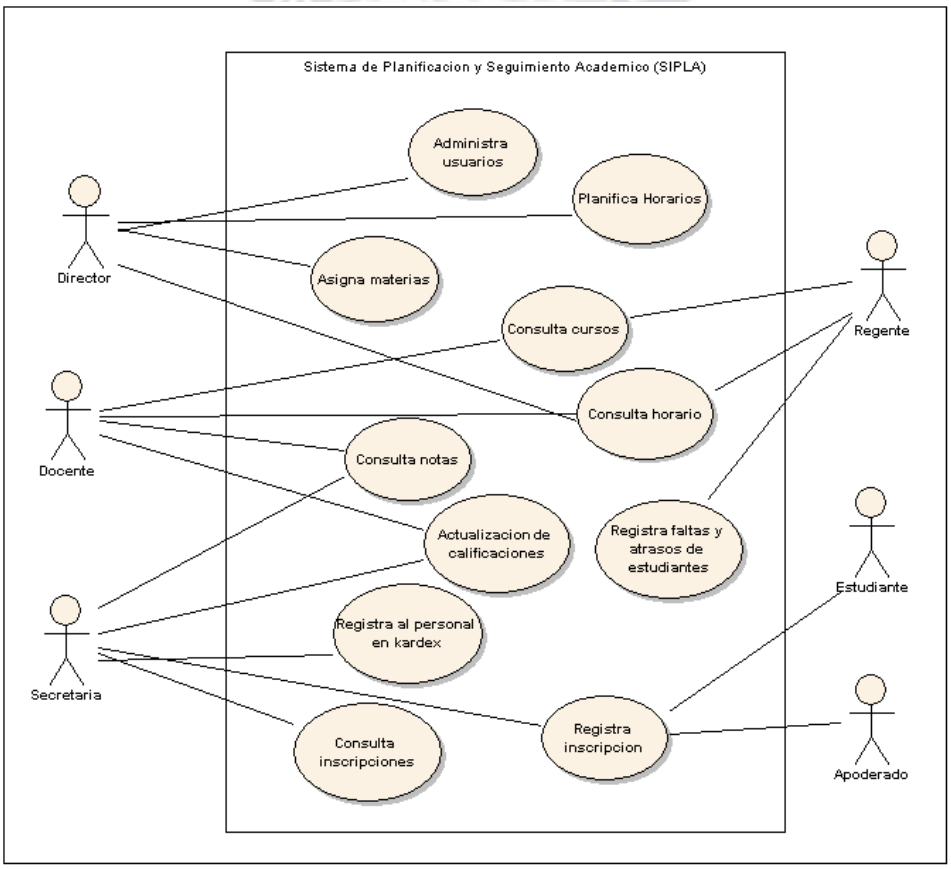
3.2 Análisis y Diseño

Como se menciono en el Capitulo 2, los pasos para el análisis OO según [Presman, 2003] comienzan con la especificación de los casos de uso, seguido de el Modelo CRC, el modelo Objeto-Relación y el Modelo Objeto-Comportamiento, se utilizara el diagrama de clases para el Modelo Objeto-Relación y se empleara los diagramas de secuencia y de colaboración para el Modelo Objeto-Comportamiento. Para complementar el diseño se verán los diagramas de contexto navegacional y el diseño de interfaz abstracta.

3.2.1 Casos de Uso

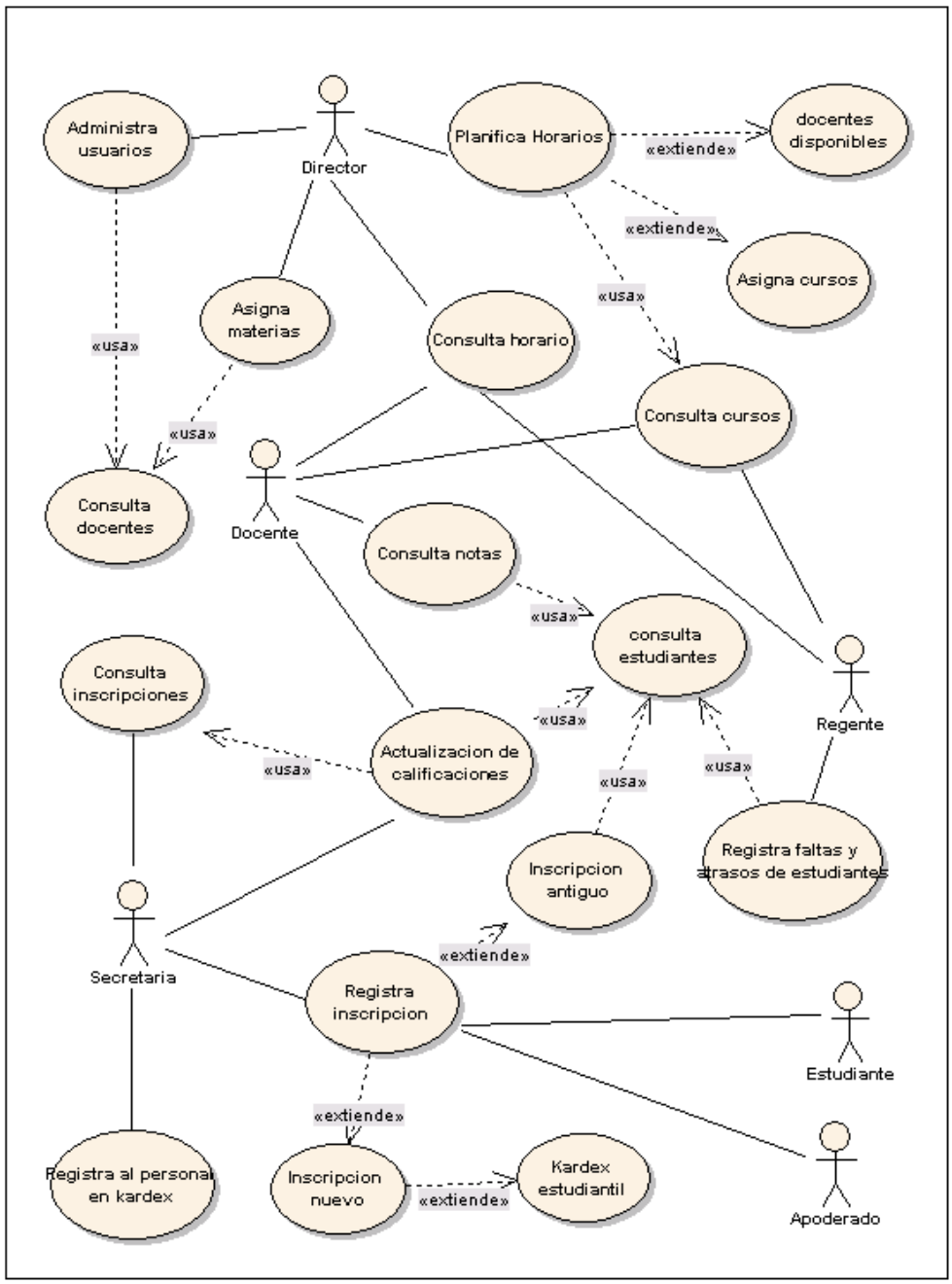
Luego de identificar los requisitos del sistema y además los actores, se procede con el modelado de casos de uso en la Figura 3.1 se observa el diagrama de casos de uso del Sistema de Planificación y Seguimiento Académico, donde se puede observar los casos de uso primarios del sistema.

Figura 3.1 Modelo de casos de uso del Sistema



Una vez obtenido el diagrama de casos de uso del sistema se puede obtener un diagrama de casos de uso mas detallado como se observa en la Figura 3.2, que representa los procesos que realiza cada caso de uso, el diagrama fue realizado en base al análisis de requisitos.

Figura 3.2 Diagrama de casos de uso detallado.



Habiendo detectado los casos de uso del sistema, se procede a hacer una descripción de los mismos. La Tabla 3.4 describe el caso de uso Planifica horarios junto al curso normal de eventos del mismo caso de uso. En la Tabla 3.4 indica un curso alterno en la línea 4, el cual indica que se realizara en el caso de existir un choque en la asignación de aulas a docentes, para lo cual verificara que el periodo que se quiere asignar no este ocupado por otro docente.

Tabla 3.4 Descripción del caso de uso Planifica horarios

<table>
<tr><td>Caso de uso</td><td colspan="2">Planifica Horarios</td></tr>
<tr><td>Actores</td><td colspan="2">Director</td></tr>
<tr><td>Propósito</td><td colspan="2">Realizar horarios para los diferentes cursos y docentes</td></tr>
<tr><td>Resumen</td><td colspan="2">El caso de uso se da cuando el director tiene que elaborar los horarios, entonces se verifica la disponibilidad de docentes para los diferentes cursos y se va asignando a cada día los diferentes docentes a las diferentes aulas.</td></tr>
<tr><td>Tipo</td><td colspan="2">Primario, esencial</td></tr>
<tr><td>Referencias cruzadas</td><td colspan="2">R9,R10,R11,R12</td></tr>
<tr><td colspan="3">Curso Normal de Eventos</td></tr>
<tr><td>Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td>1.- El director ingresa al sistema y desea planificar el horario

3. selecciona docentes para cada curso por periodo y día

5.el director posee información de los horarios de docentes y cursos</td><td colspan="2">2. muestra información sobre docentes disponibles para cada día.

4. verifica los datos y guarda los parámetros</td></tr>
<tr><td colspan="3">Cursos Alternos
Línea 4: si existe dos docentes asignados a un mismo curso en el mismo periodo se cancela la asignación.</td></tr>
</table>

La Tabla 3.5 describe el caso de uso asigna materias, el cual se presenta cuando el director académico debe definir las materias que puede dictar un determinado docente. El curso normal de eventos del caso de uso asigna materias que esta descrito en la misma, se puede apreciar el curso alterno en la línea 6 donde el sistema debe verificar el número máximo de materias que puede dictar un docente.

Tabla 3.5 Descripción del caso de uso Asigna materias

Caso de uso	Asigna materias
Actores	Director
Propósito	Definir que materias puede dictar un docente
Resumen	El caso de uso se da cuando el director tiene que definir las materias que un docente puede dictar para esto verifica la especialidad del docente y asigna un numero máximo de materias a cada docente..
Tipo	Primario, esencial
Referencias cruzadas	R4,R9,R18
Curso Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1.- El director ingresa al sistema y solicita el listado de docentes 3. selecciona un docente 5.selecciona las materias que el docente puede dictar 7. selecciona otro docente o sale del modulo	2. muestra información sobre docentes disponibles. 4. muestra las materias que puede asignar. 6. guarda las materias que puede dictar el docente.
Cursos Alternos Línea 6:.verifica el numero máximo de materias que puede dictar el docente, si excede del limite muestra un mensaje.	

La Tabla 3.6 describe al caso de uso Registra inscripción, junto a su respectivo curso normal de eventos, donde se puede apreciar que al inscribir un estudiante puede darse dos casos, que el estudiante sea antiguo o que el estudiante sea nuevo.

Tabla 3.6 Descripción del caso de uso Registra inscripción

Caso de uso	Registra inscripción
Actores	Secretaria, Estudiante, Apoderado
Propósito	Guarda los datos de inscripción de un estudiante
Resumen	Este caso de uso se da generalmente a principio de año cuando un alumno realiza su inscripción, para esto la secretaria registra los datos del estudiante del apoderado y además los datos del curso y paralelo al que pertenecerá el estudiante.
Tipo	Primario, esencial
Referencias cruzadas	R1, R3, R6

Curso Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1.- la secretaria ingresa al sistema y desea realizar la inscripción. 3. selecciona una opción de inscripción 5. llena o actualiza los datos del estudiante y apoderado asi como datos de la gestión actual. 7. continua inscribiendo a otros estudiantes o sale del sistema	2. muestra dos opciones una para alumnos antiguos y otra para alumnos nuevos. 4. muestra un formulario de inscripción de la opción elegida 6. verifica y guarda datos de la inscripción
Cursos Alternos Línea 4: en caso de ser alumno nuevo, muestra un formulario de kardex estudiantil Línea 5: siendo un alumno nuevo, se deberá llenar el kardex estudiantil con datos del estudiante y datos del apoderado del estudiante.	

La Tabla 3.7 describe el caso de uso registro de faltas y atrasos de estudiantes que esta directamente relacionado con el actor Regente, siendo el mismo que tiene la tarea del control de asistencia de los estudiantes. También muestra el curso normal de eventos del mismo caso de uso, describe de manera general los pasos que el regente realiza al anotar una falta o atraso de un estudiante

Tabla 3.7 Descripción del caso de uso Registra faltas y atrasos de estudiantes

Caso de uso	Registra faltas y atrasos de estudiantes
Actores	Regente
Propósito	Guardar datos de faltas y atrasos de un estudiante
Resumen	El caso de uso se da cuando el regente quiere guardar los datos de asistencia de los estudiantes, para registrar una falta o atraso de un estudiante lo selecciona de un curso.
Tipo	Primario
Referencias cruzadas	R7
Curso Normal de Eventos	

Acción del Actor	Respuesta del Sistema
1.- el regente ingresa al sistema y entra al modulo de faltas y atrasos de estudiantes 3. elige un curso de la lista 5. selecciona un estudiante 7. elige la opción requerida 9. elige otro estudiante o sale del sistema.	2. muestra una pantalla para seleccionar el curso y paralelo 4. muestra el listado de estudiantes. 6. se genera opciones de atraso o falta para el estudiante seleccionado 8. guarda las observaciones realizadas

3.2.2 Tarjetas CRC (Clases Responsabilidad Colaboracion)

Luego de identificar los casos de uso y haber realizado su descripción correspondiente, se identifican las clases candidatas e identificamos sus responsabilidades y clases colaboradoras correspondientes en las tarjetas CRC tal como se indico en la sección 2.3.5.

La Tabla 3.8 representa la tarjeta CRC de la clase Docente, que tiene como colaboradores a las clases horario y curso, se puede apreciar la responsabilidad, define tiempo libre, la cual hace referencia a que un docente tiene un tiempo libre definido para poder dictar clases en el establecimiento.

Tabla 3.8 Tarjeta CRC para la clase Docente

Docente	
Ingresar la calificación	Clase Estudiante
Revisa materias asignadas	Clase Materia
Revisa asignados	Clase Curso

La Tabla 3.9 muestra la tarjeta CRC correspondiente a la clase Curso que tiene a las clases docente y horario como clases colaboradoras, la responsabilidad, revisa docentes asignados, hace referencia a que la clase curso debe obtener a los docentes que pertenecen a un determinado curso.

Tabla 3.9 Tarjeta CRC para la clase Curso

Curso	
Revisa docentes asignados	Clase Docente
Obtiene horario del curso	Clase Horario
Obtiene cursos y paralelos	

La tarjeta CRC de la clase Inscripción, esta representada en la Tabla 3.10, donde se puede ver las clases colaboradoras Estudiante y Curso, a partir de la clase Inscripción se puede obtener los inscritos en una determinada gestión, tomando en cuenta la responsabilidad Obtiene estudiantes inscritos.

Tabla 3.10 Tarjeta CRC para la clase Inscripción

Inscripción	
Registrar datos de la gestión actual	Clase Estudiante
Obtiene estudiantes inscritos	Clase Curso
Revisa si se realizo la inscripción	

Las clases Curso y Docente son las clases colaboradoras para la clase Horario como se puede ver en la Tabla 3.11, la responsabilidad, asignar docentes a curso, tiene la clase Curso como colaboradora la cual debe proveer los cursos disponibles para la asignación de docentes.

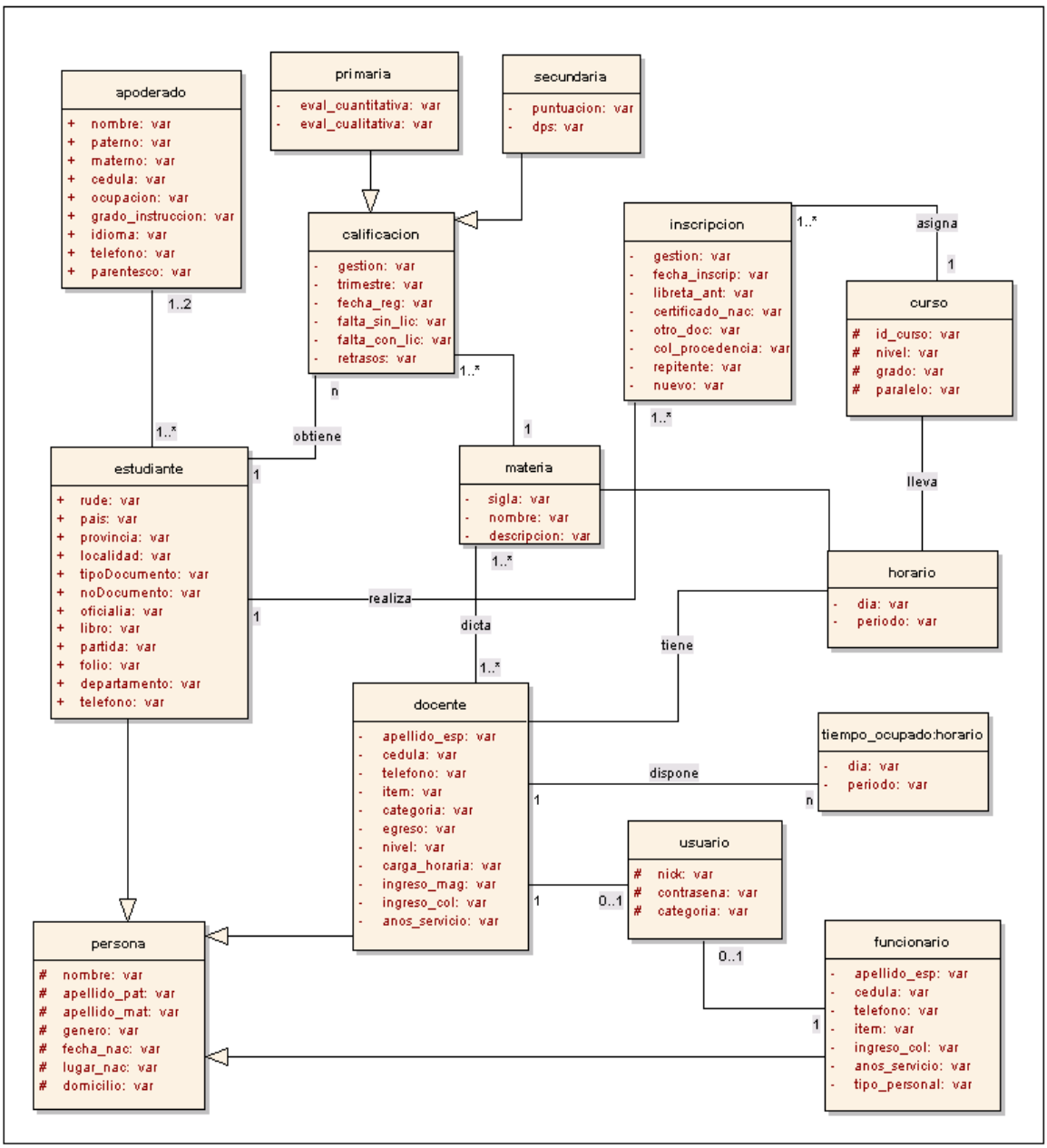
Tabla 3.11 Tarjeta CRC para la clase Horario

Horario	
Asignar docentes a curso	Clase Curso
Obtiene docentes disponibles	Clase Docente
Devuelve cursos asignados	Clase Docente Clase Curso

3.2.3 Modelo Objeto-Relacion

Luego de haber identificado las tarjetas CRC candidatas, se tiene una idea de las clases necesarias para el desarrollo del sistema, al identificar las tarjetas CRC también se llega tener una idea mas clara de las operaciones que las clases tendrán que implementar, y de las clases con las que estarán relacionadas.

Figura 3.3 Relaciones entre clases



Esto nos lleva a identificar las relaciones entre clases multiplicidades y jerarquías entre ellas, como podemos observar en la Figura 3.3 que nos muestra el diagrama de relaciones entre las clases candidatas del Sistema de Planificación y Seguimiento Académico.

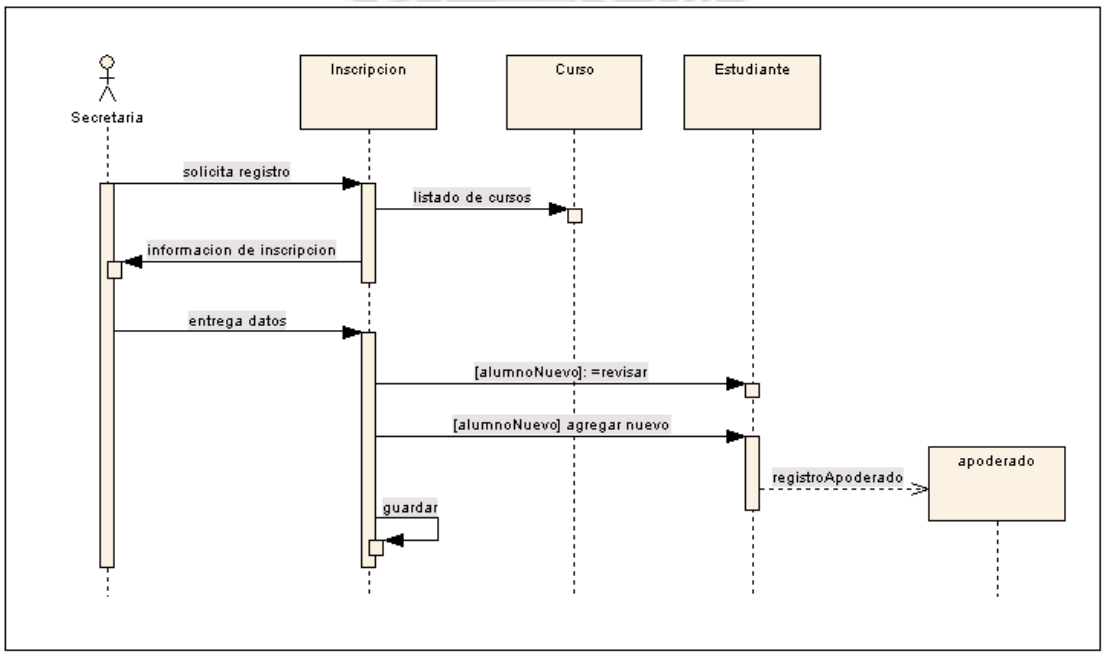
3.2.4 Modelo Objeto-Comportamiento

Para realizar el Modelo Objeto-Comportamiento, en el cual se ve el comportamiento de lo diferentes objetos del sistema, se utilizaran los Diagramas de Secuencia y de Colaboracion**,** como se menciono en la seccion 2.3.8

3.2.4.1 Diagrama de Secuencia

Habiendo identificado las relaciones y jerarquías entre clases con el diagrama de clases propuesto en la Figura 3.3 y analizando las tarjetas CRC vistas en la sección 3.2.2, se procede a realizar los diagramas de secuencia de los casos de uso, los cuales muestran las interacciones entre objetos vistas en el tiempo de vida de un objeto.

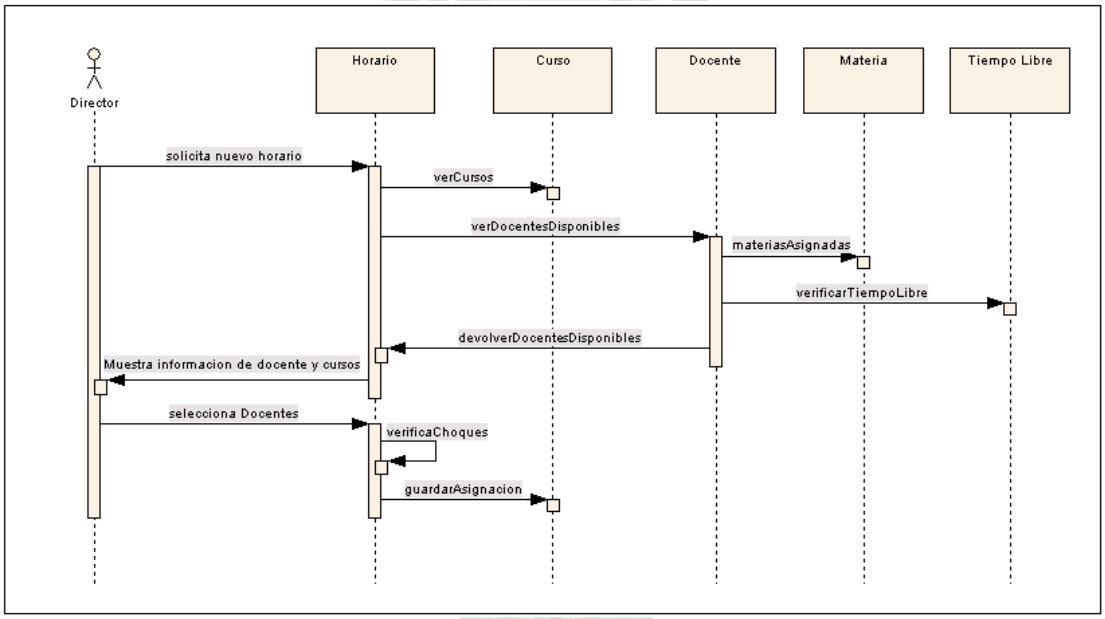
Figura 3.4 Diagrama de secuencia Registra inscripción



La Figura 3.4 muestra la interacción entre objetos que intervienen en el caso de uso de inscripciones, se puede apreciar que cuando un alumno es nuevo en el colegio debe registrar los datos del apoderado para realizar su inscripción.

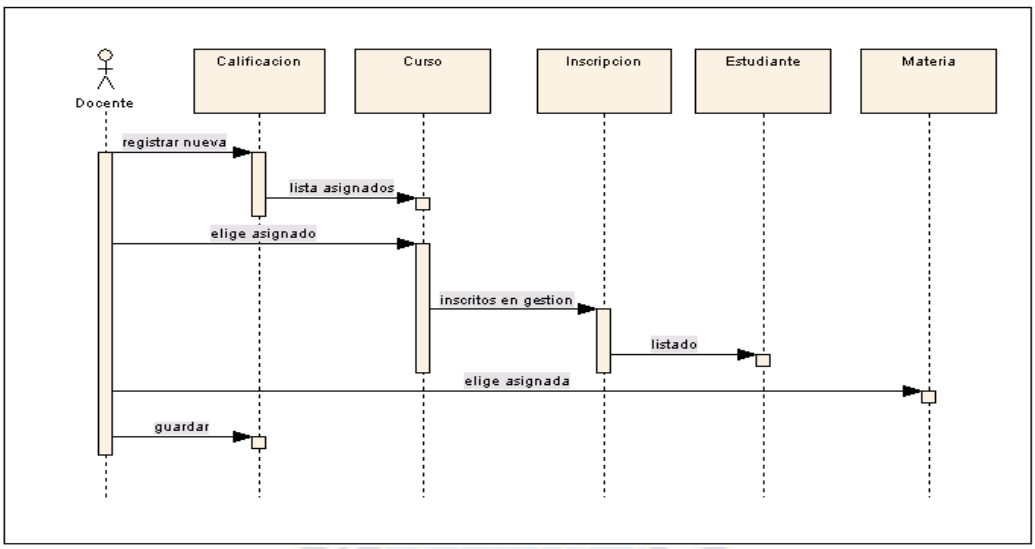
La Figura 3.5 muestra el diagrama de secuencia para el caso de uso planifica horarios, se puede apreciar que para realizar la asignación de aulas primero se debe verificar la disponibilidad de tiempo de los docentes, y después de elegir los docentes se realiza una verificación de posibles choques que podrían existir, para tener una idea mas completa se puede ver también la Tabla 3.4.

Figura 3.5 Diagrama de secuencia Planifica Horarios



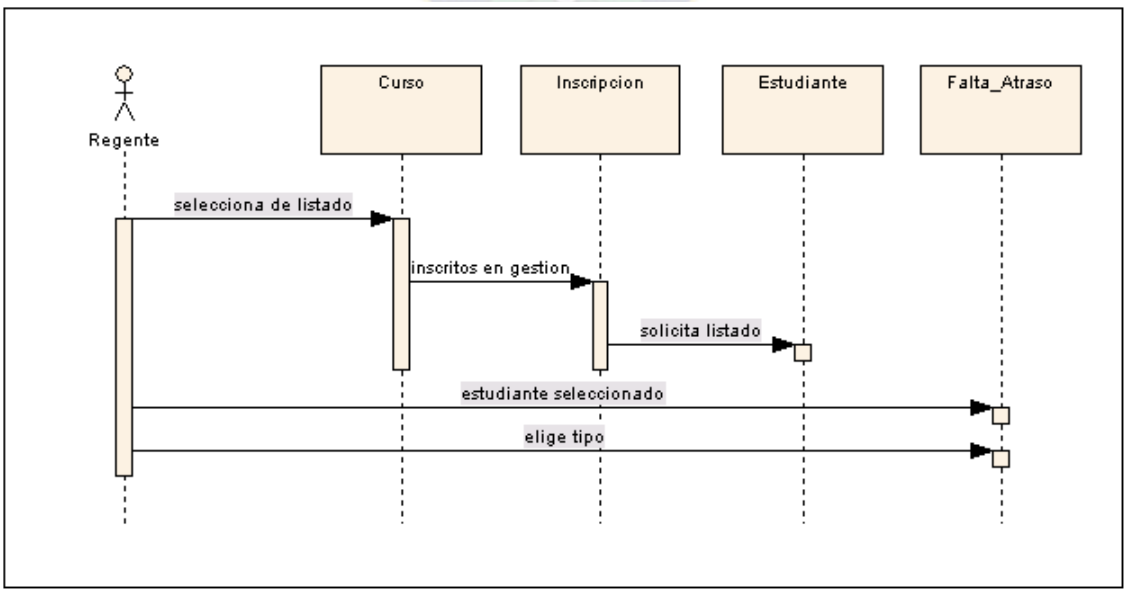
El diagrama de secuencia para el caso de uso Actualizar Calificación se encuentra en la Figura 3.6, donde se pueden observar como interactúan los objetos que intervienen en dicho caso de uso.

Figura 3.6 Diagrama de secuencia actualiza calificación



La Figura 3.7 ilustra el diagrama de secuencia del caso de uso Registro de faltas o atrasos, donde se ve que un objeto estudiante esta relacionado con un curso y para realizar el registro de una falta o atraso se accede primero al objeto curso.

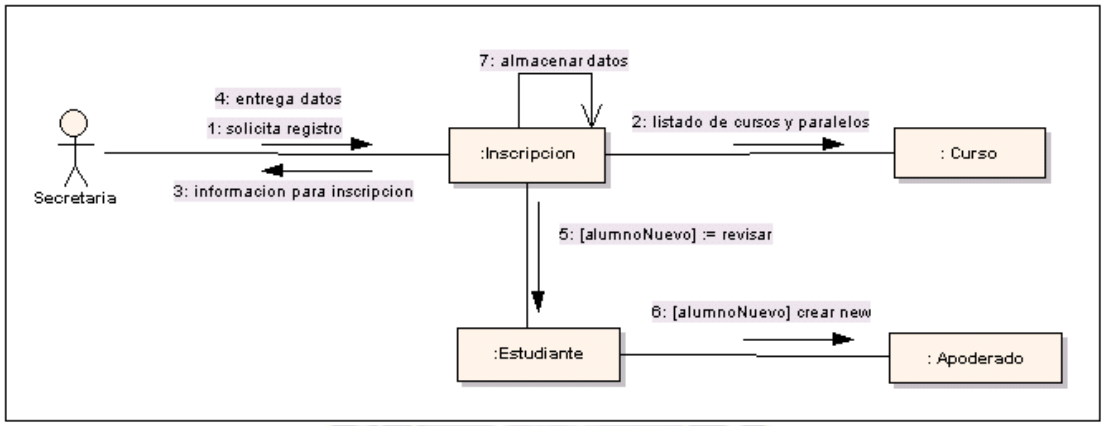
Figura 3.7 Diagrama de secuencia Registro faltas o atrasos



3.2.4.2 Diagrama de Colaboración

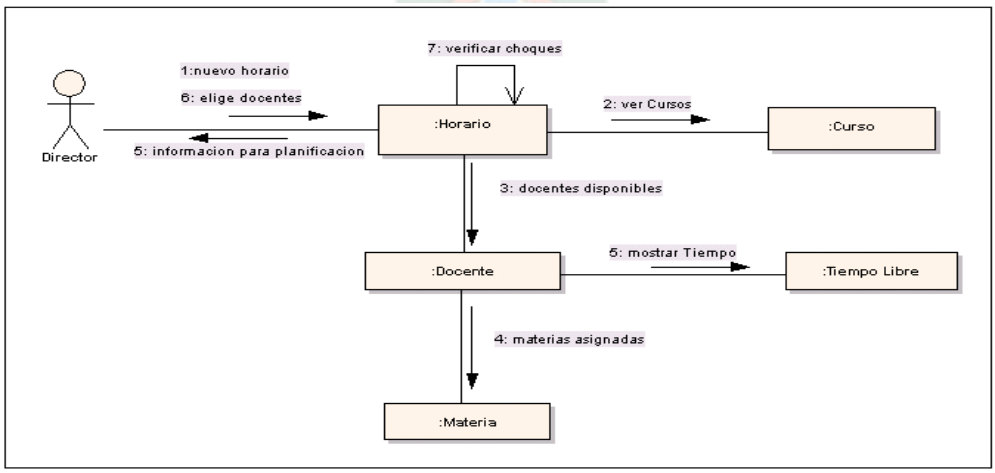
Otra forma de ver como interactúan los objetos en el sistema es con los diagramas de colaboración. La Figura 3.8 representa el diagrama de colaboración Registra inscripción, el cual muestra los objetos involucrados para el caso de uso, se puede ver la relación directa entre estudiante y apoderado cuando un alumno es nuevo

Figura 3.8 Diagrama de colaboración Registra Inscripción



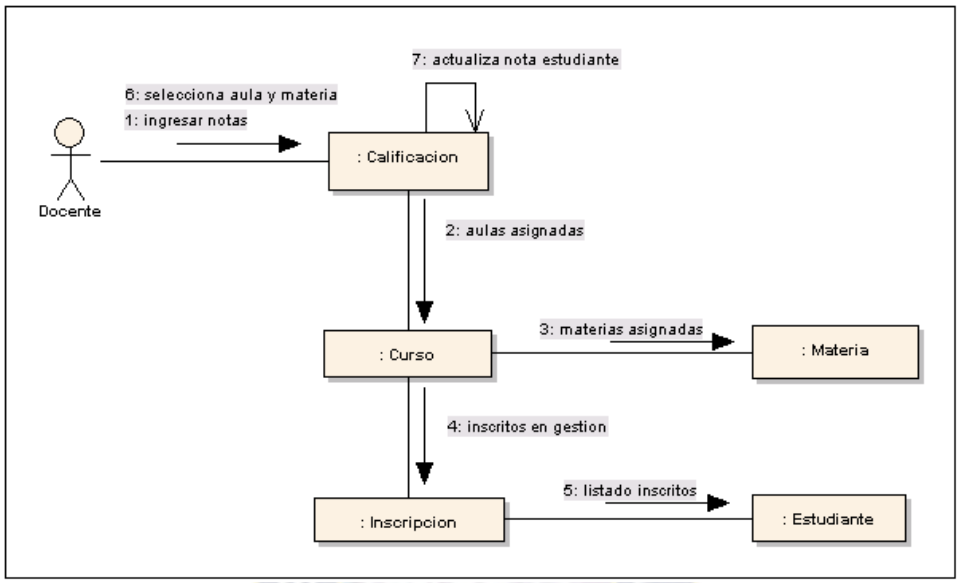
.En el diagrama de colaboración Planifica Horarios, mostrado en la Figura 3.9 se puede apreciar la relación entre Docente y Tiempo Libre, esta relación se da cuando se desea saber la disponibilidad de un docente, para el caso de uso es necesario tener esta información puesto que sin esta información no se llegaría a tener los datos de los docentes que pueden ser asignados a un determinado día y periodo.

Figura 3.9 Diagrama de colaboración Planifica Horarios



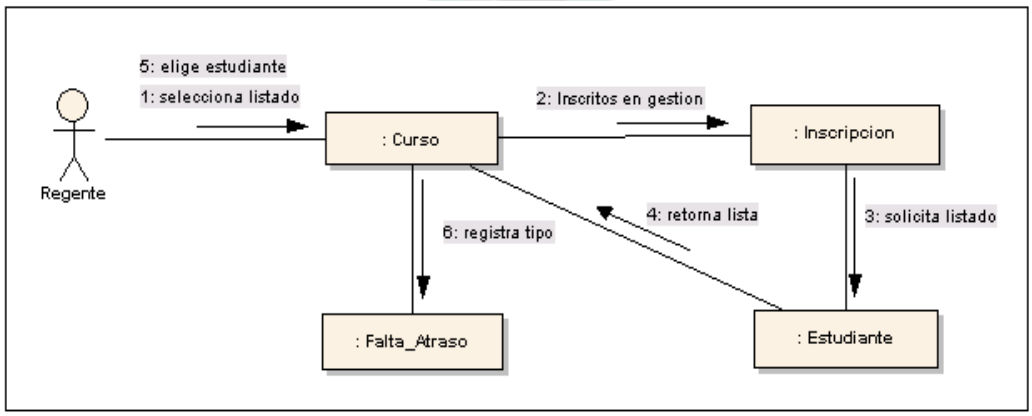
El diagrama de colaboración de la Figura 3.10 para el caso Actualiza Calificación se puede observar la relación entre Curso y Materia,

Figura 3.10 Diagrama de colaboración Actualiza Calificación



La Figura 3.11 ilustra el diagrama de colaboración para Registro de faltas o atrasos, se puede observar como interactúan los objetos curso, estudiante, inscripción para poder registrar una falta o atraso de un estudiante.

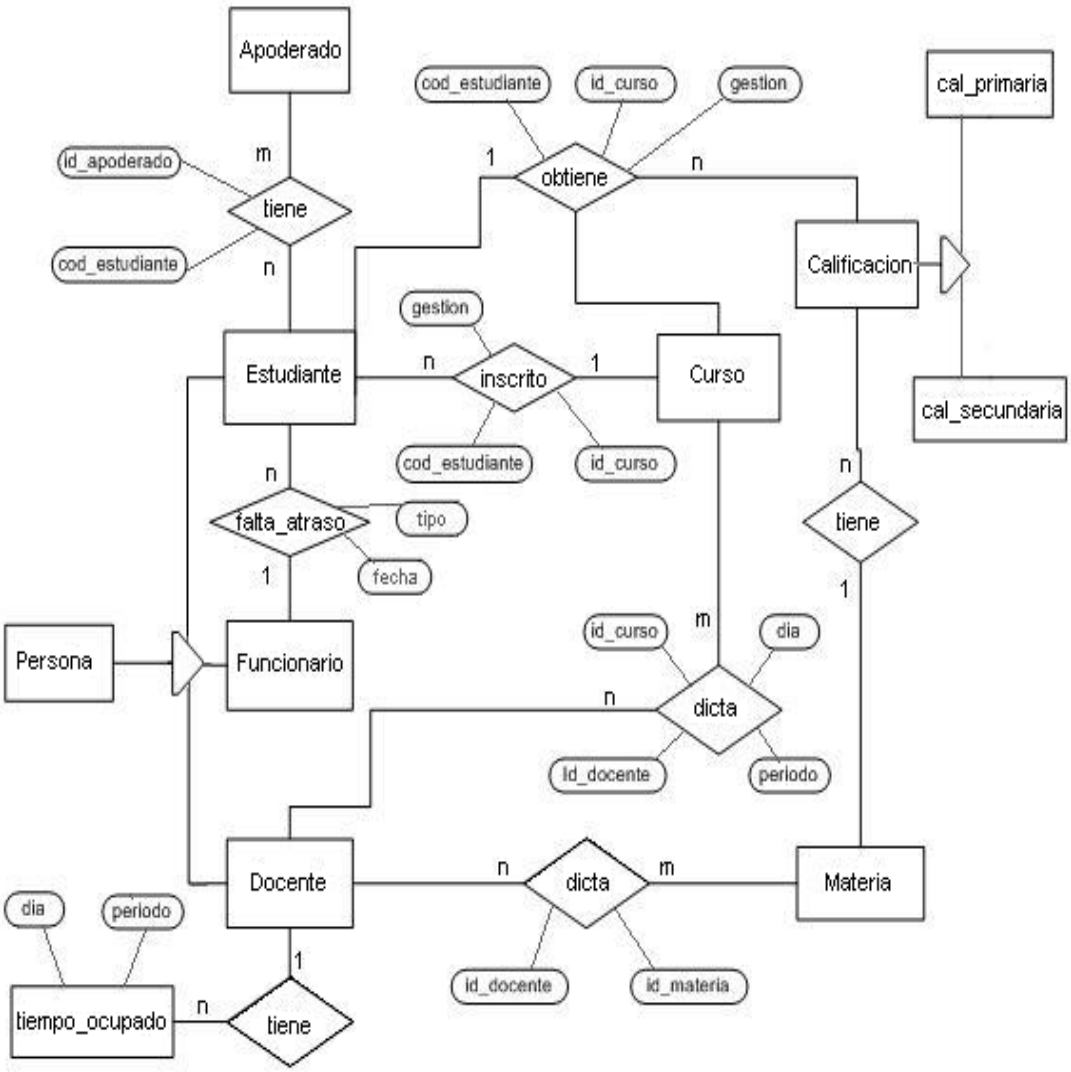
Figura 3.11 Diagrama de colaboración Registro de faltas o atrasos



3.2.5 Modelo Entidad Relacion

Con la ayuda del diagrama de clases de la Figura 3.3, se puede evidenciar las relaciones entre objetos del dominio y es posible construir el diagrama Entidad Relación (ver Figura 3.12), para modelar la base de datos del sistema.

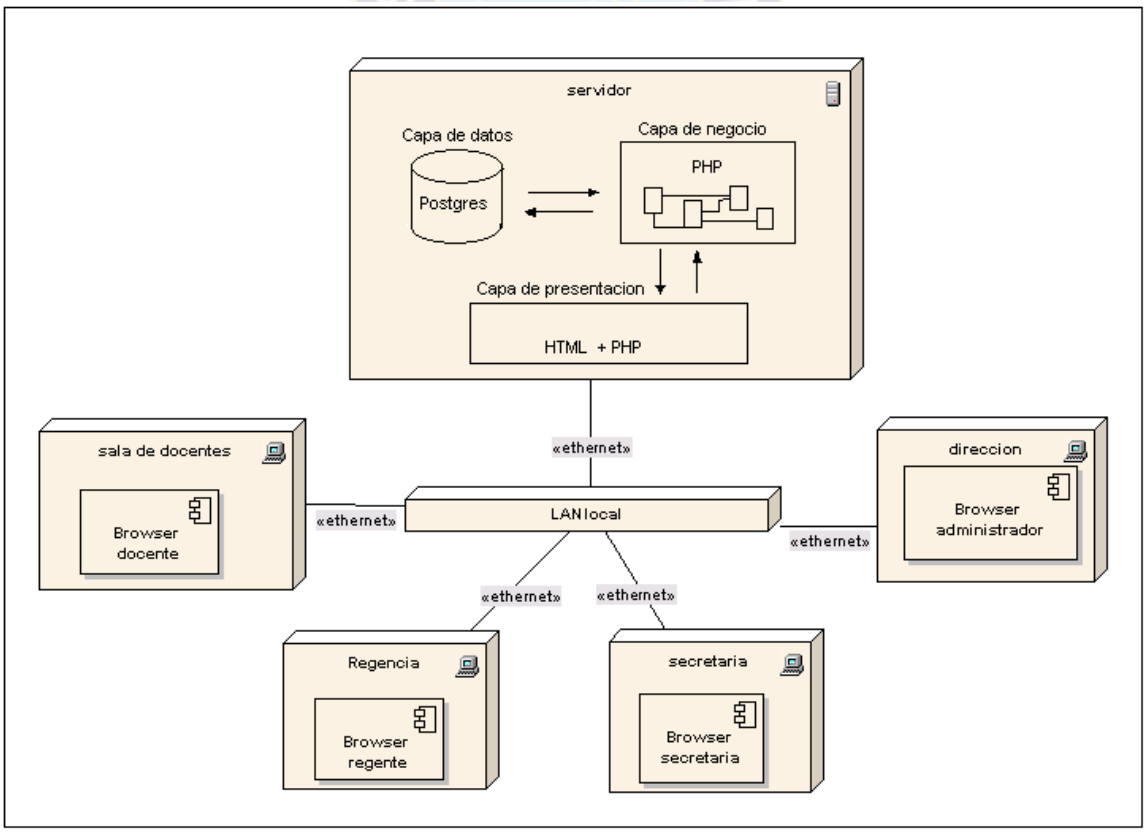
Figura 3.12 Diagrama Entidad Relación



3.2.6 Diseño del Sistema

En la etapa del diseño del sistema se debe definir la arquitectura del mismo como se menciono en la sección 2.5.1, para lo cual se tomará en cuenta el patrón de arquitectura de software MVC (Modelo Vista Controlador), puesto que es frecuentemente utilizado en aplicaciones web. El patrón MVC separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos [WikMVC, 2009]. Para el caso del proyecto desarrollado, las Vistas serian las paginas html y código PHP que provee datos dinámicos a las mismas, los modelos seran las clases implementadas en PHP que manipularan los datos y el acceso a la base de datos del sistema y el controlador las acciones del usuario como ser la navegación por las paginas, clics en los enlaces y botones, etc. La Figura 3.13 ilustra la arquitectura propuesta para el sistema.

Figura 3.13 Arquitectura Propuesta para el Sistema



3.2.7 Diseño de Objetos

Para el diseño de objetos se tomara en cuenta la descripción de los mensajes que cada objeto puede recibir y puede enviar, dando una descripción breve del objetivo del mensaje como se menciono en la sección 2.5.2 se realizara solo la descripción del protocolo y no así la descripción de implementación.

Tomando los mensajes de la Figura 3.8 se hará la descripción de cada protocolo como sigue:

- MENSAJE (estudiante) -> revisaExistencia: DEVUELVE valor bool// mensaje que devuelve falso o verdadero dependiendo si un estudiante es antiguo..
- MENSAJE (curso) -> curso_paralelo: DEVUELVE cod_curso // mensaje que muestra los cursos y paralelos disponibles.
- MENSAJE (apoderado) -> registrarNuevo: GUARDA id_apoderado, parentesco, nombre y apellido, cedula, ocupación // almacena los datos personales de un apoderado.
- MENSAJE (inscripcion) -> almacenarDatos: GUARDA cod_estudiante, id_curso, gestión // mensaje que almacena los datos de inscripción de un estudiante inscrito en un curso para una determinada gestión, clases colaboradora curso; estudiante.

Viendo los mensajes de la Figura 3.9 que corresponde al diagrama de colaboración Planifica Horarios se realizara la descripción de los protocolos.

- MENSAJE (horario) -> elegirDocentes: MUESTRA docentes, cursos, dia, periodo; //mensaje requerido para capturar la asignación de docentes a cursos, necesita de clases colaboradoras docente, curso.
- MENSAJE (curso) -> curso_paralelo: DEVUELVE cod_curso // mensaje que muestra los cursos y paralelos disponibles.
- MENSAJE (docente) -> devolverLibres: DEVUELVE docentes; // devuelve a todos los docentes libres en un determinado nivel con los valores de día y periodo establecidos, clase colaboradora horario.
- MENSAJE (materia) -> listaAsignadas: MUESTRA sigla; // mensaje que devuelve un listado de materias asignadas a un docente.

Haciendo referencia a los mensajes de la Figura 3.10 que corresponde al diagrama de colaboración actualizar calificación se realizara la descripción de los protocolos.

- MENSAJE (calificacion) -> ingresarNota: GUARDA id_curso, cod_estudiante, puntos, dps, trimestre; //mensaje requerido para capturar datos de la nota de un estudiante para un determinado trimestre.
- MENSAJE (curso) -> aulasAsignadas: DEVUELVE id_curso, // mensaje requerido para saber los cursos de un docente, clase colaboradora docente.
- MENSAJE (materia) -> listaAsignadas: MUESTRA sigla; // mensaje que devuelve un listado de materias asignadas a un docente.
- MENSAJE (inscripcion) -> inscritosGestion: DEVUELVE cod_estudiante; // mensaje que devuelve los alumnos inscritos en una gestión, clase colaboradora estudiante.

En la Figura 3.14 se puede observar el diagrama de clases completo donde cada clase tiene sus atributos y métodos definidos para ser llevados a la implementación.

3.2.8 Diagrama de Contexto Navegacional

Los diagramas de contexto navegacional muestran la información que puede ser vista por un determinado usuario, y a las posibilidades de manipulación de la información accedida. La Figura 3.15 muestra el contexto navegacional del usuario tipo Secretaria, se puede ver que tiene acceso a la información del Kardex docente y puede agregar registros de este tipo y no así realizar la modificación o eliminación de los mimos. El diagrama visto es resultante del modelo de casos de uso y del Modelo Objeto-Relación, el cual indica que clases están relacionadas, y los casos de uso donde se establece que información puede ser accedida por un usuario particular.

Figura 3.14 Diagrama de Clases Sistema de Planificación y Seguimiento Académico

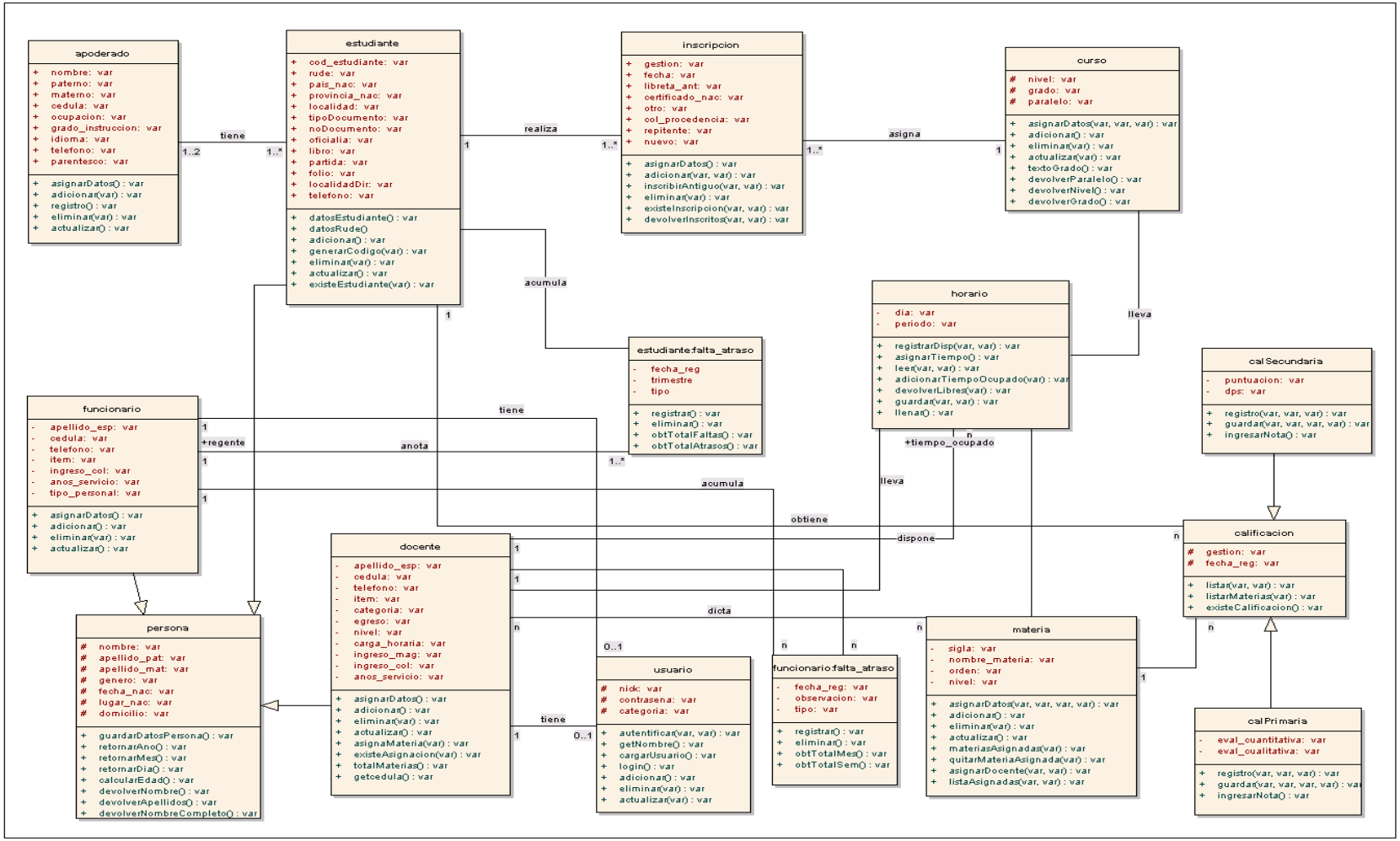
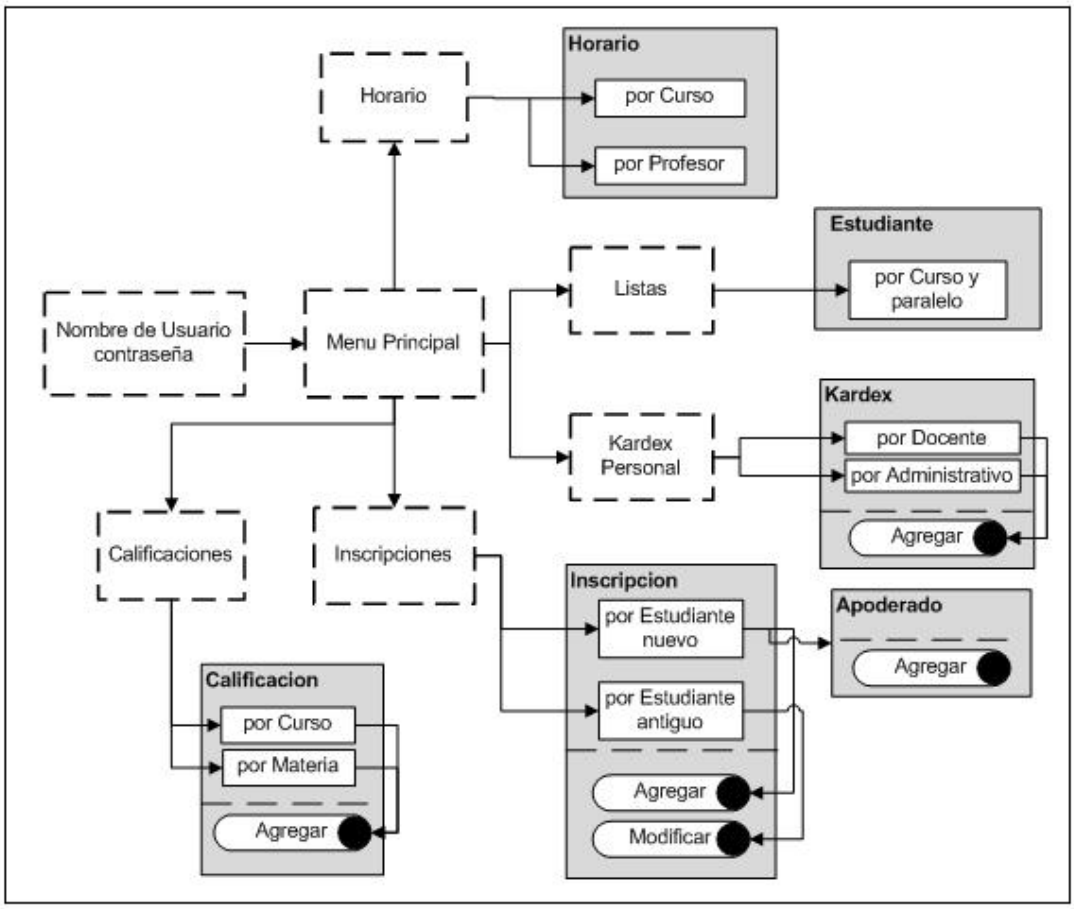
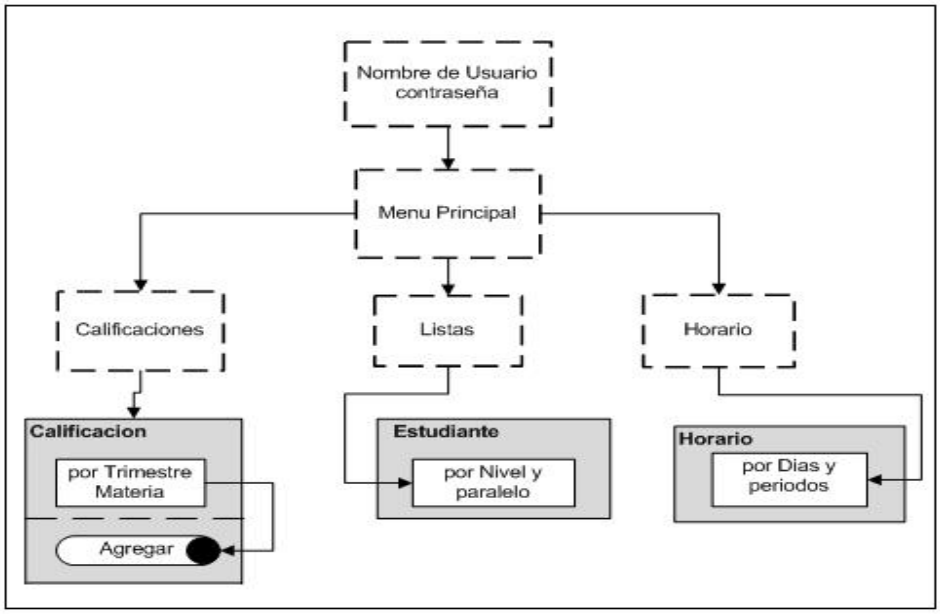


Figura 3.15 Contexto Navegacional Secretaria



La Figura 3.16 ilustra el contexto navegacional de un docente, se puede ver que la información a la que accede es solo de carácter informativo, y solo en el caso de calificaciones tiene posibilidad de ingresar las notas de los estudiantes.

Figura 3.16 Contexto Navegacional Docente



El contexto navegacional del usuario Director es como se ve en la Figura 3.17, donde se puede evidenciar que tiene muchas mas posibilidades de manipular la información referente al sistema, entre las cuales podemos ver la gestión de cursos, materias y docentes, esto para la planificación de los horarios, esta información no puede ser manipulada por otros usuarios.

La Figura 3.18 muestra el contexto navegacional del usuario regente, al igual que el usuario docente la información que tiene para poder navegar no puede ser modificada por el mismo, excepto el registro de faltas o atrasos de los estudiantes.

Figura 3.17 Contexto Navegacional Director

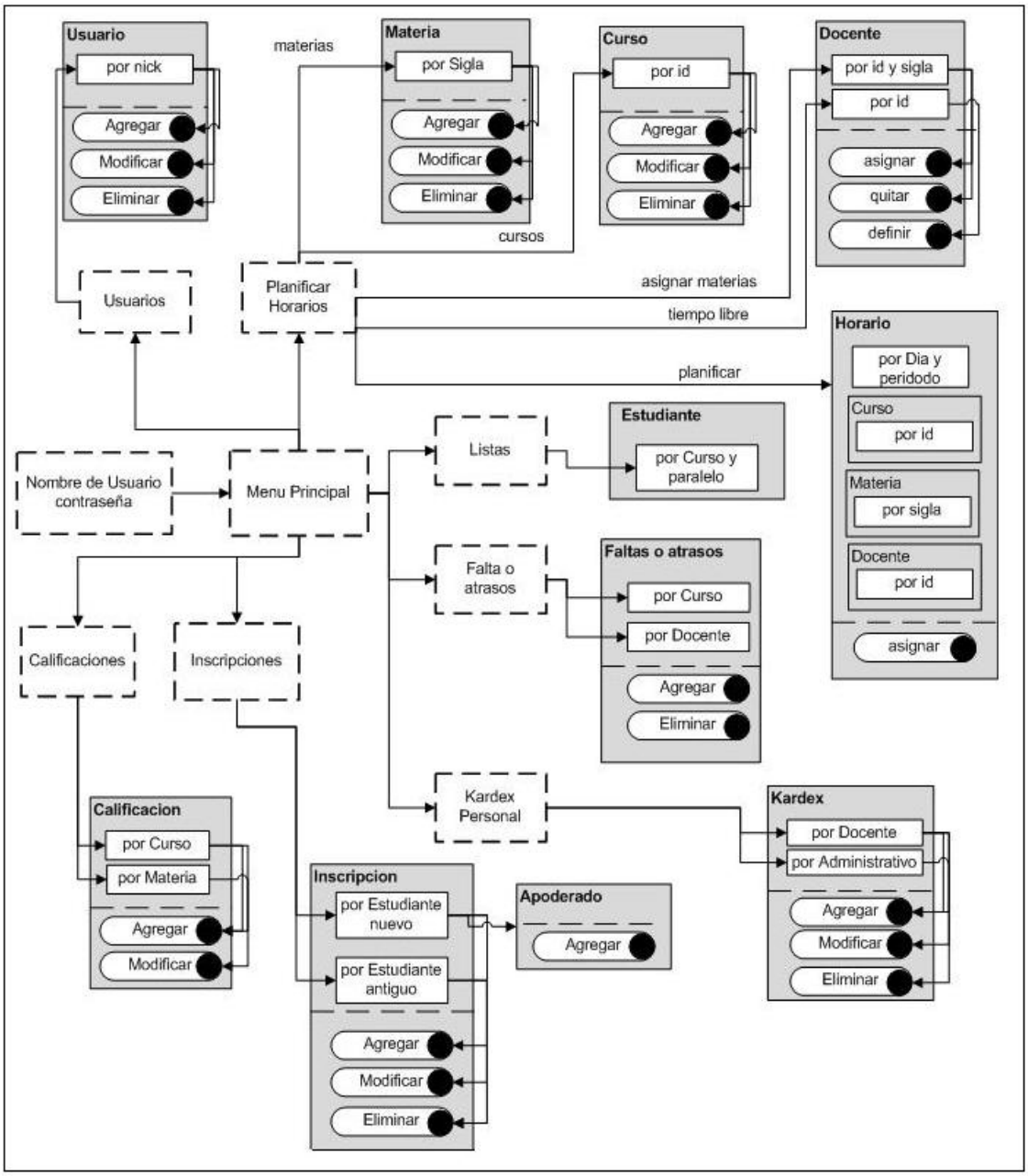
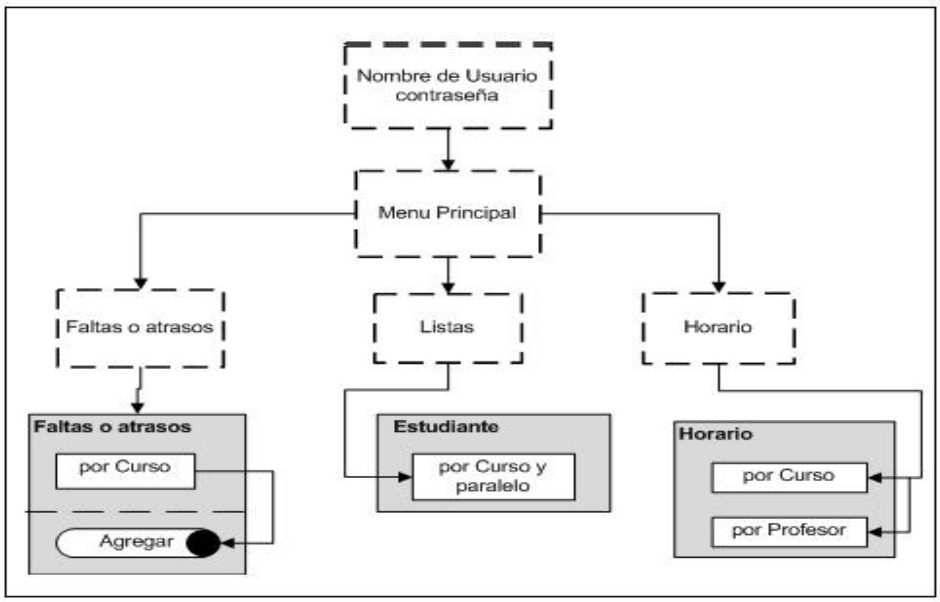


Figura 3.18 Contexto Navegacional Regente



3.2.9 Diseño de la Interfaz Abstracta

Las ADV's (Diseño de Interfaz Abstracta) especifican las interfaces de objetos que los usuarios percibirán, la Figura 3.19 muestra la ADV de ingreso al sistema, que se vera en la pagina principal.

Figura 3.19 ADV ingreso de usuarios

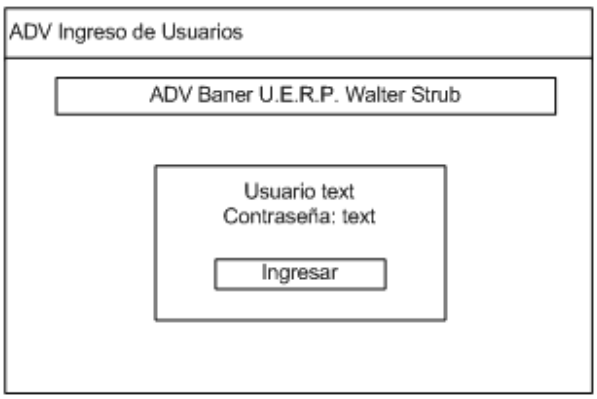
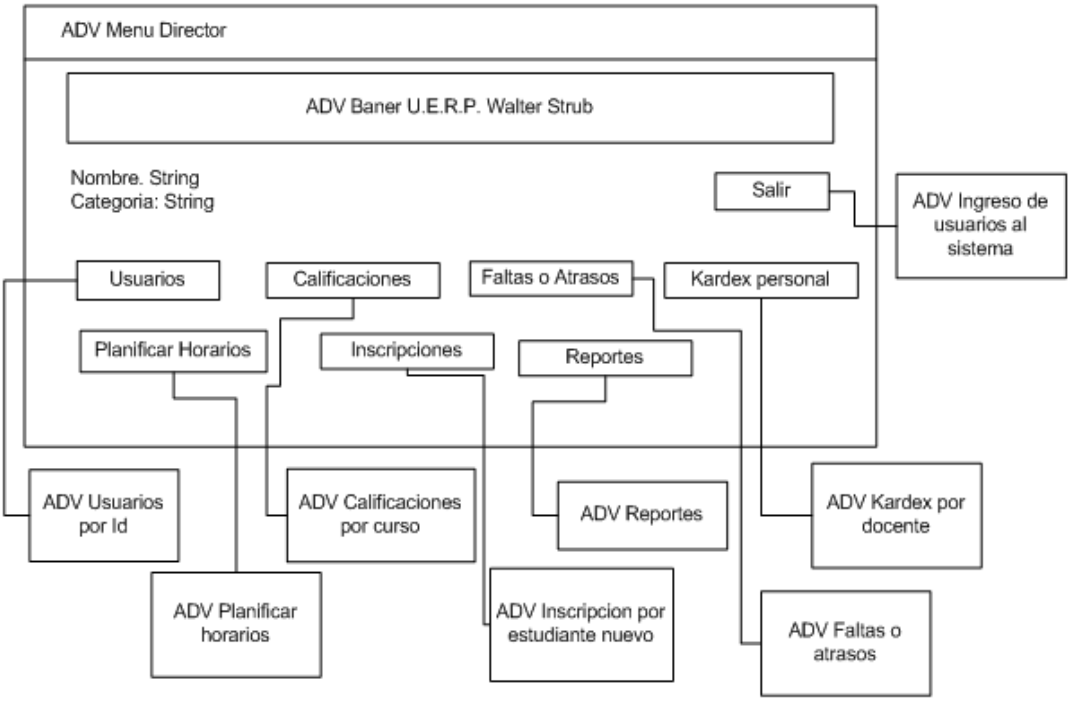
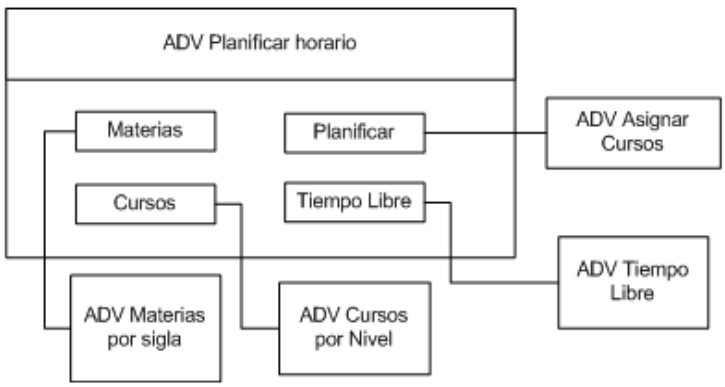


Figura 3.20 ADV Menú Director



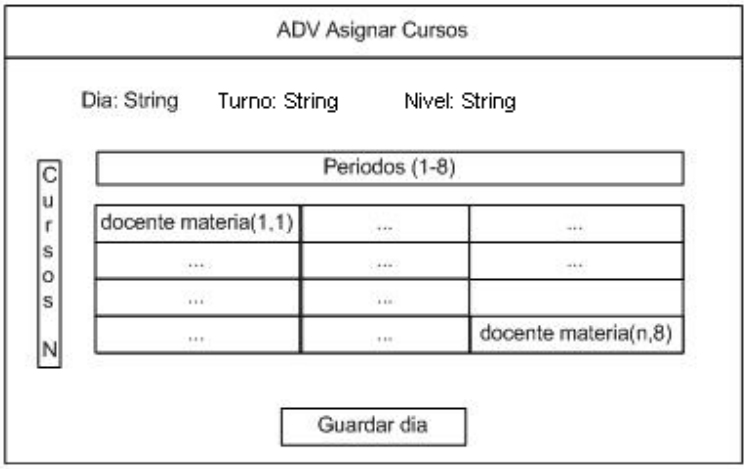
La Figura 3.20 ilustra la ADV del menú que tiene el usuario Director, donde se ven los enlaces a otros ADV's, como ser usuarios, planificar, horarios entre otros. La ADV Planificar Horario vista en la Figura 3.21, contiene enlaces a otras ADV's, como ser materias, cursos, esta ADV hace referencia a una parte del contexto navegacional vista en la Figura 3.17.

Figura 3.21 ADV Planificar horario



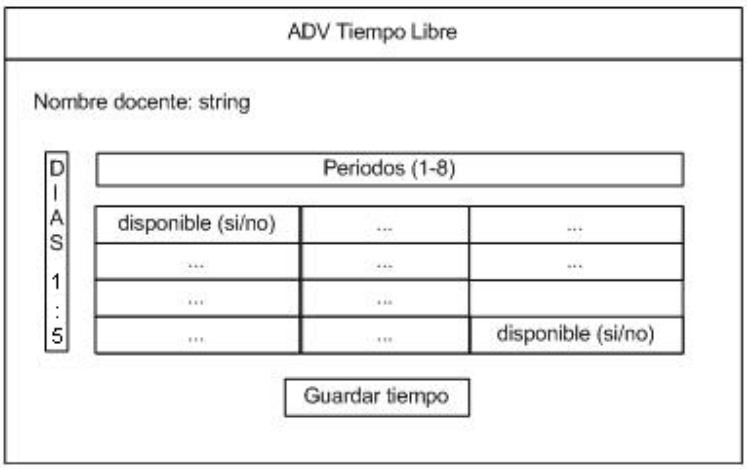
La Figura 3.22 muestra la ADV asignar cursos, donde se proporciona la información de los docentes disponibles y materias que pueden dictar cada uno de ellos, estos datos están dispuestos en una matriz que indicara los cursos y periodos para un determinado día.

Figura 3.22 ADV Asignar Cursos



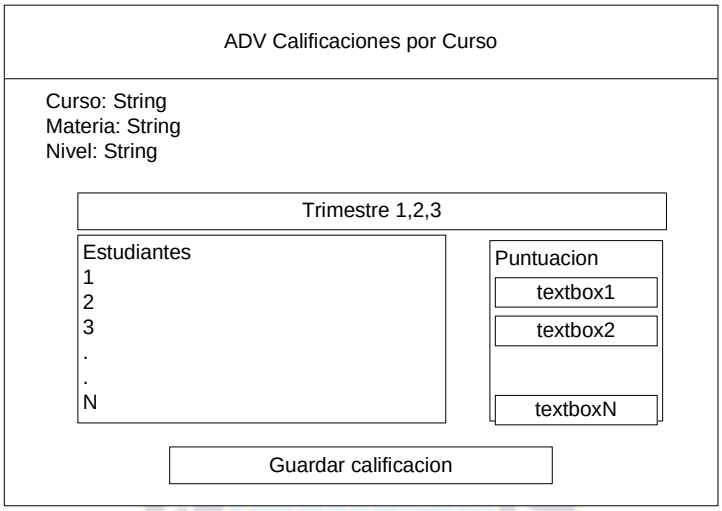
La ADV Tiempo Libre mostrada en la Figura 3.23, proporciona datos sobre un docente determinado y los días y periodos que tiene tiempo disponible, esta ADV será solo accedida desde la ADV menú Director.

Figura 3.23 ADV Tiempo Libre



La ADV Calificaciones por Curso vista en la Figura 3.24, muestra los datos sobre curso, materia y estudiantes, y la puntuación para un determinado trimestre.

Figura 3.24 ADV Calificaciones por curso



3.3 Implementación

Una vez realizado el análisis y diseño del sistema se procede a implementar el código, que es el resultado de los diferentes modelos de análisis y diseño que se construyeron. En las líneas de código que se muestran a continuación se podrá ver la clara separación de la parte HTML de la lógica de la aplicación, lo cual lleva a un código limpio y entendible.

La porción de código siguiente, muestra como interactúan los objetos de tipo materia y de tipo docente, la clase materia tiene la responsabilidad de listar las materias de un profesor, donde la clase docente seria la colaboradora.

```
<?

//funcion propia de php5 que sirve para cargar clases automaticamente
function __autoload($class_name){
      include "../modelos/$class_name.php";
}

$visitante = new usuario();
if($visitante->cargarUsuario()){
    $visitante->opciones();

menu(array("listarMaterias.php","Materias","listarCursos.php",..));
```

```
    // se crean dos objetos uno de materia y otro de docentes
    $materias=new materia();
    $docente1= new docente();

    $docente1->asignar($_GET["cod"]);
    $nombreCompleto=$docente1->devolverNombreCompleto();
    // interaccion entre docente y materia
    $materias->listaAsignadas($nombreCompleto,$docente1->obtCedula());


}
else $visitante->mostrar();

?>
```

La siguiente porción de código es la que realiza la inscripción de un estudiante antiguo, se puede ver como interactúan los objetos del tipo estudiante e inscripción, la clase colaboradora estudiante interactúa con la clase inscripción que llega a ser la clase responsable.

```
    $antiguo= new inscripcion();
    if (!isset($_POST['cod_estudiante']))
    {
        $antiguo->registro('1');
    }
    else
    {
        $est_antiguo=new estudiante();
        $codigo=$_POST['cod_estudiante'];
        $codigo=strtoupper($codigo);
        $gestion=date('Y');
       // metodos que devuelven valores tipo bool correspondientes a las
      //clases estudiante e inscripción respectivamente
        if($est_antiguo->existeEstudiante($codigo)&&
            !$antiguo->existe($gestion,$codigo))
        {
            $fecha= date("Y-m-d");
            $ins=$_POST['tipoInscripcion'];
            if($ins=='nuevo'){
                $nuevo='si';$rep='';
            }
            else{
                $nuevo='';$rep='si';
            }
            $antiguo->asignarDatos($gestion,  $fecha,  $_POST['libreta'],
            $_POST['certificado'], $_POST['otrodoc'], $_POST['nombreUE'],
            $rep, $nuevo);

            $antiguo->inscribirAntiguo($_POST['grado'],$codigo);
            mensaje("la Inscripcion termino con Exito");
```

```
        }
        else
            mensaje("no existe el codigo de estudiante o el estudiante ya
            esta inscrito en la gestion $gestion");

    }
```

Interfaces implementadas

En las siguientes figuras se vera una parte de las interfaces web implementadas para el Sistema de Planificacion y Seguimiento Academico. La Figura 3.25 muestra el formulario de registro del personal, donde se debe seleccionar el tipo de personal para que el sistema seleccione la tabla correspondiente en la base de datos.

Figura 3.25 Interfaz de registro de docentes y administrativos

En la Figura 3.26 se muestra la interfaz para la asignacion de cursos a docentes, donde se tiene los docentes acomodados en cada lista por orden de carga horaria, donde los primeros en la lista tienen mayor carga horaria, tambien se toma en cuenta la disponibilidad de tiempo de los mismos, al realizar la selección de los docentes el sistema verificara que un determinado docente no sea asignado en diferentes cursos en un mismo periodo.

Figura 3.26 Interfaz para asignación de docentes a cursos

Fuente: [Elaboración Propia]

En la Figura 3.27 se ve el formulario para ingreso de notas de nivel Primaria, donde se tiene campos para ingresar las notas cuantitativas para cada alumno y cualitativas que son para un grupo correspondiente a las notas cuantitativas dependiendo del rango en que se encuentre.

Figura 3.27 Interfaz ingreso de calificaciones Primaria

3.4 Pruebas de Integración basadas en el uso

Como menciona [Pressman, 2003], para realizar las pruebas de integracion se debe hacer una revision al Modelo CRC y al Modelo Objeto relacion, esto para tener un conjunto de clases colaboradoras para poder realizar las pruebas. De este modo se realizaran los casos de pruebas. La Tabla 3.12 describe el caso de prueba para saber las materias asignadas de un docente.

Tabla 3.12 Caso de Prueba Materias asignadas a un Docente

Caso de Prueba :	Materias asignadas que tiene un docente
Clases implicadas:	Docente, Materia
Propósito	Tener conocimiento de las materias que puede dictar un docente
Pasos a Seguir	Se debe tener un identificador del docente Se deben tener materias asociadas al docente Se debe pasar un mensaje a la clase materia con el identificador del docente. La clase materia debe responder con un listado de las materias del docente

Tabla 3.13 Caso de Prueba Docentes disponibles

Caso de Prueba :	Docentes disponibles en un determinado día
Clases implicadas:	Docente, Horario
Propósito	Saber que docentes están disponibles en un determinado dia
Pasos a Seguir	Se debe pasar un mensaje a la clase Horario con datos del nivel, día y periodo, a verificar, como se ve en el siguiente mensaje $horariolunes->leer("LUNES",1); $horariolunes->devolverLibres("PRIMARIA"); La clase Horario debe responder con un listado de las Docentes disponibles en dicho día y periodo.

La Tabla 3.13 narra el caso de prueba para obtener docentes disponibles para un día y periodo especifico. En la Tabla 3.14 se describe el caso de prueba para obtener alumnos inscritos en una gestión.

Tabla 3.14 Caso de Prueba Inscritos en gestión

Caso de Prueba :	Estudiantes inscritos en una gestión
Clases implicadas:	Inscripcion, Curso, Estudiante
Propósito	Saber que estudiantes están inscritos en un curso en determinada gestión
Pasos a Seguir	Se debe tener un identificador del curso Se deben tener estudiantes inscritos Se debe pasar un mensaje a la clase Inscripcion con datos de la gestión y curso que se desea saber, como se ve en el mensaje $gestion_actual->devolverInscritos($gestion,$curso) La clase Inscripción debe responder con un datos de los Estudiantes que están inscritos en gestión y curso..

En la Tabla 3.15 se describe el caso de prueba para el registro de falta o atraso de un estudiante elegido.

Tabla 3.15 Caso de Prueba anotar falta estudiante

Caso de Prueba	Estudiantes inscritos en una gestión
Clases implicadas:	Estudiante, falta_atraso
Propósito	Verificar el registro de una falta o atraso
Pasos a Seguir	Se debe tener el identificador del Estudiante Se debe pasar un mensaje a la clase falta_atraso con datos del trimestre, tipo y el identificador del alumno, como se ve $anotar->asignarDatos($trimestre,$tipo,$observacion); $anotar->guardar($alumno); La clase falta_atraso debe guardar los el dato del Estudiante y los demás datos en fecha que se realiza el registro

3.5 Calidad de Software

La calidad de software es el “Grado con el cual el cliente o usuario percibe que el software satisface sus expectativas” [Fuertes, 2008].

Los requisitos del software son la base de las medidas de la calidad. La falta de concordancia con los requisitos funcionales es una falta de calidad. El Software se puede medir directa e indirectamente. La calidad está compuesta por una composición de muchos atributos tanto internos como externos, para medir la calidad del Sistema de Planificación y Seguimiento Académico se tomaron en cuenta los atributos de Funcionalidad, Confiabilidad, Usabilidad y Portabilidad.

3.5.1 Funcionalidad

La métrica de Punto función es una métrica del modelo de análisis. Las métricas del software orientadas a la función utilizan una medida de la funcionalidad entregada por la aplicación como un valor de normalización. Ya que la funcionalidad no se puede medir directamente, se debe derivar indirectamente mediante otras medidas directas como se menciono en la sección 2.9.2.

Número de entradas de usuario

Se contaron todos las entradas de usuario para las principales funciones del sistema, se tomo en cuenta la entradas que realmente afectan a una determinada función como indica [Braude, 2003], también no se toma en cuanta la parte de usuarios ya constituye una función adicional a las del sistema.

- Kardex del personal, entrada de datos del funcionario o docente y entrada de tipo de personal, entrada de modificación de datos;
- Kardex de estudiantes, entrada de datos del estudiante y datos del apoderado, entrada de modificación de datos de estudiante y apoderado;
- Inscripción de estudiantes, entrada del curso y paralelo que le corresponde;
- Registro de materias, datos sobre la materia, nivel y orden, modificación de los datos;
- Registro de cursos, por nivel, grado y paralelo;
- Calificaciones, ingreso de calificaciones por curso, materia, modificación de las mismas;
- Faltas y atrasos de estudiantes, registro de falta o atraso de un estudiante por curso;

- Faltas y atrasos de docentes, registro de falta o atraso de un docente;
- Asignación de materias, asignación de una materia a un docente;
- Tiempo disponible de un docente, registra días y periodos para un docente;
- Asignación de cursos a docentes, por nivel primaria o secundaria, y para cada día de lunes a viernes.

Número de salidas de usuario

Se cuentan todas las pantallas informes o listados que son visibles por el usuario a continuación se detallan las mismas.

- Kardex del personal, listados de docentes y funcionarios;
- Inscripción de estudiantes, listados de estudiantes por curso para una gestión;
- Listados de materias por nivel, materias que tiene un docente;
- Cursos por nivel;
- Boletín de calificaciones por curso y por alumno;
- Faltas y atrasos acumulados de estudiantes;
- Faltas y atrasos acumulados de docentes;
- Selección de docentes disponibles por nivel, para un determinado día;
- Horarios por curso y por docente.

Número de peticiones de usuario

Para las peticiones de usuario se tomaron en cuenta las acciones que puede solicitar el usuario como ser altas, bajas, y cambios a la información referente de las funciones principales del sistema

- Guardar datos de un docente, actualizar datos y eliminar;
- Guardar datos de un funcionario, actualizar datos y eliminar;
- Adicionar curso, modificar y eliminar;
- Adicionar materias, modificar y eliminar;
- Estudiantes y apoderados, adicionar, modificar datos;
- Guardar y eliminar materias que dicta un docente, guardar y modificar tiempo disponible de un docente;
- Guardar y eliminar la asignación de un docente a un curso;
- Registrar atrasos de docente y estudiantes, eliminar registros.

Número de archivos

Se toma en cuenta la base de datos y cada tabla cuenta como un archivo, los tablas que se forman a partir de la combinación de las tablas principales no se toman en cuenta.

- Registro de Docentes;
- Registro de Funcionarios;
- Registro de Estudiantes;
- Registro de Apoderados;
- Registro de Materias;
- Registro de Cursos;
- Registro de Calificaciones;
- Registro de Inscripciones;
- Registro de Horarios;
- Registro de Faltas o Atrasos.
- Registro de Disponibilidad docente

Número de interfaces externas

Como interfaces externas se tomaron en cuenta la conexión con al base de datos, la pantalla o monitor, la interface propia del visualizador de archivos PDF, y la elaboración de plantillas para calificaciones en formato Excel, puesto que el sistema no se comunica con otros sistema solo se tomaron estas.

Tabla 3.16 Calculo de puntos de función no ajustado

		Factor de Ponderación	
Parámetros de Medición	**Cuenta**	**Simple**	**Total**
Numero de entradas de usuario	29	3	87
Numero de salidas de usuario	23	4	92
Numero de peticiones de usuario	31	3	93
Numero de archivos	11	7	77
Numero de interfaces externas	4	5	20
		Cuenta Total	369

Luego de haber hecho el conteo necesario, se procede a llenar la tabla con los datos obtenidos, para obtener los puntos de función no ajustados (Ver Tabla 3.16).

Ahora se calcularan las ponderaciones para las 14 características generales del proyecto, como se muestra en la Tabla 3.17 y tal como se indica en la sección 2.9.2. La escala de 0 a 5 para el factor viene dada de la siguiente forma:

Valor 0: no importante o aplicable

Valor 1: incidental

Valor 2: moderado

Valor 3: medio

Valor 4: significativo

Valor 5: esencial

Tabla 3.17 Calculo de ajuste de complejidad

Nro	Cuestionamiento	Factor
1	¿Requiere el sistema copias de seguridad y de recuperación fiables?	3
2	¿Se requiere comunicación de datos?	5
3	¿Existen funciones de procesamiento distribuido?	0
4	¿Es crítico el rendimiento?	3
5	¿Se ejecutan el sistema en un entorno operativo existente y fuertemente utilizado?	4
6	¿Requiere el sistema entrada de datos interactiva?	5
7	¿Requiere la entrada de datos interactiva que las transacciones de entrada se lleven a cabo sobre múltiples pantallas u operaciones?	3
8	¿Se actualizan los archivos maestros de forma interactiva?	3
9	¿Son complejas las entradas, las salidas, los archivos o las peticiones?	4
10	¿Es complejo el procesamiento interno?	3
11	¿Se ha diseñado el código para ser reutilizable?	2
12	¿Están incluidas en el diseño la conversión y la instalación'?	3
13	¿Se ha diseñado el sistema para soportar múltiples instalaciones en diferentes organizaciones?	4
14	¿Se ha diseñado la aplicación para facilitar los cambios y para ser fácilmente utilizada por el usuario?	3
	Total	45

Utilizando la formula indicada en la sección 2.9.2 se procede al ajuste, por tanto el cálculo de los puntos función totales ajustados es:

PF= cuenta total x [0.65 + 0.01 Σ (Fi)]

PF= 369 x [0.65 + 0.01(45)]

PF= 405.9

Para calcular el valor de funcionalidad se hace una comparación con el valor máximo que se podría obtener si el valor de complejidad seria 70, con lo que se tiene la siguiente relación.

PF_{max}= 369 x [0.65 + 0.01 (70)]

PF_{max}= 369 x [1.35]

PF_{max}= 498.15

Funcionalidad= 405.9 / 498.15 = 0.81

3.5.2 Usabilidad

La usabilidad es un atributo de calidad que mide el grado en que el software es fácil de usar. Viene reflejado por la facilidad de comprensión, facilidad de aprendizaje y operatividad. Es un factor que solo se puede medir indirectamente y esta relacionado con el usuario y con lo amigable que puede ser el software con el usuario. Para realizar esta medición se utilizo un cuestionario y un rango de valores para la evaluación.

Tabla 3.18 Rangos para evaluar la usabilidad

Escala	**valor**
muy bueno	5
bueno	4
aceptable	3
deficiente	2
pésimo	1

El cuestionario se puede ver en la Tabla 3.19 consta de 10 preguntas junto a la respectiva valoración del usuario, donde se ven los valores máximos y mínimos obtenidos para cada pregunta, el valor de evaluación corresponde a la media obtenida a partir de 12 usuarios del sistema,

Tabla 3.19 Valoración para el cuestionario de Usabilidad

No.	Pregunta	Min	Max	Evaluación
1	¿Es fácil de recordar las órdenes y aprender las operaciones básicas?	3	4	3,67≈ 4
2	¿Cómo considera los mensajes que genera el sistema, son entendibles?	3	4	3,75≈ 4
3	¿Considera usted que el Sistema es una herramienta útil?	3	5	4.10≈ 4
4	¿Los reportes que genera son útiles para su trabajo?	4	5	4,67≈ 5
5	¿El Sistema le sirve de ayuda para la toma de decisiones?	3	4	3,42≈ 3
6	¿Recomendaría el sistema a otras instituciones similares?	3	5	4
7	¿Le es fácil orientarse respecto a las acciones que tiene a su disposición, en un determinado instante?	3	4	3,42≈ 3
8	¿El sistema le proporciona suficiente información para realizar sus tareas?	3	4	3,83≈ 4
9	¿La velocidad de respuesta del sistema es suficientemente rápida?	3	5	4
10	¿El sistema llego a cumplir con todas sus expectativas?	3	5	3,92≈ 4
			Total	39

Considerando la Tabla 3. para calcular la usabilidad del sistema se tiene:

Usabilidad = [(4+4+4+5+3+4+3+4+4+4)/10 x100] / 5

Usabilidad = [3.9 x 100] / 5

Usabilidad = 78%

3.5.3 Portabilidad

Se define como la facilidad con que el software puede ser llevado de un entorno de Hardware y Software a otro. Donde se deben tomar atributos internos como ser facilidad de instalación, facilidad de ajuste, facilidad de adaptación al cambio. Para tener una idea mas clara de la portabilidad del sistema se valorara a cada sub-atributo con el rango de valores de la Tabla 3.20

Tabla 3.20 Rangos para evaluar la portabilidad

Escala	valor
muy bueno	5
bueno	4
aceptable	3
deficiente	2
pésimo	1

Valorando la portabilidad del Sistema de Planificación y Seguimiento Académico, se tienen las siguientes apreciaciones:

- Facilidad de instalación, para la instalación del sistema se debe tener instalado en el servidor el gestor de base de datos Postgres, servidor Apache, y soporte de PHP5;
- Facilidad de ajuste, es fácil de ajustar al servidor donde se lo instale, se tendrá que ajustar el archivo de conexión a la base de datos, con los parámetros propios del servidor;
- Facilidad de adaptación al cambio, si tomamos en cuenta la migración del Sistema a otro gestor de Base de datos, para adaptarlo se realizaría una modificación a las clases del sistema con lo cual el sistema podría trabajar con otro gestor de base de datos;
- Soporte de sistemas operativos, el sistema puede ser instalado en servidores con sistemas Windows y Linux, los cuales son muy utilizados en la actualidad.

Tabla 3.21 Valoración para la portabilidad

Sub Atributo	valor
Facilidad de instalación	3
Facilidad de ajuste	4
Facilidad de adaptación al cambio	3
Soporte de sistemas operativos	5

Por lo tanto en base a la Tabla 3.21 se puede tener una idea cuantitativa de la portabilidad como sigue:

Portabilidad=[(3+4+3+5)/4 x 100] / 5

Portabilidad=75%

Se puede decir que el Sistema de Planificación y Seguimiento Académico, tiene un buen nivel de portabilidad.

4 Conclusiones y Recomendaciones

4.1 Conclusiones

Con la implementación del Sistema de Planificación y Seguimiento Académico, se logro mejorar el rendimiento de las actividades académicas de la U.E. Rvdo. P. Walter Strub, como ser los procesos de inscripción de alumnos, elaboración de horarios a partir de la asignación de cursos a los docentes, elaboración de boletines por alumno y por curso, tener un mejor control de la asistencia del alumnado y también del personal que trabaja en la U.E. Rvdo. P. Walter Strub, ahora toda la información generada por dichos procesos se encuentra centralizada.

El uso de Métodos de Análisis y Diseño Orientado a Objetos ayudo a realizar el proyecto de forma mas ordenada y entendible, cumpliendo con los objetivos planteados para el presente proyecto.

Con la implementación del kardex por alumno y por docente se tiene información actualizada y a la mano para una consulta.

La automatización de los procesos de inscripción y la generación de boletines de calificación se logro minimizar los tiempos que se tomaban al realizarlos de forma manual.

El sistema ayuda a la planificación de horarios, tomando en cuenta la doble asignación que solía darse al realizar los horarios de forma manual., con lo que el director podrá organizar los horarios de manera mas eficiente y en menor tiempo.

Con la consulta del historial académico de los alumnos, se tiene la posibilidad de obtener información histórica sobre el rendimiento académico de un estudiante.

El sistema ayuda a tener un mejor control sobre asistencia del personal y alumnado, teniendo en el caso de los estudiantes un historial mas completo con la información generada.

4.2 Recomendaciones

El sistema con las características que presenta puede ser implantado en cualquier Unidad Educativa publica, por otra parte es necesario desarrollar un modulo para la parte contable para que el sistema pueda operar en un colegio privado donde es necesario tener la información de pago de pensiones de los estudiantes.

Para una mejor asignación de aulas se podría hacer un análisis particular, ya que se podría desarrollar un algoritmo genético o implementar uno existente, también como otra alternativa se podría utilizar algoritmos meta heurísticos, esto para realizar una asignación de aulas sin la intervención de personas, teniendo el sistema alguna forma de inteligencia propia.

Bibliografía

Libros:

[Braude, 2003] Braude J, año 2003, “Ingenieria de Software una perspectiva orientada a objetos”, 529 pag, Alfaomega, Mexico

[Fowler, 1999] Fowler M, año 1999, “UML gota a gota”, 224 pag, Addison Wesley Longman, Mexico.

[Larman, 1999] Larman C, “UML y Patrones”, 507 pag, Pearson Addison Wesley, Madrid (España)

[Pressman, 2003] Pressman R., Año 2002, “Ingeniería de Software Un enfoque Práctico”, Quinta Edición, 601 paginas, Mc Graw Hill, España.

[Schmuller, 1999] Schmuller J, “Aprendiendo UML en 24 Horas”, 387 pag.

[Sommer, 2005] Sommerville I, Año 2005,”Ingeniería de Software”, 7ma edición, 687 paginas, Pearson Addison Wesley, Madrid (España)

[Weitzenfeld, 2005] Weitzenfeld A, “Ingeniería de Software Orientada a Objetos con UML Java e Internet”, 678 paginas, Thomson, Mexico

Proyectos:

[Cutipa, 2006] Cutipa G., Año 2006, “Gestión Académica y Administrativa del Colegio Particular Luz a las Naciones”, 109 paginas, Proyecto de Grado UMSA.

[Mercado, 2007] Mercado S., Año 2007, “Sistema de Control y Seguimiento Académico para el Colegio “Rvdo. P. Esteban Bertolusso”, 80 paginas, Proyecto de Grado UMSA.

[Viracochea, 2006] Viracochea E., Año 2006, “Sistema de Gestión Académica para la (Unidad Educativa Piloto Intervida) UEPI”, 92 paginas, Proyecto de Grado UMSA.

Artículos:

[Fuertes, 2008] Fuertes J, “Calidad de Software” Facultad de Informática, Universidad Politécnica de Madrid, 16 pag.

[García, 2000] García R, “Curso del Servidor Apache” Año 2000, 80 paginas

[Palma, 2004] Palma W, “Propuesta de un modelo navegacional para el desarrollo de aplicaciones basadas en OOHDM”, Año 2004, 10 paginas

[Popkin, 2002] Popkin Software and Systems, “Modelado de Sistemas con UML”, año 2002, 24 pag.

[Rosario, 2000] Rosario H, “Prototipos”, Año 2000, 4 paginas, Universidad de Carabobo.

[Rodríguez, 2002] Rodríguez J, “Manual JavaScript”, Año 2002, 89 paginas

[SCHRO, 1998] Schwabe D, Rossi G, “Developing Hypermedia Applications using OOHDM”, Año 1998, 20 pag.

[Silva, 2002] Silva D,”Construyendo aplicaciones web con una metodología de diseño orientada a objetos”, Año 2002, 21 paginas.

Enlaces web:

[Guayaquil, 2009] Colegio Americano de Guayaquil, “Sistema Académico”
Disponible: http://www.colegioamericano.edu.ec
Visitado: Abril de 2009.

[Dilogros, 2009] DISEÑARTE, “Informática & Diseño Especializado en el sector educativo” Disponible: http://www.dilogros.com/
Visitado: mayo de 2009.

[Lamarca, 2009] Lamarca M, “Modelo OOHDM”
Disponible: http://www.hipertexto.info/documentos/oohdm.htm
Visitado: junio de 2009.

[Morales, 2009] Morales M, “Intro al PHP+Postgres”
Disponible: http://brak.unsl.edu.ar
Visitado: Junio de 2009

[Wiki, 2009] Wikipedia, “Ingeniería de Requisitos”,
Disponible: http://es.wikipedia.org/
Visitado: mayo de 2009.

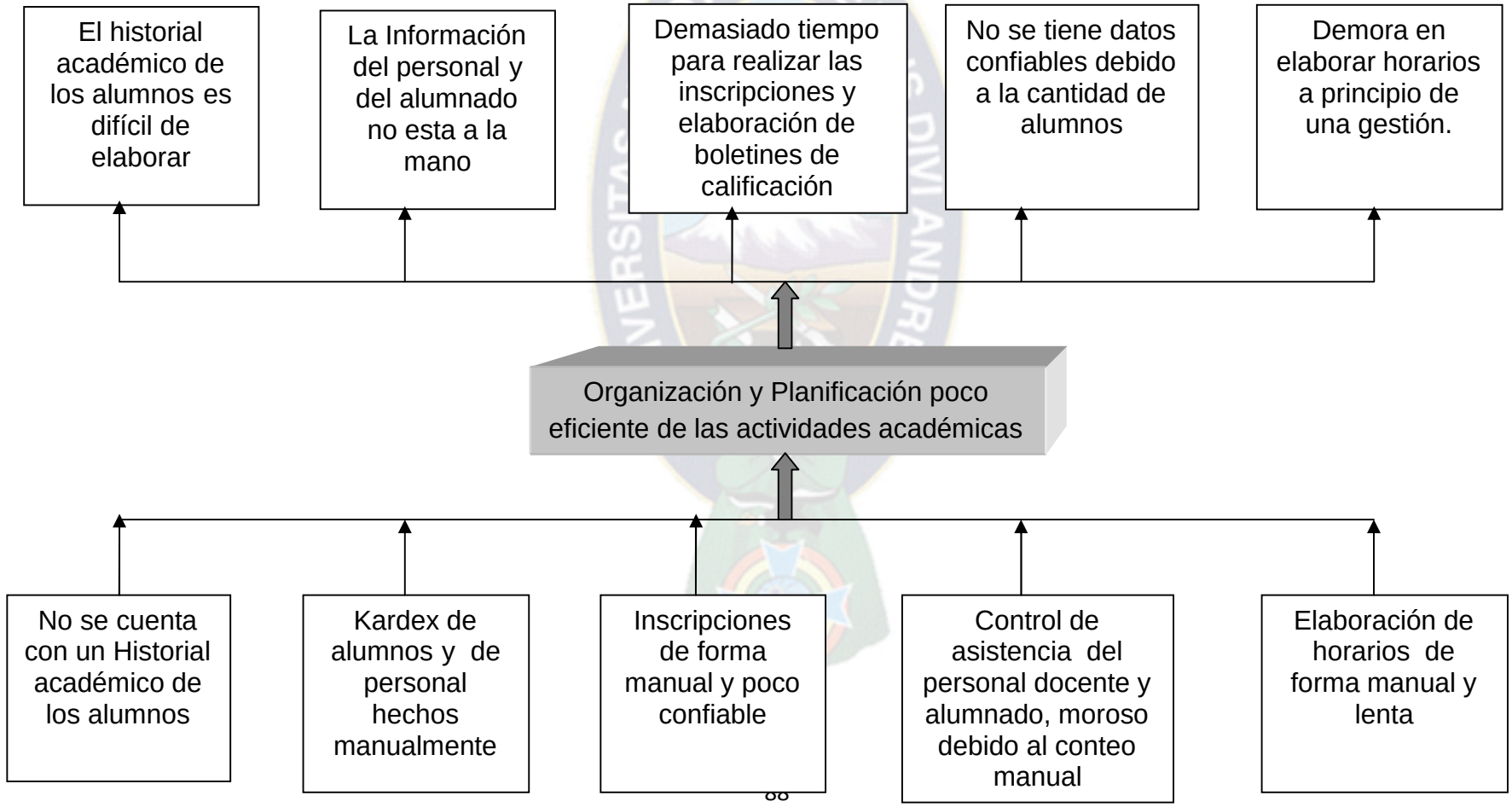
[WikPos, 2009] Wikipedia, “Postgres”,
Disponible: http://es.wikipedia.org/
Visitado: mayo de 2009.

[WikMVC, 2009]. Wikipedia, “Modelo Vista Controlador”,
Disponible: http://es.wikipedia.org/
Visitado: agosto de 2009.

ANEXOS

ANEXO A

ARBOL DE PROBLEMAS



ANEXO B

ARBOL DE OBJETIVOS