PONTIFICIA UNIVERSIDAD CATOLICA DE VALPARAISO

FACULTAD DE INGENIERIA

ESCUELA DE INGENIERIA INFORMATICA

REGISTRO ACADÉMICO PARA LOS ESTABLECIMIENTOS EDUCACIONALES DE DAEM CONCÓN

PROFESOR GUÍA:

ALEXANDRU CRISTIAN RUSU

CLAUDIO MARTÍNEZ ROMERO - FELIPE POLHWEIN SÁNCHEZ

MEMORIA DE TESIS PARA OPTAR AL TITULO PROFESIONAL DE INGENIERO DE EJECUCION EN INFORMATICA

DICIEMBRE DEL 2005

Agradezco:

A Dios, por estar siempre iluminando mi camino y ayudarme a lograr todas mis metas.

A mi familia, por brindarme su incondicional apoyo en los buenos y malos momentos a lo largo de mi vida y sobretodo en mis años de universidad.

A la universidad por entregarme los conocimientos y la formación de la cuál me hacen el profesional exitoso soy ahora.

Claudio Martínez Romero

Agradezco :

A Dios, por conducirme en los buenos y malos momentos de mi existir y por guiarme en el transcurso de mi vida.

A mi familia, por apoyarme, quererme y darme fuerzas en todo momento para lograr los objetivos que me he propuesto.

A la universidad por entregarme sólidos conocimientos profesionales para desenvolverme en el mundo laboral y por mejorar mi disposición como persona.

Felipe Polhwein Sánchez

Resumen

El presente proyecto informático, llamado "Registro Académico para los Establecimientos Educacionales de DAEM Concón.", tiene como finalidad apoyar parte de las labores realizadas en el área administrativa de los establecimientos dependientes de DAEM Concón. Para cumplir con este objetivo, es que el sistema automatiza parte de los procesos realizados por los departamentos de Dirección, Subdirección, Unidad Técnico Pedagógica (U.T.P), Profesores y Secretaria, entre los cuales se encuentran la confección de certificados, confección de libreta de notas, registros de asistencias, registros de datos de los docentes y de los alumnos, además de la generación de actas finales dirigidas al MINEDUC para asuntos de subvención, de esta manera se agilizará la realización de estas labores, disponiendo así de más tiempo para efectuar procesos quizás mas importantes en materia educacional.

En el desarrollo se utilizaron metodologías y herramientas estudiadas con el fin de llevar el proceso de desarrollo lo más ordenado e íntegro posible, manteniendo bien definidas las etapas de análisis, diseño, codificación y pruebas, según el paradigma RUP. Previo a esta etapa se realizó un estudio de factibilidad y riesgos para definir la viabilidad del proyecto y posterior a la etapa de desarrollo se culminó con la puesta en marcha del sistema.

Los resultados obtenidos fueron satisfactorios de acuerdo con los objetivos del proyecto planteados desde un comienzo, por lo tanto se cumplieron a cabalidad los requisitos propuestos por el cliente y dando solución a la problemática expuesta por DAEM Concón, siendo la culminación sistema de Registro Académico.

Abstract

The present computer science project, called "Registro Académico para los Establecimientos Educacionales de DAEM Concón.", has like purpose of supporting part of the workings made in the administrative area of the dependent establishments of DAEM Concón. In order to achieve this objective, it's the system automates part of the processes developed by the departments of Direction, Sub direction, Pedagogical Technical Unit (P.T.U), Professors and Secretary, between who they are the certificate preparation, calification registry, registries of attendances, registries of students and teachers ducational data, in addition to final the act generation from to MINEDUC for the subvention subjects, this way will make agile the accomplishment of these workings, this means they will have more time to complete important processes in educational matter.

In the development its used methodologies and tools studied with the purpose of taking the process of possible the most ordered and complete development, maintaining defined the stages of analysis, design, codification and tests well, according to paradigm RUP. Previous to this stage a feasibility study was made and risks to define the viability of the project and subsequent to the development stage was culminated with the beginning of the system in production.

The obtained results were satisfactory in agreement with the raised objectives of the project from a beginning, therefore the requisites proposed by the client were full achieved and giving problematic solution to exposed by DAEM Concón, being the culmination system of Academic Registry.

Capítulo 1: Introducción

1.1 Introducción

La mayoría de las organizaciones actuales utilizan variados sistemas informáticos para el manejo de la información como por ejemplo: ayudan a agilizar el trabajo, permiten otorgar una mejor atención a los clientes, etc. Estos sistemas dependen de las necesidades que posee la organización, las cuales requieren alcanzar una mejora en el manejo de información, almacenamiento de datos, e incluso en la toma de decisiones.

El **proceso de desarrollo** de un sistema transcurre como una **secuencia de pasos** bien definidos, los cuales están basados en argumentos sólidos. Estos pasos son los que permiten traducir las necesidades de un cliente a un **"modelo de sistema".**

El tema de proyecto ***"Registro Académico para los Establecimientos Educacionales de DAEM Concón"*** se desenvuelve en el ámbito de la educación, cuyo principal objetivo es fomentar el desarrollo de la educación en todos sus niveles y el aprendizaje de calidad para todos los niños, jóvenes y adultos durante su vida.

Para organizar el proceso de desarrollo del sistema se utilizará el paradigma **RUP** *e*n cuya su primera parte corresponde a la etapa de **Inicio**, se definirán aspectos claves para llevar a cabo las bases de este proyecto, se establecerán los objetivos y se propondrá una solución adecuada a las necesidades basadas en los requerimientos presentados por el cliente. Esta solución será analizada en todos los aspectos en base a un estudio de factibilidad y en los posibles riesgos que pudiera tener la ejecución del mismo.

Durante la etapa de **análisis** se darán a conocer los requisitos refinados y estructurados, describiendo qué hará el sistema en términos de orientación a objetos, sin considerar el cómo será implementado. Posteriormente durante la etapa de **diseño** del sistema se verá el cómo será implementado en término de arquitectura gracias a la utilización de los recursos

que nos ofrece el paradigma RUP, la metodología orientada a objeto y las herramientas escogidas, con el fin de tener las bases para una próxima implementación de la solución.

Al inicio de la etapa de **Construcción** nuevamente se realiza las etapas de análisis y diseño a nivel de refinamiento de los modelos para verificar posibles inconsistencias, falencias, etc. y así efectuar las correcciones necesarias al sistema antes de ser codificado. También en esta etapa se confeccionará el **Plan de Pruebas**, el cuál se enfocará sobre cada componente del sistema para verificar y validar el producto de software fuera codificado de manera correcta. Finalmente en la etapa de **Transición**, siguiendo con el paradigma RUP, se hace una entrega de la versión "beta", o más bien, se prueba de manera funcional el prototipo de sistema.

1.2 Descripción de la Organización.

El **Departamento de Administración de Educación Municipal (DAEM) de Concón** supervisa cuatro establecimientos educacionales, tres de los cuales corresponden a escuelas básicas y uno a un liceo.

DAEM controla al personal docente que son 140 profesionales, y al personal paradocente que son 45 personas, entre las que se cuentan secretarias, auxiliares, inspectores de patio, etc. Los profesores se dividen en tres categorías: titulares, que son aquellos que trabajan para un mismo empleador de manera indefinida; a contrata, que son los profesores que poseen contrato vigente con fecha de término y reemplazantes, que son aquellos docentes que están disponibles para ser contratados como reemplazos de los profesores que hayan presentado licencia médica. En estos momentos, existen 87 profesores titulares y 53 a contrata.

El Departamento de Educación se rige bajo el departamento **jurídico del Mineduc**, el cual posee todo y cada uno de los documentos jurídicos que establece el funcionamiento en forma legal de la educación chilena.

El financiamiento de DAEM proviene de la subvención educacional, la cual se entrega de acuerdo a la asistencia de los alumnos. Esta subvención se paga mensualmente usando el

promedio de asistencia estudiantil de los últimos tres meses, también es necesario recalcar que también llega financiamiento extra por conceptos de excelencia académica y de desempeño difícil, estipulados en las leyes vigentes. También existe un aporte municipal e ingresos externos vía proyectos. El liceo politécnico tiene financiamiento compartido, con una cuota mensual fijada por los apoderados.

En la ilustración 1.1 se muestra el organigrama que corresponde al Departamento de Administración Educacional de Concón.

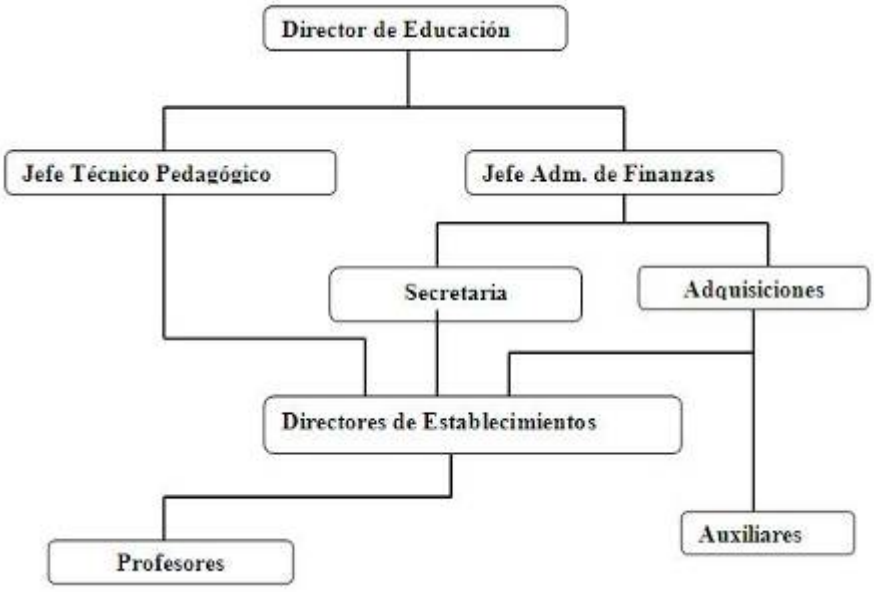

Ilustración 1-1 Organigrama DAEM Concón

1.2 Descripción de Procedimientos Actuales

A continuación se explicará la obtención y manejo de la información relevante para cada establecimiento educacional necesaria en la generación de los registros académicos.

- El proceso de **matrícula** es realizado por los profesores dos veces en el año, en diciembre y en marzo para aquellos alumnos que se integran a cada establecimiento y en noviembre para aquellos que continúan en sus respectivos establecimientos. En este proceso se recogen todos los antecedentes personales, de salud, familiares y socioeconómico del alumnado.

- El registro de la **asistencia** por curso es realizado por la secretaría de cada colegio, se debe contabilizar dentro del día. Este proceso tiene como finalidad generar un informe para la **subvención**.

- Proceso de confección de informe de **notas parciales**, realizado por los profesores jefes de cada curso, las notas traspasadas del **libro de clases** al informe que se entrega mensualmente en las reuniones de padres y apoderados.

- Proceso de confección de la **libreta de notas** el cual posee los promedios final por periodo, promedio general por periodo, nota de examen y/o prueba especial (si existe), promedio final por asignatura, promedio final anual, porcentaje de asistencia por periodo y final, también posee la evaluación del periodo del área de desarrollo y su **situación final** (aprobado o reprobado).

- Proceso de confección de certificados los cuales son los siguientes:

  - **Certificado Anual de Estudios:** Posee información de las asignaturas cursadas en el año con su promedio final, porcentaje de asistencia y situación final.

  - **Certificado de Concentración de Notas**: Contiene los promedios finales por asignatura, promedios generales y la situación final, nombre del establecimiento y porcentaje de asistencia de cada año.

  - **Certificado de Notas Parciales**: Posee notas parciales por asignatura, días de asistencia, de inasistencia y observaciones.

  - **Certificado de Alumno Regular** el cual certifica que el alumno pertenece al establecimiento.

- Proceso de generar un informe del rendimiento académico del establecimiento, el cual es emitido a fin de cada periodo por **U.T.P.**

- En el caso del liceo politécnico la elección de las asignaturas electivas, que deben ser según la preferencia de los alumnos.

- Proceso de generar las **actas finales**, las cuales deben ser enviadas al MINEDUC al final de cada año.

1.3 Situación Actual y Problemas Detectados

En la actualidad los tres colegios y el liceo politécnico poseen un pequeño sistema informático provisto por DAEM Concón, el cual tiene algunos de los procesos descritos anteriormente automatizados y es capaz de gestionar la mayor parte de los registros académicos. Sin embargo, mucho de los procesos se siguen efectuando en forma manual, lo que conlleva a la pérdida de tiempo. Dicho sistema es **monousuario**, por lo tanto en cada colegio existe solamente una persona que lo utiliza, en el caso del liceo politécnico la persona encargada del ingreso de la información al sistema no da abasto en el tiempo asignado, conllevando a la inutilización de muchas funciones del programa

El sistema está implementado en un **solo computador**, posee una base de datos en **Microsoft Access** y no posee documentación o **manual de usuario** alguno. Este motor de base de datos no cumple con las propiedades **ACID**, por lo cual existe una inconsistencia de datos dentro del programa. Cuando se ingresan los datos de matrícula, si el usuario comete un error, el sistema se cae perdiendo así toda la información ingresada hasta el momento, o bien, si al ingresar datos se ingresa alguno errado o si se necesita modificar el dato, el sistema no lo permite, por lo cual se necesita llamar a un especialista para poder ingresar a la base de datos y arreglar el error o modificar los datos, implicando así una dependencia exagerada del especialista. Al final de cada periodo el sistema genera las estadísticas correspondientes, pero al existir inconsistencia en los datos, las estadísticas son inútiles teniendo que ser realizadas a mano por el encargado. En el momento de ingresar un nuevo alumno al sistema y luego asignarle un curso, si la lista ya esta compuesta el nuevo alumno se ingresara al final de la lista sin poder ordenarlo por orden alfabético, ya que si se mueve al alumno en el numero de la lista correcto, se genera un desorden en cuanto a la asistencia y notas de los alumnos.

En el momento que se inicia el nuevo año académico, la base de datos del año anterior ya no es accesible, imposibilitando así cualquier consulta a años anteriores, además de no poder llevar un registro histórico, importante para la generación de informes de estadísticos.

1.4 Objetivos del Proyecto

1.4.1 Objetivo general

• Desarrollar un sistema informático que gestione todos los **registros académicos** de los colegios dependientes de DAEM Concón, para controlar la información escolar relevante, necesaria para la **asignación de los recursos (subvención, desempeño difícil, excelencia académica).**

1.4.2 Objetivos específicos

Para poder cumplir con el objetivo general del proyecto se definieron los siguientes objetivos específicos:

• Tener **información** personal y académica **actualizada** del alumnado y del cuerpo docente de cada establecimiento.

• Mejorar el proceso de **matrícula, control de asistencia y registro de notas**.

• Minimizar el tiempo invertido, por los docentes y administrativos en la generación y manipulación de registros.

• Gestionar mecanismos apropiados para la generación de informes **(actas)** de acuerdo a los estándares establecidos en el **RECH (Registro de Estudiantes de Chile) del Mineduc**.

• Ofrecer una herramienta que sea un **real apoyo** a las necesidades de DAEM Concón y sus establecimientos.

1.6 Metodología de Trabajo

La metodología de trabajo describe las actividades que debe realizar el equipo desarrollador para cumplir con los objetivos del proyecto. A continuación se describirán dichas actividades:

• Realizar **reuniones** técnicas formales con la persona a cargo de la organización, en este caso es el director de DAEM Concón exponiendo nuestra idea de trabajo

• Coordinar e inspeccionar en terreno como es el funcionamiento de cada establecimiento educacional dependientes de DAEM Concón.

• **Entrevistar** a cada usuario final para conocer la real problemática y poder establecer alternativas de solución.

• Evaluar y elegir la **mejor alternativa** y establecer los objetivos generales y específicos del proyecto.

• Coordinar **reuniones periódicas** con el cliente.

• Verificar las tareas específicas de cada usuario **(stakeholders)** que participará del sistema.

• Documentarse si existen trabajos similares u otros sistemas que gestionan la labor académica.

• Estudio y elección de **paradigma, metodología** y de las **herramientas** que servirán para la implementación del sistema.

• Un **estudio de factibilidad** del proyecto, el cual determinará si el proyecto es viable o no. Para ello, se toman en consideración la factibilidad técnica, económica, operacional y legal.

• La **implementación** de la solución propuesta con todas sus etapas correspondientes.

Capítulo 2: Estudio Preliminar

2.1 Estudio de Factibilidad

Para todos los sistemas nuevos, el proceso de **ingeniería de requerimientos** empieza con el estudio de factibilidad. La entrada de éste es una descripción resumida del sistema y de cómo se utilizará dentro de una organización. El resultado del estudio es un informe que recomienda si es conveniente llevar a cabo la ingeniería de requerimientos y el proceso de desarrollo de sistema [1].

2.2.1 Factibilidad Técnica

La factibilidad técnica consiste en determinar, para cada escenario si la solución que trae consigo es susceptible de llevar a cabo el desarrollo del sistema, con los recursos informáticos con los que cuenta la empresa, tanto computacionales como de comunicación, incluyendo los conocimientos técnicos respectivos, disponibles en la organización [1]. En consecuencia, luego de un análisis, cada establecimiento educacional cuenta con los siguientes recursos:

Recursos de software:

- Sistema operativo **Microsoft Windows** 98 y XP.

Software de ofimática Microsoft Office 97 y XP.

- Recursos de hardware:

Equipos HP, Intel Celeron 700 Mhz, 192 Mb RAM.

- Equipos AMD Duron 1.2 Ghz, 256 Mb RAM.

Equipos Intel Pentium 3 700 Mhz, 256 Mb RAM.

- **Servidor** AMD Athlon XP 2000+, 512 RAM, Disco 80 Gb NTFS, Windows XP Profesional Service Pack 2.

De acuerdo a las especificaciones técnicas descritas, se ha determinado que para la implementación del sistema solamente se requiere de:

- Un sistema administrador de base de datos **(SABD)**, PostgreSQL-8.0.2 para Microsoft Windows XP.

Un lenguaje que permita generar y procesar la información de formularios. Para lo cual se utilizará PHP4.

- Un servidor web el cual permita dar soporte al lenguaje especificado.

De lo anteriormente visto se concluye que los colegios y el liceo politécnico cuenta con los requerimientos mínimos en cuanto a hardware y software para la implementación del sistema. Por lo tanto técnicamente **es factible** de llevar a cabo el proyecto.

2.2.2 Factibilidad Operacional

La factibilidad operacional consiste en determinar la capacidad potencial de la organización para llevar a cabo el proyecto en término de los **planes, políticas y procedimientos vigentes**, es decir, se trata de averiguar a qué se expone la organización al incorporar un nuevo sistema, cual será la reacción que suscitará en los demás procesos, en los recursos humanos y en general en toda la organización, considerando incluso los clientes [1].

El sistema que se pretende implementar se ajustará a la forma de trabajar en el proceso de gestión de registros académicos, sin tener que realizar cambios severos en la manera de trabajar que tiene cada establecimiento. Para los usuarios el sistema no entorpecerá las tareas específicas sino facilitará la realización de éstas.

Dentro de los cuatro colegios, la mayoría de los empleados cuenta con conocimientos a nivel de usuario en el manejo de computadores. Además el personal tiene ideas claras referentes al funcionamiento de sus tareas, lo cual facilitará la implantación del nuevo sistema con un mínimo de capacitación.

Por lo tanto **es factible** llegar realizar el sistema cumpliendo operacionalmente la forma de trabajar en los establecimientos educacionales.

2.2.3 Factibilidad Legal

Bajo este punto del análisis se pretende verificar que al implantar un nuevo sistema dentro de una empresa, no se incurra en algún **delito** contra la **propiedad intelectual** de otros desarrolladores de software. Para ello se debe considerar la legislación vigente en el ámbito nacional correspondiente a leyes informáticas. También es importante considerar las **políticas internas de la empresa**, tratando de afectar de la menor forma posible el accionar o el desarrollo interno de la empresa y que al utilizar el sistema incurra en algún delito, en el caso para DAEM, se debe verificar que el sistema no viole las **leyes vigentes** de la **educación en Chile**.

DAEM cuenta con las licencias necesarias de los sistemas operativos y se utilizará software libre para la implementación del sistema, por lo tanto es factible legalmente de realizar el proyecto.

2.2.4 Factibilidad Económica.

Para la implementación de este sistema, los establecimientos educacionales dependientes de DAEM Concón ya cuentan con los requerimientos mínimos, por lo tanto no se necesita invertir en tecnologías (impresoras, computadores).

Para el desarrollo de este sistema, se ha decidido utilizar **herramientas de libre distribución** puesto a que DAEM no cuenta con la disponibilidad de herramientas de desarrollo ni tampoco con el capital para invertir en una de ellas.

El costo incurrido en el desarrollo del sistema será asumido por los desarrolladores, puesto a que se trata de un **proyecto de carácter académico**. Por lo tanto se concluye que el sistema es económicamente factible de realizar.

2.2 Análisis de Riesgos

Una tarea importante dentro del desarrollo de un proyecto de software es la anticipación de los riesgos que podrían afectar la programación del proyecto o la calidad del mismo, y a la vez poder crear acciones preventivas ante estos riesgos. Los resultados de este análisis de riesgos se deben tener documentado durante todo el desarrollo del mismo junto con el análisis de lo que podría pasar ante la eventualidad de un riesgo. El ser capaz de identificar estos riesgos y crear un plan de prevención se denomina Administración de Riesgos [4].

Existen cuatro etapas para el proceso de administración de riesgos:

Identificación de Riesgos.
Análisis de Riesgos.
Planeación de Riesgos.
Supervisión de Riesgos.

Tabla 2-1 Identificación del Risego

Riesgo	Tipo Riesgo	Descripción
Rotación de Personal	Proyecto	Antes de que termine el Proyecto el personal experimentado abandona su puesto de trabajo.
Cambio Administración	Proyecto	Cambio de la administración organizacional con diferentes prioridades.
Ausencia del personal experimentado	Proyecto	El personal principal para el desarrollo del sistema se encuentra enfermo y no disponible en los momentos críticos.
Cambios en los Requerimientos	Proyecto y Producto	Dependiendo de la evolución del proyecto irán apareciendo otros requerimientos en forma anticipada.

Retrasos en la especificación	Proyecto y Producto	Las especificaciones principales del sistema no van a estar listas antes de la fecha de entrega.
Subutilización del soporte del sistema	Proyecto y Producto	El SABD que sustenta la Base de Datos del sistema no es bien utilizada.
Dificultad en el proceso de transacciones	Proyecto y Producto	El SABD no puede procesar demasiadas transacciones por segundo según lo esperado.
Bajo desempeño con la utilización de PHP	Producto	El lenguaje con el que se ha desarrollado el proyecto no tiene el desempeño adecuado.
Bajo desempeño de la herramienta CASE	Producto	La herramienta CASE que ayuda al proyecto no tiene el desempeño anticipado.
Cambio de Tecnología	Negocio	La tecnología en la cual se pretendía implementar el sistema ha sido cambiada por nueva tecnología.
Competencia del producto	Negocio	Un producto de la competencia sale al mercado antes de la finalización del proyecto.

Tabla 2-2 Análisis de Riesgos

Riesgo	Probabilidad	Efectos
Antes de que termine el Proyecto el personal experimentado abandona su puesto de trabajo.	Baja	Catastrófico
Cambio de la administración organizacional con diferentes prioridades.	Baja	Serio
El personal principal para el desarrollo del sistema se encuentra enfermo y no disponible en los momentos críticos.	Moderada	Serio

Dependiendo de la evolución del proyecto irán apareciendo otros requerimientos en forma anticipada.	Moderada	Serio
Las especificaciones principales del sistema no van a estar listas antes de la fecha de entrega.	Alta	Serio
El SABD que sustenta la Base de Datos del sistema no es bien utilizado.	Moderada	Serio
El SABD no puede procesar demasiadas transacciones por segundo según lo esperado.	Moderada	Serio
El lenguaje con el que se ha desarrollado el proyecto no tiene el desempeño adecuado.	Alta	Tolerable
Es ineficiente el uso de la herramienta CASE	Moderada	Insignificante
La tecnología en la cual se pretendía implementar el sistema ha sido cambiada por nueva tecnología.	Baja	Tolerable
Un producto de la competencia sale al mercado antes de la finalización del proyecto.	Moderada	Serio

Tabla 2-3 Planeación del Riesgo

Riesgo	Estrategia
Ausencia Personal Experimentado	Se debe de mantener en conocimiento de esta posible falta al cliente, y las posibilidades que puede desarrollar el cliente para sobrellevar el riesgo.
Reestructuración Organizacional	Se debe de preparar un documento indicando los beneficios que otorga el proyecto para la empresa

Enfermedad del Personal	Se debe de hacer rotación en los puestos de trabajo para que todos sepan el trabajo de los demás.
Cambio en los Requerimientos	Rastrear la información para valorar el impacto de los requerimientos, maximizar la información oculta en ellos
Demoras temporales de las Especificaciones	Reorganizar las labores dentro del grupo de trabajo, y potenciar las virtudes del grupo laboral.
Desempeño del SABD	Estudiar la posibilidad de compra de una base de datos con un desempeño más alto.
Lenguaje defectuoso	Investigar los lenguajes que se han adquirido, y las posibilidades que ofrece el mercado frente a la disponibilidad de lenguajes similares.
Bajo desempeño de la Herramienta CASE	Desechar la herramienta y elegir una nueva, o no utilizar alguna.
Cambio de Tecnología	Averiguar las posibilidades que ofrecen los distintos equipos que se encuentran en el mercado para la implementación del proyecto.

Tabla 2-4 Supervisión del Riesgo

Tipo de Riesgo	Indicadores Potenciales
Ausencia de Personal Experimentado	La especialización de las tareas dentro de la empresa siempre es un obstáculo cuando hay ausencia laboral
Reestructuración Organizacional	Falta de acciones por parte de la plana administrativa.

Cambio en los Requerimientos	Demasiadas peticiones de cambios en los distintos departamentos de la empresa.
Demoras temporales de las Especificaciones	Fracaso en el cumplimiento de los tiempos acordados.
Desempeño de la herramienta CASE	Rechazo del equipo para utilizar la herramienta, quejas sobre la herramienta.
Desempeño del SABD.	Tecnología atrasada, defectuosa, mal aprovechamiento de los recursos.
Lenguaje defectuoso	Software defectuoso, falta de conocimiento de los desarrolladores frente a un lenguaje, poca familiaridad del usuario frente a los lenguajes propuestos por los desarrolladores.
Cambio de Tecnología	Entrega retrasada del hardware muchos problemas tecnológicos reportados
Competencia del Producto	La organización adquiere un producto en el mercado

2.3 Estudio y Elección de Paradigma

2.3.1 Ciclo de Vida Clásico

El “Ciclo de Vida Clásico” o “Modelo en Cascada”, dice de un desarrollo secuencial del desarrollo del software, el cual empieza con un nivel de sistemas, continúa con el análisis, el diseño, la codificación, las pruebas y el mantenimiento.

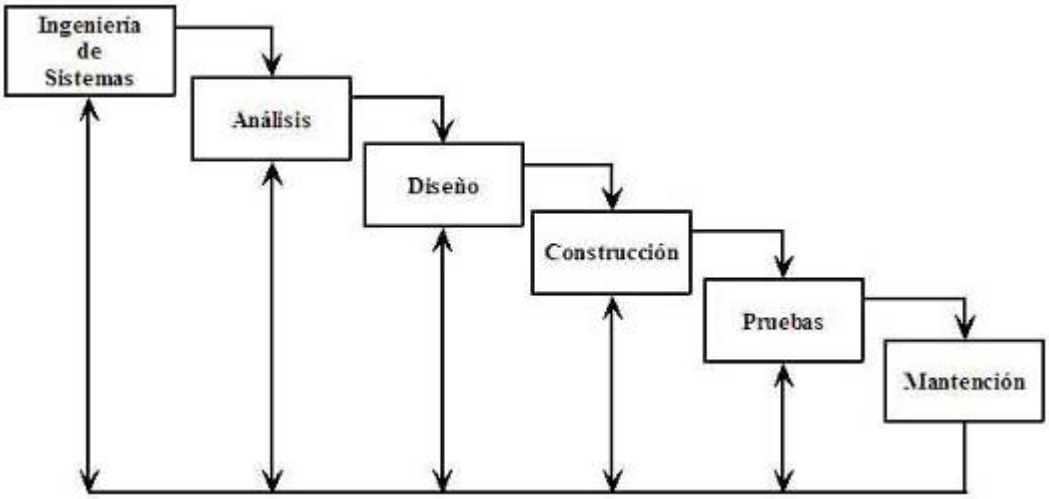

Ilustración 2-1 Ciclo de Vida Clásico

Ingeniería y Modelado de Sistemas: Es en esta etapa se recolectan los datos necesarios de los requisitos del sistema. Estos datos reunidos sirven al propósito de conectarlos con el análisis y el diseño. Esta *ingeniería de sistemas* se puede asociar tanto a los requisitos de los niveles estratégicos de empresa y como al nivel de área de negocios.

Análisis de los Requisitos del Software: En este se debe tener en claro el dominio de la información en donde se desarrollará el software, así como también la tarea que se pretende solucionar, el comportamiento, el rendimiento y su interconexión. Acá se debe de tener muy presente al cliente, si bien es una parte esencial dentro de todo el proyecto, es bien importante en esta etapa puesto que es él quien documenta y repasa los requisitos del sistema y del software.

Diseño: En esta etapa se deben tomar en cuenta, arquitectura del software, representaciones de interfaces, y los algoritmos. El diseño traduce los requisitos en una representación del software, la cual se puede evaluar antes de codificarla.

Generación de Código: Es la traducción del diseño a una forma comprensible para la máquina. Depende mucho de que el diseño esté bien confeccionado, el hecho de que esta tarea sea fluida.

Pruebas: Cuando el código está listo entonces comienza el proceso de probar los procesos lógicos internos del software, asegurando de esta manera que todo el código sea utilizado, y

en los procesos externos funcionales para detectar errores y asegurar la producción de resultados.

Mantenimiento: Una vez que el software está instalado, es muy probable que sufra ciertos cambios, debido a diversos factores. Por lo que acá se debe de aplicar desde el inicio el paradigma a un problema que ya está en la empresa y no a uno nuevo.

En este paradigma se pueden encontrar los siguientes problemas:

Los proyectos no siguen el modelo secuencial propuesto.
Dificultad en la obtención de los requisitos, ya que para el cliente es muy difícil establecer sus necesidades.
Las versiones para el cliente no están disponibles sino hasta avanzado el proyecto.
El cliente debe tener paciencia para esperar el resultado final.
Retrasos por parte de los responsables del desarrollo del software.

2.3.2 El Modelo de Construcción de Prototipos

La Construcción de Prototipos es un proceso que facilita al programador la creación de un modelo del software a construir. El modelo puede tomar una de las tres formas siguientes:

Un prototipo en papel a un modelo basado en PC que describa la interacción hombre-máquina.
Un prototipo que implemente algunos subconjuntos de la función requerida del programa deseado.
Un programa existente que ejecute parte o toda la función deseada, pero que tenga otras características que deban ser mejoradas en el nuevo trabajo de desarrollo.

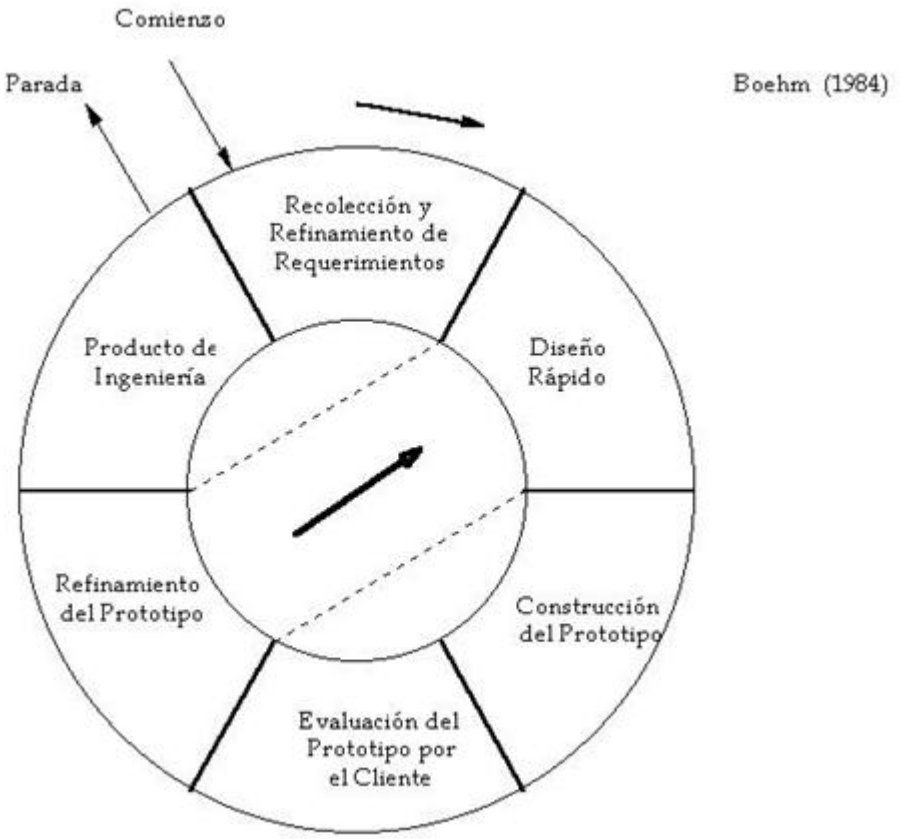

Ilustración 2-2 Construcción de Prototipos

Recolección de Requerimientos: El desarrollador y el cliente buscan y definen los objetivos globales para el software, identifican los requisitos conocidos, y las áreas del esquema en donde es obligatoria más definición.

Diseño Rápido: Se centra en una representación de esos aspectos del software que serán visibles para el usuario / cliente, por ejemplo enfoques de entrada y formatos de salida.

Construcción del Prototipo: El prototipo lo evalúa el cliente / usuario y lo utiliza para refinar los requerimientos del software a desarrollar. La interacción ocurre cuando el prototipo satisface las necesidades del cliente, a la vez que permite que el desarrollador entienda mejor lo que hay que hacer.

Evaluación del Cliente: Consiste en la presentación del prototipo al cliente. En esta etapa se permite especificar los requerimientos que por alguna razón no se incluyeron en la etapa de recolección de requerimientos, o bien aclarar defectos encontrados en el prototipo

construido. Si el prototipo no es aprobado, se debe regresar a la etapa de diseño rápido con los nuevos requerimientos y correcciones, para volver a construir un nuevo prototipo. En caso contrario el prototipo avanza a la etapa de producto final.

Producto Final: En esta etapa se elabora un software nuevo basado en el prototipo aprobado por el cliente.

Se debe agregar que este Paradigma es una herramienta que facilita la interacción con el cliente en la obtención y refinamiento de los requerimientos, y la detección de errores existentes en el prototipo del software, ya que pueden ser incluidos en el próximo prototipo.

2.3.3 Modelo en Espiral

Este modelo es un modelo de proceso de software que reúne la naturaleza interactiva del Modelo de Construcción de Prototipos con la parte rigurosa y sistemática del Modelo Lineal Secuencial. Este se divide entre tres y seis sectores de tareas:

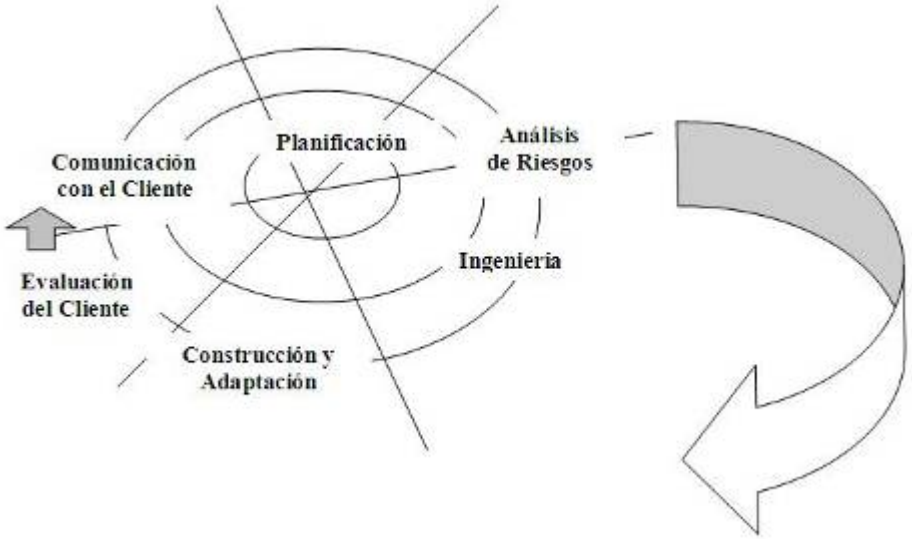

Ilustración 2-3 Modelo de Espiral

Comunicación con el Cliente: se establecen las tareas requeridas para el proyecto.

Planificación: aquellas tareas que definen los recursos, el tiempo a ocupar y los datos anexos al proyecto.

Análisis de Riesgos: son esas actividades relacionadas en la evaluación de los riesgos de gestión y en los técnicos.

Ingeniería: acá es donde se construyen las representaciones de las aplicaciones.

Construcción y Adaptación: acá se construye y adapta el soporte, para el usuario.

Evaluación del Cliente: esta es la última etapa la cual evalúa las representaciones del software, y donde se obtiene la reacción del cliente.

El espiral usa la construcción de prototipos para reducir los riesgos, y permite al desarrollador aplicar el paradigma en cualquier etapa de la evolución; y mantiene el enfoque lineal del Modelo Lineal Secuencial, pero ajustándolo al marco de trabajo interactivo del mundo real.

2.3.4 Técnicas de Cuarta Generación

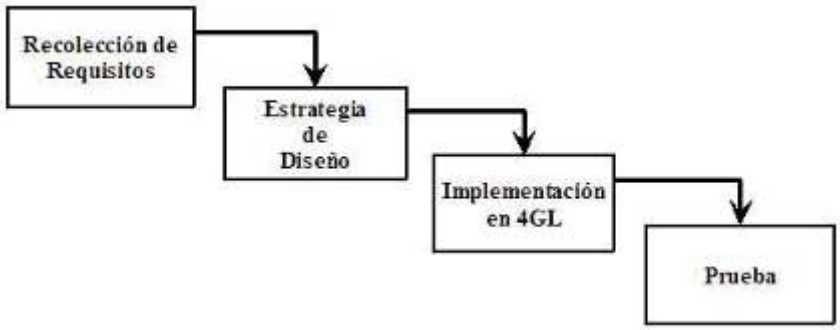

Ilustración 2-4 Técnica de Cuarta Generación

Este paradigma se orienta hacia la posibilidad de especificar el software usando formas de lenguaje especializado o notaciones gráficas que describan el problema que hay que resolver en términos que los entienda el cliente.

Al igual que otros paradigmas, T4G comienza con el paso de reunión de requisitos.

Idealmente, el cliente describe los requisitos, que son, a continuación, traducidos directamente a un prototipo operativo.

Para aplicaciones pequeñas, se puede ir directamente del paso de recolección de requisitos al paso de implementación, usando un lenguaje de cuarta generación no procedimental (L4G). La implementación mediante L4G permite, al que desarrolla el software, centrarse en la representación de los resultados deseados, que es lo que se traduce automáticamente en un código fuente que produce dichos resultados. Para transformar una implementación T4G en un producto, el que lo desarrolla debe dirigir una prueba completa, desarrollar con sentido una documentación y ejecutar el resto de las actividades de integración requeridas en los otros paradigmas de ingeniería de software.

Se debe considerar que pueden existir otros modelos además de los expuestos con anterioridad, muchos de esos resultarán de variantes en los modelos anteriores o incluso combinaciones de ellos. Algunos de estos modelos son:

El Modelo Incremental: El cual combina los elementos lineales del Modelo Lineal Secuencial y el de Construcción de Prototipos, es por esto, que cada una de las secuencias lineales genera un "incremento" del software.

El Modelo de Ensamblaje de Componentes: El denominado Paradigma de Orientación a Objetos, se basa en la creación de clases que encapsulan datos, tal como los algoritmos que manejan los datos.

Estas tecnologías de objetos otorgan un ámbito de trabajo técnico para los modelos de procesos basados en componentes para la ingeniería de software.

El Modelo de Ensamblaje de Componentes es evolutivo y exige un enfoque interactivo para la creación del software, al igual que el modelo en espiral, pero este configura las aplicaciones desde componentes preparados de software.

El Modelo de Desarrollo Concurrente: Este modelo se representa como un esquema que contiene varias actividades que están relacionadas.

Se utiliza para el desarrollo de aplicaciones cliente/servidor y define actividades en dos sentidos:

Una dimensión de sistemas: Diseño, ensamblaje, Uso.
Una dimensión de componentes: Diseño, realización.

La concurrencia del modelo se puede lograr mediante la ocurrencia simultánea de las actividades tanto de sistemas como de componentes, y también mediante la implementación de muchos componentes para una aplicación cliente/servidor, las cuales se pueden diseñar y realizar concurrentemente.

2.3.5 Paradigma Proceso Unificado de Desarrollo de Software de Rational (RUP)

Este es un proceso de desarrollo de software, corresponde a un conjunto de actividades necesarias para realizar un sistema informático. Pero esto es más que un simple proceso, ya que se dice que es un marco de trabajo genérico que se puede especializar para una gran variedad de sistemas, para las distintas áreas de aplicación, tipos de organización, niveles de aptitud y tamaños de proyecto.

El Proceso Unificado está basado en componentes, lo cual quiere decir que el sistema en construcción está formado por componentes de software intercomunicados a través de interfaces definidas.

Para la generación de los esquemas de un sistema, trabaja en base al Lenguaje Unificado de Modelado (UML), siendo este una parte esencial del paradigma.

Cuando hablamos de Proceso Unificado, casi siempre involucra una metodología de Análisis de Orientación a Objetos, del que se deriva el concepto de casos de uso, los cuales representan a los requisitos funcionales del sistema, denominando al conjunto de ellos Modelo de Casos de Uso, en donde se describe la funcionalidad del sistema completo.

Este paradigma es iterativo e incremental, ya que se cuenta con la iteración de las tareas del desarrollo del software. Estas divisiones del proyecto generan el incremento del mismo. Las iteraciones hacen referencia a pasos en el flujo de trabajo, y los incrementos, al crecimiento del producto. Para que todo esto sea más efectivo se debe de controlar las iteraciones

(mediante selección y ejecución planificada). Debido a esto, el Proceso Unificado otorga ciertos beneficios al desarrollo de los sistemas:

La iteración controlada reduce el coste de riesgo a los costes de un sólo incremento.
Reduce el riesgo de no cumplir con los compromisos adquiridos dentro de un período determinado.
Acelera el ritmo total del esfuerzo del desarrollo, ya que se trabaja para conseguir metas claras a corto plazo.
Los requerimientos del usuario, en vez de ser definidos en forma temprana, se van refinando a medida que se avanza en las iteraciones, haciendo más fácil la adaptación a los requisitos cambiantes.

El ciclo de vida de un sistema se compone de varios ciclos, cada uno consta de cuatro fases: inicio, elaboración, construcción y transición.

Fase de inicio: En este se describe el producto final, y esta fase también debe de señalar las principales funciones del sistema para los usuarios, como debe de ser la arquitectura, y la planificación del proyecto y el costo de desarrollo del mismo.

Fase de elaboración: Se especifican los casos de uso y se diseña la arquitectura del sistema a modo de vistas de todos los modelos del sistema.

Fase de Construcción: En esta se crea el producto, en donde la descripción inicial evoluciona en un producto preparado para ser otorgado a la comunidad de usuarios. Cada vez que se termine esta fase se debe de consultar si el producto cubre las necesidades de los usuarios de manera suficiente como para hacer la entrega del mismo.

Fase de Transición: Se cubre el período durante el cual el producto se convierte en versión beta. Esta versión beta es la etapa en donde un grupo de usuarios experimentados prueba el producto e informa de las deficiencias y eventuales errores que tenga el sistema. De esta manera los desarrolladores pueden corregir los problemas y mejorar las versiones del producto.

Cada una de estas fases se subdivide a su vez en otras tareas, Requisitos, Análisis, Diseño, Implementación y Prueba. Donde cada uno de los ciclos genera una nueva versión del sistema, siendo estas un producto listo para la entrega. Constan de códigos fuentes, manuales y otros productos asociados. Sin embargo estos productos se deben ajustar no sólo a los requerimientos del cliente sino que también a la de la gente que va a trabajar con él.

Los productos terminados incluyen los requisitos, casos de uso, especificaciones no funcionales y casos de prueba. Debe de incluir el modelo de la arquitectura y el modelo visual.

2.3.6 Elección de Paradigma

De acuerdo a los estudios realizados con respecto a los paradigmas se puede decir, que cada uno de ellos cuenta con ciertas ventajas y desventajas que los hacen óptimos de ser aplicables en determinadas situaciones.

Basándose en este estudio se ha descartado el paradigma de Ciclo de Vida Clásico, ya que este modelo presenta principalmente dificultad en la obtención de los requisitos, ya que para el cliente es muy difícil establecer sus necesidades, el que en muchas ocasiones no sabe lo que realmente necesita para su sistema, además brinda la desventaja de que la versión no está disponible hasta avanzado del proyecto, por lo cual el cliente no tiene una visión de lo que será el sistema terminado, lo que hace que muchas veces se impaciente por obtener un resultado rápido. Otra de sus desventajas es que se producen demoras muchas veces por parte de los desarrolladores, ya que para avanzar a una próxima etapa se debe haber terminado la anterior. En muchas ocasiones para mayor eficiencia de este paradigma se requiere no seguir el modelo secuencial propiamente tal como esta descrito.

En cuanto al Modelo en Espiral se ha descartado debido a ser un modelo que principalmente tiene la desventaja del modelo que requiere una gran habilidad para determinar los riesgos del proyecto y en el cual se basa el éxito del proyecto, en general este paradigma es bueno desde el punto de vista que es iterativo e incremental y que se basa en el modelo de prototipo.

Con respecto al modelo de prototipos las principales desventajas con que cuenta son que muchas veces el cliente piensa que el prototipo mostrado en una primera versión corresponde al producto final, lo que en realidad no es así ya que es sólo una muestra de lo que puede llegar a ser el sistema, siendo este prototipo descartado y construido totalmente de nuevo. Otra de las desventajas con las cuales cuenta este paradigma es en muchas ocasiones por querer mostrar un prototipo al cliente se utilice un lenguaje que no es el más apropiado.

Después de tal evaluación se puede decir que el paradigma a utilizar en este proyecto es el *Paradigma de Proceso Unificado de Desarrollo de Software de Rational* el cual es un paradigma basado en la construcción de prototipos, por lo cual toma aquellas ventajas del modelo de prototipos, además el que tiene como ventaja el ser evolutivo.

Esto permite poder refinar los requerimientos hechos por el cliente, como así también generar "mini proyectos" los cuales serán refinados a lo largo del desarrollo del proyecto, dependiendo de los nuevos requisitos dados por el cliente, como los cambios generados por la propia evolución del sistema. Además brinda las ventajas de:

La iteración controlada reduce el coste de riesgo a los costes de un sólo incremento.

Reduce el riesgo de no cumplir con los compromisos adquiridos dentro de un período determinado.

Acelera el ritmo total del esfuerzo del desarrollo, ya que se trabaja para conseguir metas claras a corto plazo.

Los requerimientos del usuario, en vez de ser definidos en forma temprana, se van refinando a medida que se avanza en las iteraciones, haciendo más fácil la adaptación a los requisitos cambiantes.

2.4 Estudio y Elección de Metodología

2.4.1 Análisis Estructurado

Tiene como principal desventaja la falta de herramientas efectivas para solucionar los problemas relativos al desarrollo del sistema. Y estas herramientas son ineficaces e ineficientes, lo que va en desmedro de la especificación de requerimientos; esto es, que no otorga los elementos necesarios para conformar una especificación de requerimientos que refleje efectivamente las características del sistema desde el punto de vista de los datos y las alteraciones que se puedan hacer a ellos. Esto deriva en el uso de la narración escrita en lenguaje natural, aún así es particularmente difícil de comprender las relaciones entre los elementos que conforman aquello que está siendo descrito. En estos términos la especificación resulta redundante por las características propias de los lenguajes naturales.

2.4.2 Análisis Estructurado Moderno

Se basa fundamentalmente en el uso de gráficas para reflejar el sistema. Los modelos utilizados en esta metodología son del tipo redes o grafos, con lo cual se pretende representar los flujos de datos y procesos que circulan en el sistema. Estos grafos (DFD) son los que se utilizan para modelar el comportamiento de los datos al interior del sistema. Estos diagramas se utilizan para representar el sistema en cada una de las fases de la etapa de análisis en que son requeridos, es decir, cuando se describe el sistema tal como funciona actualmente, y cuando se describe el nuevo sistema y su estructura.

2.4.3 Análisis Orientado a Objetos

El Análisis Orientado a Objetos maneja métodos que permiten al ingeniero de software modelar un problema a través de la representación de objetos, atributos y operaciones como las componentes primarias del modelado. Una amplia variedad de métodos de análisis orientado a objetos han sido propuestos, pero todos poseen un conjunto de características posibles:

Representación de clases o jerarquías de clases.

Creación de modelos objeto-relación.

Derivación de modelos objeto-comportamiento.

El análisis de sistemas orientados a objeto ocurre a muchos niveles diferentes de abstracción. Al nivel de negocios o empresarial, las técnicas asociadas con AOO pueden acoplarse con un enfoque de ingeniería de la información. Esta técnica se denomina normalmente análisis de dominio. Al nivel de aplicación, el modelo de objetos se centra en los requisitos específicos del cliente, pues estos afectan a la aplicación que se va a construir.

El proceso de AOO comienza con la definición de los casos de uso, escenarios que describen como se debe usar el sistema OO. Se aplica entonces la técnica de modelado clase-responsabilidad-colaborador (CRC) a clases documentos, a sus atributos y operaciones. También aporta una vista inicial de las colaboraciones que ocurren entre los objetos. La etapa siguiente en el AOO es la clasificación de los objetos y la creación de una jerarquía de clases. Pueden usarse subsistemas (temas) para encapsular objetos relacionados. El modelo objeto-relación proporciona una indicación acerca de cómo están interconectadas unas clases con otras, y el modelo objeto-comportamiento indica el comportamiento de objetos individuales y el comportamiento global del sistema OO.

Los objetos modelan casi cualquier aspecto identificable del ámbito del problema: entidades externas, cosas, sucesos, papeles, unidades organizativas, lugares y estructuras, todas ellas pueden ser representadas como objetos.

- Para el desarrollo de ésta se consideran fundamentalmente los siguientes pasos:

- Identificar los objetos considerando que los depósitos de datos que aparecen en los Diagramas de Flujo de Datos.

- Identificar las operaciones asociadas a cada objeto, para lo cual se realizan los procesos y se determina a qué objetos potenciales la asociación resulta más natural. Para ello es necesario considerar que:

Algunos candidatos a objetos no tienen operaciones que los acrediten como susceptibles de convertirse en objetos.

Algunos procesos presentan correspondencia con más de un objeto, caso en el cual es necesario descomponer esos procesos.

Habrá procesos sin candidatos a objetos. En este caso, se crea un objeto para cada uno de estos procesos.

• Representar objetos identificados, operaciones asociadas a entidades externas, y ajustar los diagramas de flujo de datos a las modificaciones introducidas.

• Definir las interfaces de los objetos.

• Realizar una evaluación final.

2.4.4 Elección de la Metodología

La elección de la metodología debe ser acorde al proyecto a desarrollar, así como también a su naturaleza, por lo tanto, se hace necesario utilizar la metodología de orientación a objetos, ya que esta nos facilita la tarea de poder identificar los elementos que componen un sistema, de esta manera, también reduce las distancias entre las actividades de análisis, puesto que trata los atributos y métodos de los elementos del sistema como un todo. A esto se le conoce como integración, y como esta se ve a lo largo del ciclo vital del sistema, entonces reduce los riesgos relativos al desarrollo de sistemas complejos. Además, esta metodología, permite al analista y al cliente poder tener una mejor comunicación ya que esta tiene herramientas de representación muy comprensibles para ambas partes.

Otro punto favorable de esta metodología, es lo explícita que puede ser al representar los puntos comunes, referente a la herencia de los atributos y métodos, permitiendo de esta manera la definición y construcción de objetos capaces de tener y utilizar métodos y atributos heredados provenientes de un objeto más grande, disminuyendo así la codificación del sistema. Con esta característica, también se puede sacar provecho de los distintos lenguajes disponibles basados en objetos y en la misma metodología, debido a la expresividad de los mismos.

Cuando en un sistema se produce un cambio, generalmente éste afecta a gran parte del mismo. El Análisis de Orientación a Objetos permite crear especificaciones que sean capaces de tolerar los cambios, ya que esta metodología hace que las estructuras de dominio del sistema sean dinámicas, otorgando la estabilidad suficiente en caso de cambios de requisitos de sistemas similares. Por lo tanto podemos decir que la Metodología de Análisis de Orientación a Objetos produce sistemas que son flexibles a los cambios.

Y una de las características esenciales de esta metodología, es que con toda la modulación de los componentes de un sistema, hace que los mismos sean componentes reutilizables, así como también los diseños de ellos, y por consiguiente las mismas aplicaciones, simplificando la tarea de los desarrolladores de crear nuevas aplicaciones o componentes para un sistema distinto.

Por lo anteriormente expuesto se considera que esta metodología es la más acorde para trabajar el proyecto académico, ya que entrega herramientas fáciles de entender tanto para los clientes como para los desarrolladores. Además de otorgar un conocimiento más acabado para los alumnos que participan en él.

2.5 Estudio y Elección de las Herramientas

2.5.1 Lenguajes de Programación

a) ASP.NET

ASP.NET, también llamado ASPX, es un lenguaje de programación generado en Common Language Runtime que puede utilizarse en un servidor para generar eficaces aplicaciones Web. Puede ser escrito en cualquier lenguaje soportado por el .net Framework, es decir: VB.net; C# y JScript.net. Otra característica fundamental es que ASP.net es un lenguaje totalmente orientado a objetos.

Ventajas:

Mejor rendimiento. ASP.NET es un código de Common Language Runtime compilado que se ejecuta en el servidor. A diferencia de sus predecesores, ASP.NET puede aprovechar las ventajas del enlace anticipado, la compilación just-in-time, la optimización nativa y los servicios de caché desde el primer momento. Esto supone un incremento espectacular del rendimiento antes de siquiera escribir una línea de código.

Compatibilidad con herramientas de primer nivel. El marco de trabajo de ASP.NET se complementa con un diseñador y una caja de herramientas muy completos en el entorno integrado de programación (Integrated Development Environment, IDE) de Visual Studio. La edición WYSIWYG (lo que ves es lo que obtienes), los controles de servidor de arrastrar y colocar y la implementación automática son sólo algunas de las características que proporciona esta eficaz herramienta.

Eficacia y flexibilidad. Debido a que ASP.NET se basa en Common Language Runtime, la eficacia y la flexibilidad de toda esa plataforma se encuentra disponible para los programadores de aplicaciones Web. La biblioteca de clases de .NET Framework, la mensajería y las soluciones de Acceso a datos se encuentran accesibles desde el Web de manera uniforme.

ASP.NET es también independiente del lenguaje, por lo que puede elegir el lenguaje que mejor se adapte a la aplicación o dividir la aplicación en varios lenguajes. Además, la interoperabilidad de Common Language Runtime garantiza que la inversión existente en programación basada en COM se conserva al migrar a ASP.NET.

Simplicidad. ASP.NET facilita la realización de tareas comunes, desde el sencillo envío de formularios y la autenticación del cliente hasta la implementación y la configuración de sitios. Por ejemplo, el marco de trabajo de página de ASP.NET permite generar interfaces de usuario, que separan claramente la lógica de aplicación del código de presentación, y controlar eventos en un sencillo modelo de procesamiento de formularios de tipo Visual Basic. Además, Common Language Runtime simplifica la programación, con servicios de código administrado como el recuento de referencia automático y el recolector de elementos no utilizados.

Facilidad de uso. ASP.NET emplea un sistema de configuración jerárquico, basado en texto, que simplifica la aplicación de la configuración al entorno de servidor y las aplicaciones Web. Debido a que la información de configuración se almacena como texto sin formato, se puede aplicar la nueva configuración sin la ayuda de herramientas de administración local. Esta filosofía de "administración local cero" se extiende asimismo a la implementación de las aplicaciones ASP.NET Framework. Una aplicación ASP.NET Framework se implementa en un servidor sencillamente mediante la copia de los archivos necesarios al servidor. No se requiere el reinicio del servidor, ni siquiera para implementar o reemplazar el código compilado en ejecución.

Escalabilidad y disponibilidad. ASP.NET se ha diseñado teniendo en cuenta la escalabilidad, con características diseñadas específicamente a medida, con el fin de mejorar el rendimiento en entornos agrupados y de múltiples procesadores. Además, el motor de tiempo de ejecución de ASP.NET controla y administra los procesos de cerca, por lo que si uno no se comporta adecuadamente (filtraciones, bloqueos), se puede crear un proceso nuevo en su lugar, lo que ayuda a mantener la aplicación disponible constantemente para controlar solicitudes.

Posibilidad de personalización y extensibilidad. ASP.NET presenta una arquitectura bien diseñada que permite a los programadores insertar su código en el nivel adecuado. De hecho, es posible extender o reemplazar cualquier subcomponente del motor de tiempo de ejecución de ASP.NET con su propio componente escrito personalizado. La implementación de la autenticación personalizada o de los servicios de estado nunca ha sido más fácil.

Seguridad. Con la autenticación de Windows integrada y la configuración por aplicación, se puede tener la completa seguridad de que las aplicaciones están a salvo.

Desventajas:

Dependencia de un solo proveedor.

Por ser un cambio muy fuerte en arquitectura, puede contener los problemas de primeras versiones.

Al poder combinar múltiples lenguajes, puede dar lugar a código mantenible sólo por ciertas personas. Poco reaprovechamiento de la experiencia de recursos humanos especializados en Microsoft, ya que cambia drásticamente la plataforma. Monopoliza las herramientas de desarrollo.

b) JAVA

El lenguaje de programación Java, fue diseñado por la compañía Sun Microsystems Inc, con el propósito de crear un lenguaje que pudiera funcionar en redes computacionales heterogéneas (redes de computadoras formadas por más de un tipo de computadora, ya sean PC, MAC's, estaciones de trabajo, etc.), y que fuera independiente de la plataforma en la que se vaya a ejecutar. Esto significa que un programa de Java puede ejecutarse en cualquier máquina o plataforma. El lenguaje fue diseñado con las siguientes características en mente:

Orientado a objetos: Java fue diseñado como un lenguaje orientado a objetos desde el principio. Los objetos se agrupan en estructuras encapsuladas, tanto sus datos como los métodos (o funciones) que manipulan esos datos. La tendencia del futuro, a la que Java se suma, apunta hacia la programación orientada a objetos, especialmente en entornos cada vez más complejos y basados en red.

Distribuido: Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.

Interpretado y compilado a la vez: Java es compilado, en la medida en que su código fuente se transforma en una especie de código máquina, los bytecodes, semejantes a las instrucciones de desamblador. Por otra parte, es interpretado, ya que los bytecodes se pueden ejecutar directamente sobre cualquier máquina a la cual se hayan portado el intérprete y el sistema de ejecución en tiempo real (run-time).

Robusto: Java fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores (la aritmética de punteros), ya que se ha prescindido por completo los punteros, y la recolección de basura elimina la necesidad de liberación explícita de memoria.

Seguro: Dada la naturaleza distribuida de Java, donde las applets se bajan desde cualquier punto de la red, la seguridad se impuso como una necesidad de vital importancia. A nadie le gustaría ejecutar en su ordenador programas con acceso total a su sistema, procedentes de fuentes desconocidas. Así que se implementaron barreras de seguridad en el lenguaje y en el sistema de ejecución en tiempo real.

Indiferente a la arquitectura: Java está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix a Windows Nt, pasando por Mac y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos. Para acomodar requisitos de ejecución tan variados, el compilador de Java genera bytecodes: un formato intermedio indiferente a la arquitectura diseñada para transportar el código eficientemente a múltiples plataformas hardware y software. El resto de problemas los soluciona el intérprete de Java.

Portable: La indiferencia a la arquitectura representa sólo una parte de su portabilidad. Además, Java especifica los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos, de manera que los programas son iguales en todas las plataformas.
Estas dos últimas características se conocen como la *Máquina Virtual Java* (JVM).

Multihebra: Hoy en día ya se ven como terriblemente limitadas las aplicaciones que sólo pueden ejecutar una acción a la vez. Java soporta sincronización de múltiples hilos de ejecución (*multithreading*) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Así, mientras un hilo se encarga de la comunicación, otro puede interactuar con el usuario mientras otro presenta una animación en pantalla y otro realiza cálculos.

Dinámico: El lenguaje Java y su sistema de ejecución en tiempo real son dinámicos en la fase de enlazado. Las clases sólo se enlazan a medida que son necesitadas. Se pueden enlazar nuevos módulos de código bajo demanda, procedente de fuentes muy variadas, incluso desde la red.

Produce applets: Java puede ser usado para crear dos tipos de programas: aplicaciones independientes y applets. Las aplicaciones independientes se comportan como cualquier otro programa escrito en cualquier lenguaje, como por ejemplo el navegador de Web HotJava, escrito íntegramente en Java.

Por su parte, las applets son pequeños programas que aparecen insertados en las páginas Web, como aparecen los gráficos o el texto, pero con la capacidad de ejecutar acciones muy complejas, como animar imágenes, establecer conexiones de red, presentar menús y cuadros de diálogo para luego emprender acciones.

Ventajas:

Es un lenguaje relativamente sencillo y fácil de aprender.
Es muy productivo, con poco código puedes hacer mucho más que la misma cantidad en código de C ó C++.
Posee todas las ventajas de la programación orientada a objetos.
El JDK es una herramienta libre de licencias (sin costo), creada por Sun.- Está respaldado por un gran número de proveedores.
Existe soporte dado por Sun.
Debido a que existen diferentes productos de Java, hay más de un proveedor de servicios. Sun saca al mercado cada 6 meses una nueva versión del JDK.
Es independiente de la plataforma de desarrollo.
Existen dentro de su librería clases gráficas como awt y swing, las cuales permiten crear objetos gráficos comunes altamente configurables y con una arquitectura independiente de la plataforma.

Java permite a los desarrolladores aprovechar la flexibilidad de la Programación Orientada a Objetos en el diseño de sus aplicaciones.

Se puede acceder a bases de datos fácilmente con JDBC, independientemente de la plataforma utilizada.

Desventajas:

Java es lento comparado con C ó C++, pero en algunas tareas es casi igual de veloz.

Java no permite el acceso directo a todos los recursos de hardware de la PC. Aunque esto también podría ser una ventaja ya que así es más fácil.

Hay diferentes tipos de soporte técnico para la misma herramienta, por lo que el análisis de la mejor opción se dificulta.

Para manejo a bajo nivel deben usarse métodos nativos, lo que limita la portabilidad.

El diseño de interfaces gráficas con awt y swing no es simple.

Actualmente hay tantas bibliotecas de clases para java que es casi imposible conocer para que sirven todas y cada una de ellas, es difícil aprenderlas todas.

c) Visual Basic

Visual Basic es uno de los tantos lenguajes de programación que podemos encontrar hoy en día. Dicho lenguaje nace del BASIC (Beginner´s All-purpose Symbolic Instruction Code) que fue creado en su versión original en el Dartmouth College, con el propósito de servir a aquellas personas que estaban interesadas en iniciarse en algún lenguaje de programación. Luego de sufrir varias modificaciones, en el año 1978 se estableció el BASIC estándar. La sencillez del lenguaje ganó el desprecio de los programadores avanzados por considerarlo "un lenguaje para principiantes".

Primero fue GW-BASIC, luego se transformó en QuickBASIC y actualmente se lo conoce como Visual Basic y la versión más reciente es la 6 que se incluye en el paquete Visual Studio 6 de Microsoft. Esta versión combina la sencillez del BASIC con un poderoso lenguaje de programación Visual que juntos permiten desarrollar robustos programas de 32

bits para Windows. Esta fusión de sencillez y la estética permitió ampliar mucho más el monopolio de Microsoft, ya que el lenguaje sólo es compatible con Windows, un sistema operativo de la misma empresa.

Visual Basic ya no es más "un lenguaje para principiantes" sino que es una perfecta alternativa para los programadores de cualquier nivel que deseen desarrollar aplicaciones compatibles con Windows.

Características de Visual Basic.

Diseñador de entorno de datos: Es posible generar, de manera automática, conectividad entre controles y datos.

Los Objetos Actives son una nueva tecnología de acceso a datos mediante la acción de arrastrar y colocar datos sobre formularios o informes.

Asistente para formularios: Sirve para generar de manera automática formularios que administran registros de tablas o consultas pertenecientes a una base de datos, hoja de cálculo u objeto (ADO-ACTIVE DATA OBJECT).

Asistente para barras de herramientas: Es factible incluir barra de herramientas personalizada, donde el usuario selecciona los botones que desea visualizar durante la ejecución.

En las aplicaciones HTML: Se combinan instrucciones de Visual Basic con código HTML para controlar los eventos que se realizan con frecuencia en una página Web.

Ofrece la manera más rápida y fácil de crear aplicaciones para Windows. Es un completo juego de herramientas que simplifican el RAD, tanto para programadores expertos como para noveles.

La palabra “Visual” en Visual Basic se refiere al método de crear las pantallas (GUI). En lugar de escribir numerosas líneas de código para describir la apariencia y posición de los controles en pantalla, se arrastra y se colocan elementos prefabricados dentro de su posición en la pantalla. “Basic” se refiere al lenguaje de programación Basic. Visual Basic es una evolución del lenguaje Basic original. Los novatos pueden crear aplicaciones útiles

aprendiendo solo algunos métodos. Así y todo Visual Basic es lo suficientemente potente como para competir con otras herramientas de desarrollo más sofisticadas. Visual Basic provee de las siguientes funcionalidades:

RAD (Rapid Application Development).

Acceso a base de datos, para crear aplicaciones rápidas, de un alto rendimiento y componentes.

Desarrollo en equipo y compatibilidad con Visual Modeler, Visual Database Tools, Visual SourceSafe, y SQL Server.

Desarrollar para una plataforma como Internet sin abandonar el código existente ni los conocimientos de desarrollo.

La habilidad de crear componentes ActiveX para cliente servidor, Internet, y Microsoft Transaction Server.

Código nativo optimizado para un alto rendimiento, y posibilidad de optimización para Pentium Pro.

Ventajas

Puede ser utilizado fácilmente para crear aplicaciones Windows.

Tiene una curva de aprendizaje corta.

El soporte técnico es otorgado por Microsoft.

El desarrollo de las interfaces gráficas es rápido y sencillo debido a su conjunto de objetos gráficos.

Aunque no es una herramienta totalmente Orientada a Objetos, maneja Clases y Polimorfismo.

Se puede acceder a una base de datos fácilmente mediante los mecanismos del ADO, OLE DB u ODBC.

Existen las herramientas Crystal Reports para generar reportes.

Desventajas

Su costo es alto, debido a que Microsoft impone un esquema en el que es necesario comprar más de una licencia.
El soporte se limita a problemas específicos de la herramienta.
Depende completamente de la plataforma Windows.
Tiene características de un lenguaje Orientado a Objetos, pero no maneja Herencia, sin embargo puede simularse, aunque su implementación no es simple.

2.5.2 Lenguaje de Programación Elegido

Hypertext Preprocessor o más conocido como PHP, es un lenguaje "open source" interpretado de alto nivel incrustado en páginas HTML y ejecutado en el servidor.

No es lo mismo que un script escrito en otro lenguaje de programación como Perl o C ++. En vez de escribir un programa con muchos comandos para crear una salida en HTML, se escribe el código HTML con cierto código PHP introducido en el mismo, el que producirá cierta salida.

La diferencia a PHP de la tecnología Javascript, la cual se ejecuta en la máquina cliente, es que el código PHP es ejecutado en el servidor.

Lo mejor de usar PHP es que es extremadamente simple para el principiante, pero a su vez, ofrece muchas características avanzadas para los programadores profesionales.

Aunque el desarrollo de PHP está concentrado en la programación de scripts en la parte del servidor, se puede utilizar para muchas otras cosas. Puede hacer cualquier cosa que se pueda hacer con un script CGI, como procesar la información de formularios, generar páginas con contenidos dinámicos, o mandar y recibir cookies. Existen tres campos en los que scripts escritos en PHP son usados:

Scripts en la parte del servidor: Este es el campo más tradicional y el principal campo de trabajo. Se necesitan tres cosas para que esto funcione. El compilador PHP (CGI ó módulo), un servidor web y un navegador. Se necesita correr el servidor web con PHP

instalado. El resultado del programa PHP se puede obtener a través del navegador, conectando con el servidor web.

Scripts en línea de comandos: Se puede crear un script PHP y correrlo sin ningún servidor web ó navegador. Solamente se necesita el compilador PHP para usarlo de esta manera. Estos scripts también pueden ser usados para tareas simples de procesado de texto.

Escribir aplicaciones gráficas clientes: PHP permite además escribir aplicaciones gráficas, utilizando algunas características avanzadas en programas clientes, por ejemplo se puede utilizar PHP-GTK para escribir dichos programas.

PHP puede ser utilizado en cualquiera de los principales sistemas operativos, incluyendo Linux, variantes Unix (HP-UX, Solaris y OpenBSD), Microsoft Windows, Mac OS X, RISC OS etc. PHP soporta la mayoría de servidores web de hoy en día, incluyendo Apache, Microsoft Internet Information Server, Personal Web Server, Netscape y iPlanet, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd y muchos otros. PHP tiene módulos disponibles para la mayoría de los servidores, para aquellos otros que soporten el estándar CGI, PHP puede usarse como procesador CGI.

Otra de las ventajas que posee PHP es la programación orientada a objetos. En donde muchas librerías y aplicaciones grandes (incluyendo la librería PEAR) están escritas íntegramente usando programación orientada a objetos.

Otras de sus habilidades son la creación de imágenes, ficheros pdf y películas Flash sobre la marcha. También se pueden presentar resultados, como XHTM y ficheros XML. PHP puede autogenerar estos ficheros y grabarlos en el sistema de ficheros en vez de presentarlos en la pantalla.

Quizás la característica más destacable de PHP es su soporte para una gran cantidad de bases de datos. Poder escribir una interfaz vía web para una base de datos es una tarea simple con PHP.

Algunas de las bases de datos que soporta PHP son: MySql, Postgres, Empress FrontBase PostgreSQL, IBM DB2 MySQL Velocis, Informix ODBC Unix dbm, además de muchas más que existen en la actualidad.

Ventajas:

Muy sencillo de aprender.

Similar en sintaxis a C y a PERL.

Soporta en cierta medida la orientación a objeto. Clases y herencia.

El análisis léxico para recoger las variables que se pasan en la dirección lo hace PHP de forma automática. Librándose el usuario de tener que separar las variables y sus valores.

Se puede incrustar código PHP con etiquetas HTML.

Excelente soporte de acceso a base de datos.

La comprobación de que los parámetros son validos se hace en el servidor y no en el cliente (como se hace con javascript) de forma que se puede chequear que no se reciban solicitudes adulteradas. Además PHP viene equipado con un conjunto de funciones de seguridad que previenen la inserción de órdenes dentro de una solicitud de datos.

Desventajas:

Todo el trabajo lo realiza el servidor y no delega al cliente. Por tanto puede ser más ineficiente a medida que las solicitudes aumenten de número.

La legibilidad del código puede verse afectada al mezclar sentencias HTML y php.

La orientación a objetos es aún muy deficiente para aplicaciones grandes.

2.5.3 Motores de Base de Datos

a) SQL Server

La estrategia de Microsoft es la de hacer que SQL Server sea la base de datos más fácil de utilizar para construir, administrar e implementar aplicaciones de negocios. Esto significa tener que poner a disposición un modelo de programación rápido y sencillo para desarrolladores, eliminando la administración de base de datos para operaciones estándar, y suministrando herramientas sofisticadas para operaciones más complejas.

Disminuye el costo total de propiedad a través de características como administración multi-servidor y con una sola consola; ejecución y alerta de trabajos basadas en eventos; seguridad integrada; y scripting administrativo. En la versión 7.0 también libera al administrador de base de datos para aspectos más sofisticados del trabajo al automatizar las tareas de rutina. Al combinar estos servicios de administración con las nuevas características de configuración automática permite la automatización de sucursales y aplicaciones de base de datos insertadas.

Las innovaciones de SQL Server 7.0 que mejoran el proceso de data warehousing son los Servicios de Transformación de Datos, manejo mejorado de las consultas complejas y bases de datos muy grandes, procesamiento analítico en línea e integrado, y el Microsoft Repository. Otro componente esencial es el soporte extenso para integración de terceros. Algunos aspectos que se deberían considerar:

Escalabilidad: Se adapta a las necesidades de la empresa, soportando desde unos pocos usuarios a varios miles. Empresas centralizadas u oficinas distribuidas, replicando cientos de sites.

Potencia: Microsoft SQL Server es la mejor base de datos para Windows NT Server. Posee los mejores registros de los benchmarks independientes (TCP) tanto en transacciones totales como en coste por transacción.

Gestión: Con un completo interfaz gráfico que reduce la complejidad innecesaria de las tareas de administración y gestión de la base de datos.

Orientada al desarrollo: Visual Basic, Visual C++, Visual J++, Visual Interdev, Microfocus Cobol y muchas otras herramientas son compatibles con Microsoft SQL Server.

Gestión y administración centralizada de bases de datos.

SQL Enterprise Manager, una consola de gestión y motorización 32-bit visual basada en Windows.
Un único punto de configuración y gestión de control de datos remotos.
SQL Executive, planificador de trabajos y monitor para gestión proactiva de servidores distribuidos.
Scripts Visual Basic a través de SQL-Distributed Management Objects (SQL-DMO) basados en OLE.
DBA Assistant, para el mantenimiento automático rutinario en una única tarea planificada.
SQL Trace, para monitorizar consultas cliente-servidor mediante SQL almacenadas en archivos de registros.

Disponibilidad, fiabilidad y tolerancia a fallos.

Mirroring de dispositivos de base de datos con failover automático para tolerancia a fallos de dispositivos.
Copias de seguridad online desatendidas garantizando la consistencia de datos para la más alta disponibilidad.
Contextos de usuario protegidos, que pueden aislar los fallos a un thread de un único usuario.
Tolerancia a fallos de servidor, permitiendo failover automático a un servidor de backup o en espera.

Seguridad.

Un único ID de login tanto para red como para la DB para mejorar la seguridad y facilitar la administración.

Password y encriptación de datos en red para mejorar la seguridad.

Encriptación de procedimientos almacenados para la integridad y seguridad de código de aplicación.

Interoperabilidad e integración con desktops.

Gateway Open Data Services (ODS) programable para acceso transparente a fuentes de datos externas.

2.5.4 Motor de Base de Datos Elegido

a) PostgreSQL

Este motor de base de datos, ofrece un control de la concurrencia multi-versión, soportando casi toda la sintaxis SQL (incluyendo subconsultas, transacciones, y tipos y funciones definidas por el usuario), contando también con un amplio conjunto de enlaces con lenguajes de programación (incluyendo C, C++, Java, perl, tcl y python).

Lanzada al mercado como "Postgres95" su nombre fue cambiado a PostgreSQL, para reflejar la relación entre el Postgres original y las versiones más recientes con capacidades SQL. Postgres tiene algunas características que son propias del mundo de las bases de datos orientadas a objetos. De hecho, algunas Bases de Datos comerciales han incorporado recientemente características en las que Postgres fue pionera. Muchas organizaciones, incluyendo grandes corporaciones, instituciones gubernamentales y pequeños negocios en línea usan PostgreSQL para manejar sus datos más valiosos y aplicaciones de misión crítica.

Durante el desarrollo de Postgres95 se hizo hincapié en identificar y entender los problemas en el código del motor de datos. Con PostgreSQL, el énfasis ha pasado a aumentar características y capacidades, aunque el trabajo continúa en todas las áreas.

Ventajas:

Por su arquitectura de diseño, escala muy bien al aumentar el número de CPUs y la cantidad de RAM.

Soporta transacciones y desde la versión 7.0, claves ajenas (con comprobaciones de integridad referencial).

Tiene mejor soporte para triggers y procedimientos en el servidor.

Soporta un subconjunto de SQL92 mayor que el que soporta MySQL. Además, tiene ciertas características orientadas a objetos.

Poder instalar un número ilimitado de veces sin temor de sobrepasar la cantidad de licencias, la principal preocupación de muchos proveedores de bases de datos comerciales

Velocidad y rendimiento excepcionales

Confiabilidad a toda prueba

Seguridad de primera clase

Flexibilidad para extenderse según se requiera

Diseño altamente escalable

Red mundial de Proveedores Independientes de Software (ISV)

Muchas opciones de soporte

Mínimos requerimientos de administración

Bajo Costo Total de Operación (TCO)

Concordancia a Estándares ANSI.

Desventajas:

Consume bastantes más recursos y carga más el sistema.

Límite del tamaño de cada fila de las tablas a 8k (se puede ampliar a 32k recompilando, pero con un coste añadido en el rendimiento).

Es de 2 a 3 veces más lenta que MySQL.

2.5.5 Estudio de Herramienta CASE Rational Rose

A medida que los sistemas que hoy se construyen se tornan más y más complejos, las herramientas de modelado con UML ofrecen muchos beneficios para todos los involucrados en un proyecto, por ejemplo, administrador del proyecto, analistas, arquitectos, desarrolladores y otros. Las herramientas CASE de modelado con UML nos permiten aplicar la metodología de análisis y diseño orientados a objetos y abstraernos del código fuente, en un nivel donde la arquitectura y el diseño se tornan más obvios y más fáciles de entender y modificar. Cuanto más grande es un proyecto, es más importante utilizar una herramienta CASE. Al usar las herramientas CASE:

• Los Analistas de Negocio/ Sistemas pueden capturar los requisitos del negocio/sistema con un modelo de casos de uso

• Los Diseñadores/Arquitectos pueden producir el modelo de diseño para articular la interacción entre los objetos o los subsistemas de la misma o de diferentes capas (los diagramas UML típicos que se crean son los de clases y los de interacción)

• Los Desarrolladores pueden transformar rápidamente los modelos en una aplicación funcionando, y buscar un subconjunto de clases y métodos y asimilar el entendimiento de cómo lograr interfaces con ellos. El modelo actúa como el plano y guiará finalmente la construcción del sistema. De manera semejante, la administración es capaz de ver, puntualmente y desde un alto nivel, una representación del diseño y comprender lo que está sucediendo.

Por estas razones, las herramientas CASE de UML acompañadas con metodologías, nos brindan una forma de representar sistemas demasiados complejos para comprenderlos a través de su código fuente subyacente y nos permiten desarrollar la solución de software correcta más rápido y más económicamente.

Sin embargo, las herramientas CASE varían con respecto a las capacidades de modelado con UML, el soporte del ciclo de vida del proyecto, las ingenierías directa y reversa, el modelado de datos, la performance, el precio, el soporte, la facilidad de uso, etc. Este artículo explorará las concordancias y las diferencias entre Rose y EA en áreas tales como

el modelado con UML, el soporte del ciclo de vida del proyecto y la ingeniería de código, y espera asistirlo para elegir la herramienta correcta para su proyecto.

Características de modelado con UML

El UML estándar está compuesto por tres partes: bloques de construcción (tales como clases, objetos, mensajes), relaciones entre los bloques (tales como asociación, generalización) y diagramas (por ejemplo, diagrama de actividad). Los perfiles del UML son las extensiones a las notaciones estándares del UML usando los mecanismos de extensión del UML: los estereotipos, los valores etiquetados y las restricciones.

Rose Enterprise V.2002.05 es compatible con el UML 1.4 y soporta ocho de los nueve diagramas estándares del UML: diagrama de casos de uso, de clases, de secuencia, de colaboración, de actividad, de estados, de implementación (componentes), de despliegue y varios perfiles del UML. Si fuera necesario, el diagrama de objetos se puede crear usando los diagramas de colaboración.

Ingeniería de Código

La ingeniería de código consiste en la ingeniería directa -desde el modelo al código- y en la ingeniería reversa -desde el código al modelo-. Una vez que se completa el diseño (los modelos de diseño y de datos) se puede usar la información del modelo para generar el código fuente en un lenguaje de programación específico o los scripts DDL para una base de datos.

A medida que los desarrolladores agregan o modifican el código o la implementación de la base de datos, se pueden sincronizar el diseño y el modelo de datos con el código o con el script DDL para mantenerlos consistentes entre sí.

La forma en la que trabaja es generando los archivos de código fuente de las clases para aquellas que correspondan al mismo paquete. También habilita asistentes para crear clases y provee plantillas de código que pueden aumentar significativamente la cantidad de código fuente generada. Adicionalmente, se

pueden aplicar los patrones de diseño que Rose ha provisto 20 de los patrones de diseño GOF para Java.

Soporte del Ciclo de Vida del Proyecto

Las herramientas CASE deberían apoyar todos los miembros del equipo en el desarrollo de sus tareas. En cuanto al soporte del ciclo de vida del proyecto, Rose es principalmente una herramienta de modelado y se puede integrar con otras herramientas de Rational o de terceras partes, tales como RequisitePro, Test Manager, SoDA, MS Word, MS Project, para lograr los mismos objetivos.

2.5.6 Lenguaje de Modelado Elegido

UML nació inicialmente como un lenguaje, en paralelo con la creación del paradigma de Proceso Unificado de Desarrollo de Software de Rational , es en sí más que un lenguaje de programación, es un lenguaje para modelar, que es el procedimiento que emplean los ingenieros para el diseño de software antes de pasar a su construcción, al igual que sucede con cualquier producto manufacturado o fabricado en serie.

UML ayuda tanto al desarrollador como al usuario a entender la realidad de la tecnología y la posibilidad de que reflexione antes de invertir y gastar grandes cantidades en proyectos que no estén seguros en su desarrollo, reduciendo el coste y el tiempo empleado en la construcción de las piezas que construirán el modelo.

Este lenguaje de modelado, como es una herramienta casi exclusiva del AOO, tiene la capacidad de agrupar varios componentes en uno y por lo tanto visualizar todo el sistema, de ser posible, de una sola vez, así como también poder analizar gráficamente uno por uno de los elementos que conforman el sistema. Por estas razones y por el complemento que brinda esta herramienta al paradigma elegido se utilizará en el desarrollo del sistema a implementar.

Capítulo 3: Fase de Inicio

3.1 Requerimientos Funcionales y No Funcionales de Usuario

En este punto se redactaran los distintos requerimientos, tanto funcionales como no funcionales, que los stakeholders después de diversas reuniones solicitaron y/o dieron a conocer para la ejecución de este sistema. En los requerimientos funcionales tenemos:

Permitir agregar nuevas fichas, tanto de alumnos, como del personal docente del establecimiento.
Permitir que en las fichas de alumnos y docentes presentes en el sistema puedan ser modificadas, en los campos relevantes y sensibles de sufrir algún cambio.
Permitir eliminar (dar de baja) a los alumnos que se retiraron del establecimiento educacional, y también dar de baja a los docentes que han sido despedidos o cambiados de lugar de trabajo.
Permitir acceder a los datos de un alumno en particular y de un profesor en particular.
Permitir la modificación de notas y asistencia del alumnado.
Crear asignaturas y cursos para el año académico.
Matricular alumnos a los distintos cursos creados.
Permitir asignar y modificar a un profesor a una asignatura.
Permitir asignar y modificar a un profesor jefe a un curso.
Permitir el registro de la asistencia diaria del alumnado.
Permitir la entrega total y porcentual de la asistencia por curso para efectos de Subvención.
Permitir el ingreso de evaluación del área de desarrollo de cada alumno, por semestre.
Permitir la generación de actas finales.

Permitir la aproximación de los promedios por asignatura semestre y anual.
Generar certificado de accidente escolar.
Generar informe de notas parciales por alumno.
Listar las notas parciales de un alumno obtenidas.
Calcular los promedios finales por asignatura.
Generar Certificado Anual de Estudios.
Generar Certificado de Matrícula.
Generar Informe de Rendimiento Escolar.
Listar alumnos por curso
Listar promedios por cursos
Permitir cierre de un periodo académico.

Requerimientos no Funcionales:

Desplegar las interfaces como páginas Web.
Cumplir con los estándares del RECH, para la generación de los archivos MINEDUC.
La generación de los listados no debe superar los 5 segundos.

3.2 Requerimientos Funcionales de Sistema (Modelo de Casos de Uso)

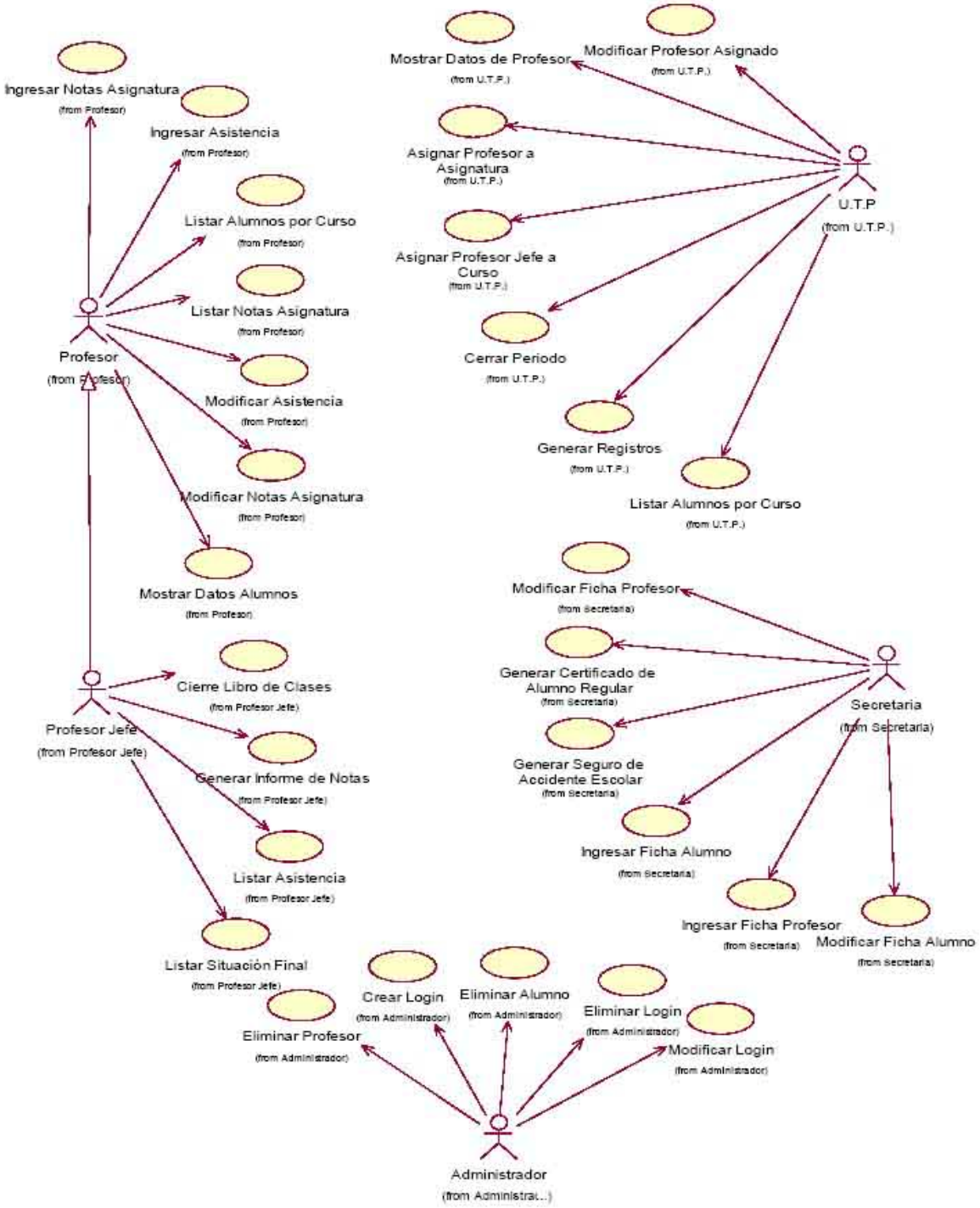

Ilustración 3-1 Diagrama de Casos de Uso General

3.3 Priorización de Casos de Uso

A continuación se mostrará la priorización de los casos de uso, eligiendo los de mayor relevancia y complejidad para el desarrollo del proyecto, el resto de los casos de uso no necesitan un mayor análisis, ya que estos son de mayor entendimiento para los desarrolladores.

Tabla 3-1 Actores del Sistema

Actor Secretaria	Actor Profesor	Actor Profesor Jefe	Actor UTP
• Ingresar Ficha Alumno. • Ingresar Ficha Profesor. • Generar Certificado de Alumno Regular. • Generar Seguro de Accidente Escolar.	• Ingresar Asistencia. • Modificar Asistencia. • Listar Alumnos por Curso.	• Cierre Libro de Clases. • Generar Informe de Notas.	• Cerrar Periodo. • Generar Registros. • Asignar Profesor a Asignatura. • Asignar Profesor Jefe a Curso.

3.4 Especificación de Casos de Uso

A continuación en la Tabla 9.1 se da a conocer la notación utilizada en la descripción de los Casos de Uso, esto para una mejor comprensión de ellos. Posteriormente en la Tabla 9.2 se da una descripción detallada de los eventos que intervienen en el Caso de Uso, para el resto de los Casos de Uso ver Anexo C.

Tabla 3-2 Notación de los Casos de Uso

Caso de Uso	Nombre del Caso de Uso

Actor Primario	Actor que requiere los servicios del sistema para cumplir sus metas
Participantes e Intereses	Se detalla con claridad los participantes que se definen en un Caso de Uso
Precondiciones	Condiciones que deben satisfacerse para poder ejecutar el Caso de Uso.
Poscondiciones	Condiciones que existirían después de ejecutar el Caso de Uso.
Escenario Principal	Descripción detallada de la interacción entre los Actores y el Sistema, narrada en forma secuencial considerando las acciones de los actores y las respuestas del sistema.
Extensiones	Alternativas que pueden ocurrir en el número de línea.
Requerimientos Especiales	Requerimientos no funcionales, atributos de calidad o restricciones.
Frecuencia de ocurrencia	El grado de ocurrencia.

Tabla 3-3 Descripción de Caso de Uso Ingresar Asistencia

Caso de Uso	Ingresar Asistencia.
Actor Primario	Profesor.
Participantes e Intereses	Profesor: rápido y preciso en el ingreso de asistencia.

Precondiciones	Profesor ha sido autentificado y reconocido por el sistema.
Poscondiciones	Profesor cierra sesión. Se han ingresado las asistencias de los alumnos del curso correspondiente.
Escenario Principal	Profesor elige la opción dentro del menú "Ingresar Asistencia". Se despliega por pantalla un cuadro para la elección del curso correspondiente. Profesor elige el curso. Se despliega herramienta para Ingresar Asistencia. Profesor ingresa el ticket por cada alumno que asistió ese día (filas) hasta el último alumno en la lista del curso. Se despliega por pantalla pidiendo la ratificación de la acción hecha. Profesor verifica la asistencia. Profesor ratifica la acción.
Extensiones	3. Profesor se equivoca en la elección del curso. 1. Profesor vuelve atrás e ingresa nuevamente el curso correspondiente. 8. Profesor no ratifica la opción, debido a un mal ingreso en

	la asistencia. Profesor vuelve a la fila del alumno correspondiente y modifica la asistencia.
Requerimientos Especiales	3. Se le proveerá a Profesor la opción de volver atrás, pudiendo así retroceder en la última acción realizada eliminando los cambios efectuados. 4. La herramienta se desplegará por pantalla de la forma de una lista de alumnos donde estarán todos los alumnos correspondientes al curso elegido por el profesor, además cada alumno será representado por una fila y los días serán registrados en las columnas, donde en cada cuadro el profesor tendrá la opción de poner "el ticket" para registrar si el alumno "Juan Pérez" vino el día "27 de Mayo del 2005". 5. La asistencia se pasará en cada asignatura del día reflejándola en el libro de clases, pero el último profesor del horario del curso será el encargado de registrar la asistencia del curso en el sistema.
Frecuencia de ocurrencia	Continuo.

Capítulo 4: Fase de Elaboración

4.1 Análisis

Durante el análisis, analizamos los requisitos que se describieron en la captura de requisitos, refinándolos y estructurándolos. El objetivo de hacerlo es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que nos ayude a estructurar el sistema entero, incluyendo su arquitectura [2].

Dicho simplemente, llevando a cabo el análisis conseguimos una separación de intereses que prepara y simplifica las siguientes actividades de diseño e implementación, delimitando los temas que deben resolverse y las decisiones que se deben tomar en esas actividades.

El lenguaje que se utiliza en el análisis se basa en el modelo conceptual que se denomina "modelo del análisis". El modelo de análisis nos ofrece un mayor poder expresivo y una mayor formalización, como por ejemplo, la que nos proporcionan los diagrama de interacción que se utilizan para describir los aspectos dinámicos del sistema.

Una de las partes mas importantes del análisis es la "realización de caso de uso-análisis", la cual es una colaboración dentro del modelo de análisis que describe como se lleva a cabo y se ejecuta un caso de uso en termino de las clases y objetos en interacción.

Dentro de los artefactos que encontramos en el modelo de análisis se pueden nombrar los siguientes:

Clase de análisis: Representa una abstracción de una o varias clases y/o subsistemas del diseño del sistema. Se centra en el tratamiento de los requisitos funcionales, define atributos a alto nivel, participa en relaciones.

Clases de interfaz: Modelan interacción entre el sistema y sus actores. Modelan partes del sistema que dependen de sus actores, lo cual implica que clarifican y reúnen los requisitos del límite del sistema.

Clases de entidad: Se utilizan para modelar información que posee larga vida y que es persistente. Muestran una estructura de datos lógica y contribuyen a comprender de que información depende el sistema.

Dentro de los artefactos que encontramos en la realización de un caso de uso-análisis se pueden nombrar los siguientes:

Diagrama de clases: Muestran las clases con sus relaciones.

Diagrama de interacción: Mostrar las interacciones entre los objetos creando enlaces entre ellos.

Requisitos especiales: Descripciones textuales que recogen los aspectos no funcionales sobre una realización de casos de uso.

4.2 Análisis de Casos de Uso

A continuación se analizarán los Casos de Uso más significativos para el sistema y la lógica del negocio. Los diagramas de análisis de cada caso de uso se encuentran en el Anexo D.

4.2.1 Caso de Uso: Ingresar Ficha Alumno

- La secretaria entrevista al apoderado.
- El apoderado le entrega la información del alumno.
- La secretaria registra la información y la ingresa en la nueva ficha del alumno.

Clases que se identificaron del caso de uso:

- Secretaria.
- Apoderado.

- Alumno.
- Ficha alumno.

4.2.2 Caso de Uso: Ingresar Ficha Profesor

- La secretaria recibe la información del profesor.
- La secretaria ingresa la información a la ficha profesor.

Clases que se identificaron del caso de uso:

- Secretaria.
- Profesor.
- Ficha Profesor.

4.2.3 Caso de Uso: Generar Certificado de Alumno Regular

- El apoderado le solicita a la secretaria la generación del certificado.
- La secretaria consulta el nombre del alumno.
- La secretaria genera el certificado.

Clases que se identificaron del caso de uso:

- Apoderado.
- Secretaria.
- Certificado.
- Alumno.

4.2.4 Caso de Uso: Generar Seguro de Accidente Escolar

- El profesor jefe le solicita a la secretaria la generación del seguro.
- La secretaria consulta el nombre del alumno.
- La secretaria genera el seguro.

Clases que se identificaron del caso de uso:

- Profesor Jefe.
- Secretaria.
- Seguro.
- Alumno.

4.2.5 Caso de Uso: Ingresar/Modificar Asistencia

- El profesor pide la asistencia a los alumnos.
- El profesor verifica la presencia de los alumnos.
- El profesor registra las asistencias.

Clases que se identificaron del caso de uso:

- Profesor.
- Alumno.
- Asistencia.

4.2.6 Caso de Uso: Listar Alumnos por Curso

- El profesor ingresa el curso correspondiente.
- El curso despliega a los alumnos.

Clases que se identificaron del caso de uso:

- Profesor.
- Curso.
- Alumno.

4.2.7 Caso de Uso: Cierre de Libro de Clases

- El profesor jefe consulta el curso.
- El profesor jefe verifica las notas de los alumnos.
- El profesor jefe cierra el libro de clases.

Clases que se identificaron del caso de uso:

- Profesor jefe.
- Notas.
- Alumno.
- Libro de clases.

4.2.8 Caso de Uso: Generar Informe de Notas

- El profesor jefe consulta el curso.
- El profesor jefe obtiene las notas de un alumno.
- El profesor jefe genera el informe.

Clases que se identificaron del caso de uso:

- Profesor jefe.

- Curso.
- Nota.
- Alumno.
- Informe.

4.2.9 Caso de Uso: Cerrar Periodo

- U.T.P. revisa todos los cursos.
- Los cursos tienen todas las asignaturas cerradas.
- U.T.P. cierra el periodo.

Clases que se identificaron del caso de uso:

- U.T.P.
- Curso.
- Asignatura.

4.2.10 Caso de Uso: Generar Registros

- U.T.P. elige el tipo de registro que desea hacer.
- U.T.P. genera el registro elegido.

Clases que se identificaron del caso de uso:

- U.T.P.
- Registro.

4.2.11 Caso de Uso: Asignar Profesor a Asignatura

- U.T.P elige el profesor disponible.
- U.T.P. elige la asignatura a la cual le designará al profesor.
- U.T.P asigna el profesor a la asignatura.

Clases que se identificaron del caso de uso:

- U.T.P.
- Profesor.
- Asignatura.

4.2.12 Caso de Uso: Asignar Profesor Jefe a Curso

- U.T.P elige el profesor jefe disponible.
- U.T.P. elige el curso al cual le designará al profesor jefe.
- U.T.P asigna el profesor jefe al curso.

Clases que se identificaron del caso de uso:

- U.T.P.
- Profesor Jefe.
- Curso.

4.3 Diagrama de Clases Conceptuales

Los diagramas de clases son utilizados durante el proceso de Análisis y Diseño de los sistemas informáticos, donde se crea el diseño conceptual de la información que se manejara en el sistema, los componentes que se encargaran del funcionamiento y la

relación entre uno y otro. A continuación se muestra el diagrama de clases del sistema desarrollado.

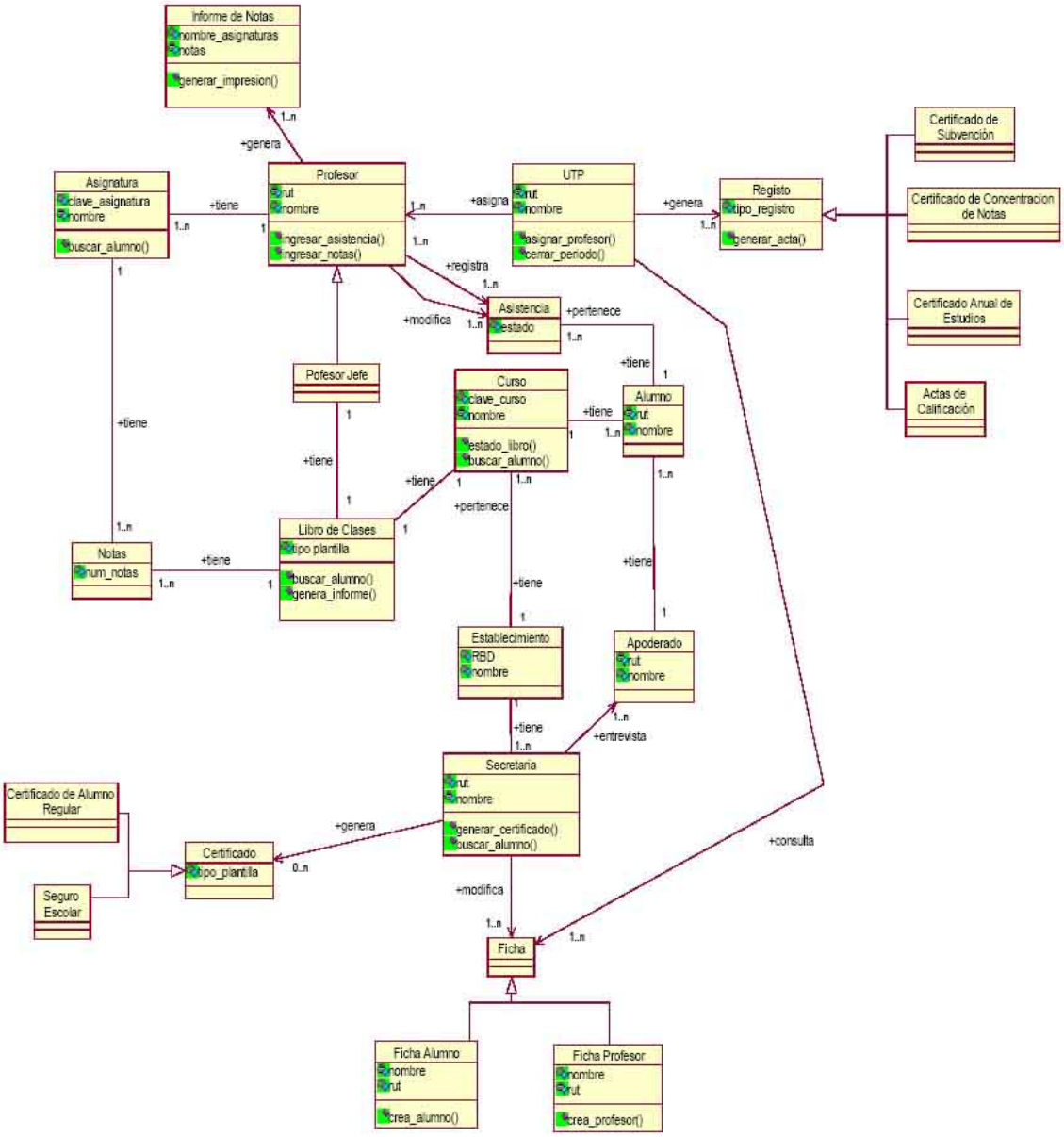

Ilustración 4-1 Diagrama de Clases Conceptual

4.4 Diseño

En el diseño modelamos el sistema y encontramos su forma para que soporte todos los requisitos, incluyendo los requisitos no funcionales y otras restricciones.

El propósito del diseño es adquirir una comprensión de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías, etc.

El diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción. El modelo del diseño esta muy cercano al de implementación, lo que es natural para guardar y mantener el modelo del diseño a través del ciclo de vida completo del software. El modelo del diseño se puede utilizar para visualizar la implementación y para soportar las técnicas de programación grafica.

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en como los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. En el modelo de diseño, los casos de uso son realizados por las clases de diseño y sus objetos. Esto se representa por colaboraciones en el modelo del diseño y denota realización de casos de uso-diseño.

Una realización de casos de uso-diseño es una colaboración en el modelo del diseño que describe como se realiza un caso de uso especifico, y como se ejecuta, en términos de clases de diseño y sus objetos.

Una clase del diseño es una abstracción sin costura de una clase o construcción similar en la implementación del sistema.

Una realización de caso de uso-diseño proporciona una traza directa a la realización de caso de uso-análisis en el modelo de análisis. Además tiene una descripción de flujos de eventos textual, diagramas de clases que muestran las clases de diseño participantes, y diagramas de interacción que muestran la realización de un flujo o escenario concreto de un caso de uso, en términos de interaccione entre objetos [2].

4.5 Diagrama de Secuencia por Caso de Uso

En breves palabras un diagrama de secuencia, muestra un conjunto de mensajes, dispuestos en una secuencia temporal. En si puede mostrar una historia individual de una transacción.

A continuación, a modo de ejemplo, se expondrá el diagrama de secuencia del caso de uso Ingresar Asistencia, el resto de los diagramas de secuencia véase en Anexo E.

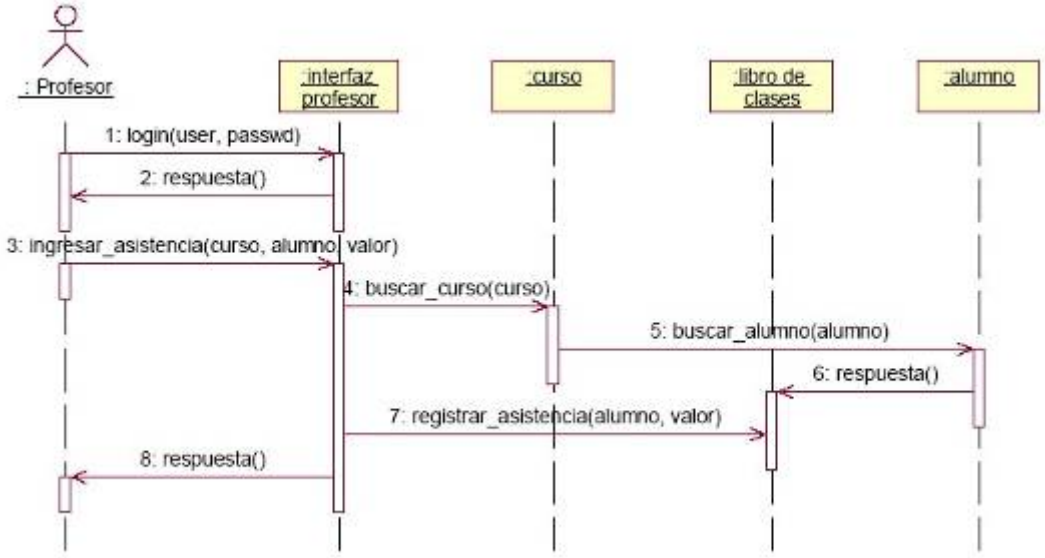

Ilustración 4-1 Diagrama de Secuencia Ingresar Asistencia

4.5 Modelo Relacional

Este modelo considera la base de datos como una colección de relaciones. De manera simple, una relación representa una tabla, en que cada fila representa una colección de valores que describen una entidad del mundo real. Cada fila se denomina tupla o registro y cada columna campo. A continuación se presenta el modelo relacional obtenido con la ayuda de la herramienta Rational Rose.

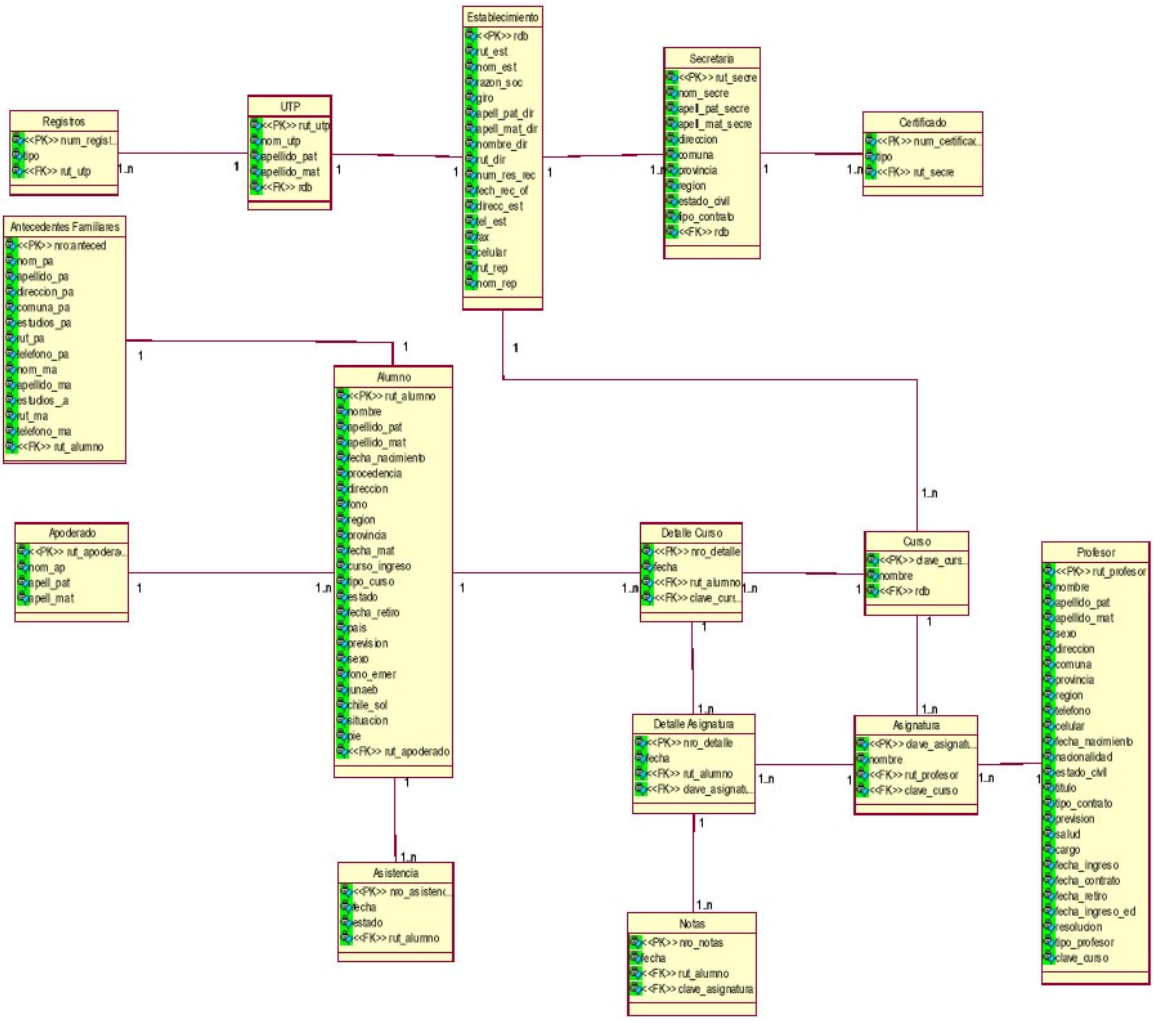

Ilustración 4-3 Modelo Relacional

Capítulo 5: Fase de Construcción

El equipo que trabaja en la fase de construcción, a partir de una línea base de la arquitectura ejecutable, y trabajando a través de una serie de iteraciones e incrementos, desarrolla un producto de software listo para su operación inicial en el entorno del usuario, a menudo llamado versión beta. A medida que se avanza se dejan cerrados los modelos de análisis, diseño e implementación.

En las primeras iteraciones de la fase de construcción, los flujos de trabajo iniciales reciben mayor énfasis; los últimos, un énfasis menor. Este énfasis se desplaza a lo largo de las iteraciones de construcción.

Al momento de construir el software, los requisitos y la arquitectura son estables. El énfasis se pone a completar las realizaciones de los casos de uso para cada uno de ellos, diseñando los subsistemas y clases necesarios, implementándolos como componentes y probándolos tanto de forma individual como en construcciones [2].

5.1 Patrones de Diseño

Se han estudiado y aplicado dos **patrones de diseño**, el patrón ***MVC*** y el patrón ***DAO*** con el objetivo de **encapsular** la información para lograr una arquitectura cercana a **3 capas** implementada en PHP.

a) **Modelo Vista Controlador (MVC)** es un patrón de diseño de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones Web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página.

- **Modelo:** Esta es la representación específica del dominio de la información sobre la cual funciona la aplicación. El modelo es otra forma de llamar a la capa de dominio. La lógica

de dominio añade significado a los datos; por ejemplo, calculando si hoy es el cumpleaños del usuario o los totales, impuestos o portes en un carrito de la compra.

• **Vista:** Este presenta el modelo en un formato adecuado para interactuar, usualmente un elemento de interfaz de usuario.

• **Controlador:** Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

• Muchas aplicaciones utilizan un mecanismo de almacenamiento persistente (como puede ser una base de datos) para almacenar los datos. MVC no menciona específicamente esta capa de acceso a datos.

Es común pensar que una aplicación tiene tres capas principales: presentación (IU), dominio, y acceso a datos. En MVC, la capa de presentación está partida en controlador y vista. La principal separación es entre presentación y dominio; la separación entre V/C es menos clara. Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo que sigue el control generalmente es el siguiente:

El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace)

El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.

El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo, produce un listado del

contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, el patrón de observador puede ser utilizado para proveer cierta dirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice la interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente. [5].

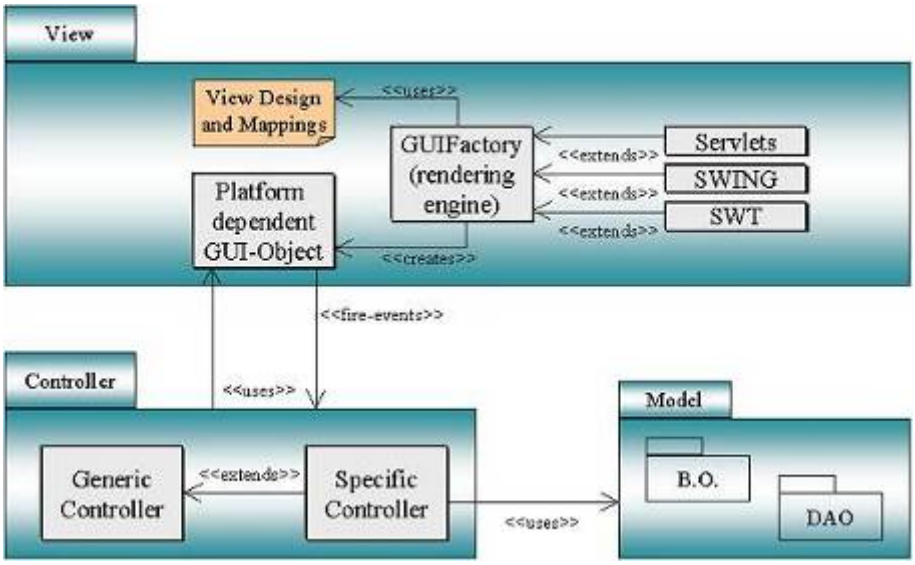

Ilustración 5-1 Patrón MVC

b) ***Data Access Object*:** El *DAO* implementa el mecanismo de acceso requerido para trabajar con la **fuente de datos**. Esta fuente de datos puede ser un almacenamiento persistente como una RDMBS, un servicio externo como un intercambio B2B, un repositorio LDAP, o un servicio de negocios al que se accede mediante CORBA Internet Inter-ORB Protocol (IIOP) o sockets de bajo nivel. Este patrón permite **adaptarse** a diferentes esquemas de almacenamiento sin que esto afecte a sus clientes o componentes de negocio. Esencialmente, el DAO actúa como un **adaptador** entre el componente y la fuente de datos [5].

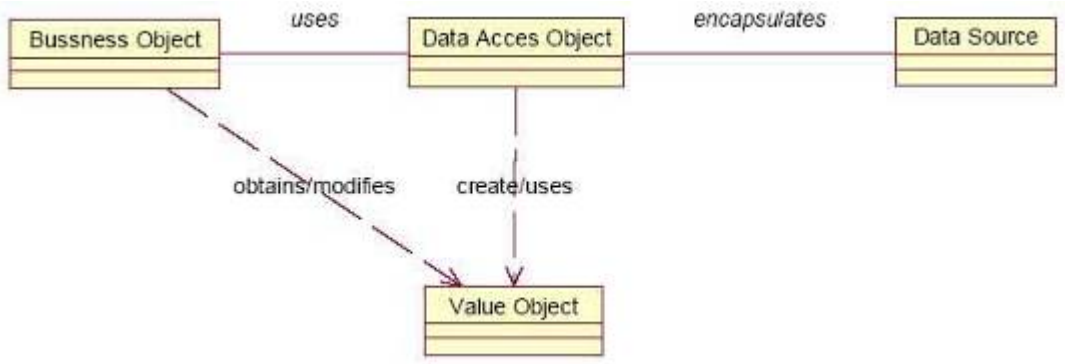

Ilustración 5-2 Patrón DAO

• **Bussness Object:** representa los datos del cliente. Es el objeto que requiere el acceso a la fuente de datos para obtener y almacenar datos.

• **Data Access Object:** es el objeto principal de este patrón. Abstrae la implementación del acceso a datos subyacente al `Business Object` para permitirle un acceso transparente a la fuente de datos.

• **Data Source:** representa la implementación de la fuente de datos. Una fuente de datos podría ser una base de datos como un RDBMS, un OODBMS, un repositorio XML, un fichero plano, etc. También lo pueden ser otros sistemas (mainframes/legales), servicios (servicio B2B u oficina de tarjetas de crédito), o algún tipo de repositorio (LDAP). [5]

5.2 Diagrama de Componentes

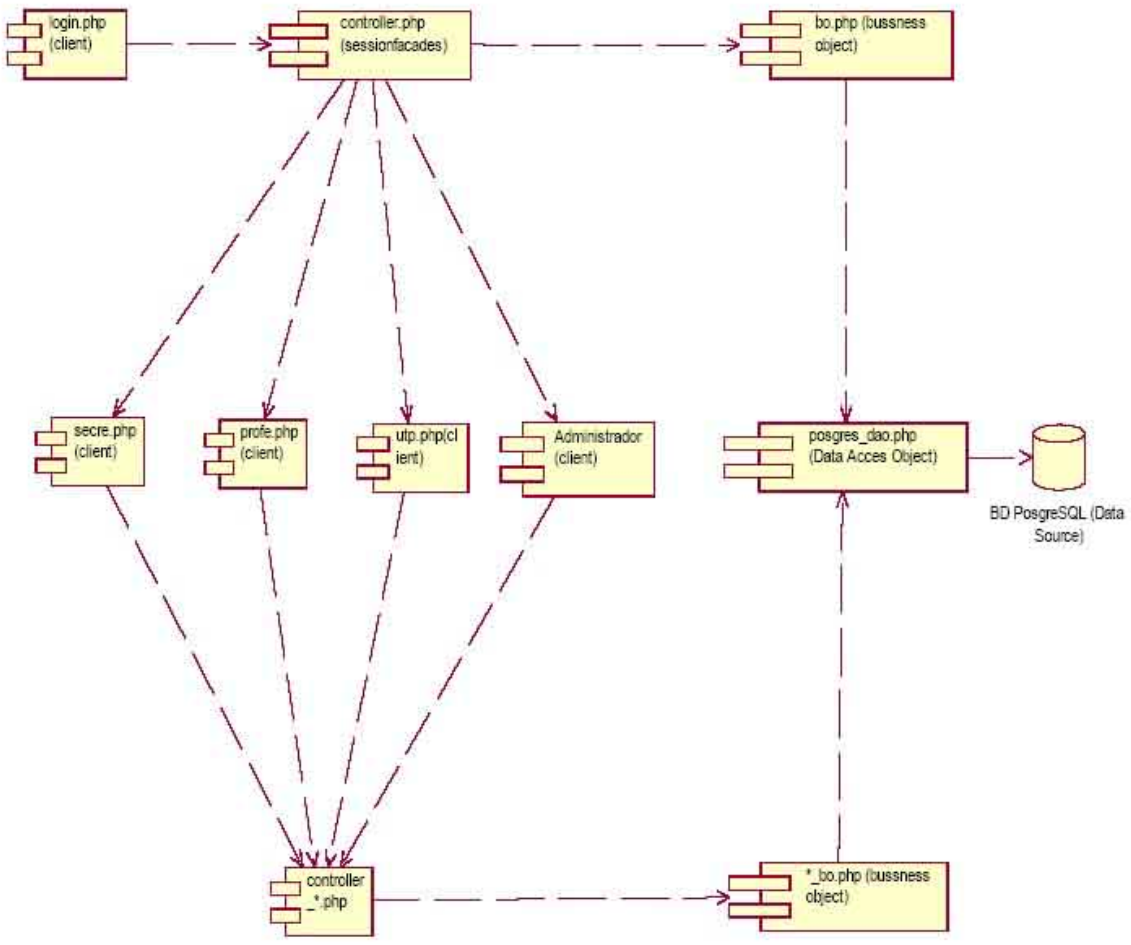

Ilustración 5-3 Diagrama de Componentes del Modelo de Implementación

5.3 Implementación

Al pasar del modelado a la implementación se ha procedido realizar los siguientes pasos lógicos:

a) Refinamiento de los modelos: los desarrolladores han revisado cada modelo generado en las etapas anteriores con el fin de encontrar errores e inconsistencias con respecto a los requerimientos expuestos por el cliente. En el modelo relacional se han incorporado nuevos campos a las tablas ya existentes por la razón de que en una de las entrevistas con el cliente expuso la necesidad de nueva información que no fue revelada en instancias anteriores.

b) Implementación del modelo físico al SABD: con el fin de seguir con la construcción del sistema, en este punto se ha dispuesto a construir las tablas que representan la abstracción de la forma de trabajo actual.

c) Definición del estilo de programación: Generalmente cada programador tiene su propio estilo para programar, por ello se generan diferentes problemas si no se sigue un criterio común a la hora de programar. Un buen estilo de programación deberá tener una estructura de código fácil de entender, no sólo para los desarrolladores, sino también para otras personas. Los desarrolladores de este proyecto han decidido utilizar los siguientes criterios de buen estilo:

- **Nombres significativos para variables, procedimientos y funciones:** Los nombres de variables y procedimientos definidos deberán ser significativos. Los nombres que se elijan deben ser autoexplicativos con respecto a su propósito, para ello se puede utilizar estándares de nombres convencionales.

- **Identación y espacios apropiados en el código:** La identación es usada para tener una mejor visibilidad en el diseño de un programa. La identación muestra las líneas que están subordinadas a otras líneas. Por ejemplo, todas las líneas que forman el cuerpo de un ciclo deberán estar identadas con la instrucción principal del ciclo. Se utilizará una tabulación por cada línea subordinada de otra.

- **Documentar el Código:** Las complicadas e inusuales secciones de código deberán ser documentadas. Idealmente cada variable y arreglo deberá tener comentarios donde se definan tal que su función pueda ser entendida después. Se utilizará el estilo de comentario del leguaje C.

- **Procedimientos coherentes:** Cada procedimiento deberá ser diseñado para una tarea simple. Si un procedimiento maneja muchas tareas, es lógico que pueda ser difícil de entender y pueda ocurrir fácilmente un error.

- **Minimizar el acoplamiento:** Pasar un parámetro es una buena práctica de programación, pero muchos parámetros puede llegar a ser muy difícil de manejar. Los procedimientos con

muchos parámetros son altamente acoplables. Hasta donde sea posible se deben minimizar el número de parámetros, sin embargo las variables globales no deberán ser usadas.

• **Minimizar el alcance de los datos:** Las variables y los arreglos pueden se accesados por el código en diferentes partes de un programa, desde cualquier lugar si son globales. Esto es lo ideal, pero si no hay cuidado algunos efectos extraños pueden ocurrir en otras partes del programa, como por ejemplo, colocando un valor en una variable por error. Restringiendo el rango de acceso, o el alcance de una variable o un arreglo se puede evitar este problema.

• **"Revisar el código 10 veces antes de compilar":** Esta técnica consiste en verificar el código en busca de errores lógicos que el compilador pasa por alto y que a simple vista se pueden detectar y corregir.

5.4 Problemas Detectados y Soluciones Propuestas

a) Poco conocimiento de los desarrolladores con HTML: el principal problema fue entender el funcionamiento de los elementos de dicho lenguaje, lo que se agravaba al tratar de vincularlo con PHP. El traspaso de la información de las variables desde un formulario hacia una pagina php no es trivial puesto que hay que tener consideraciones al momento de definir los tipos de variable con la cual van a ser insertados en la base de datos. Para soslayar este problema se estudio más a fondo dicho lenguaje.

b) Problemas con los nombres de las variables: en una primera instancia los desarrolladores tuvieron problemas con la declaración de variables, como por ejemplo existían distintos nombres de variables para referenciar el mismo elemento tanto al recibirlo del formulario a php como de php a la base de datos. La solución encontrada fue definir los criterios de programación definidos en el punto anterior.

c) Dificultad en la validación de los datos con php: en el momento de restringir los datos de entrada hacia el sistema resulto ser demasiado difícil la validación de estos con php, estimando que si se seguía la validación con este lenguaje el tiempo invertido iba a ser excesivo. La solución encontrada fue la incorporación y el estudio de un nuevo lenguaje, que no se había propuesto anteriormente, javascript; dicho lenguaje proporciona formas

relativamente sencillas de validar datos de entrada ingresados por el usuario. Una ventaja que no hay que dejar de mencionar es la existencia de gran cantidad de código de libre distribución para el uso que se estime conveniente.

d) Problema del traspaso del modelado al código: debido a la inserción de un patrón de diseño de tres capas surgieron una serie de dificultades dado a las limitaciones de php y a la poca experiencia de los desarrolladores. Uno de los problemas mas significativos fue tratar de encapsular todo lo referente con la lógica del negocio y aislarla del resto de los componentes. Estos problemas fueron resueltos con la dedicación de este patrón en cada parte de la programación.

e) Problemas con el tipo de variables en php: uno de los problemas más recurrentes fue este, debido a que una de las desventajas que posee php es que no define el tipo de sus variables al declararlas, por lo cual una misma variable puede recibir un string, un arreglo, un número, etc. la consecuencia de esta desventaja para los desarrolladores es la perdida relativa del control de lo que se esta mandando a la base de datos lo que genera inconsistencia con los tipos de variables definido en esta. La solución encontrada es prestar mucha atención en las variables utilizadas y no usar una misma variable para dos datos de distinto tipo.

5.5 Plan de Pruebas

Para lograr un correcto desarrollo de Software, es necesario establecer el plan de prueba con el fin de explicitar: el alcance, enfoque, recursos requeridos, calendario, responsables; para así poder llevar a cabo las pruebas de una manera estructurada y de acuerdo al marco de trabajo establecido. La finalidad de las pruebas es comprobar la funcionalidad del sistema y con esto afirmar que lo que se está construyendo es realmente lo propuesto en los requerimientos descritos en etapas anteriores. Una prueba exitosa es aquella que provoca que el sistema se desempeñe incorrectamente y así exponer un defecto, por lo tanto, las pruebas demuestran la presencia de fallas en el programa [4].

Para que estas pruebas cumplan con su finalidad, es necesario encontrar errores o falencias no detectadas hasta ese momento. Lo ideal es que este proceso se realice en la etapa de

desarrollo de software, ya que en un periodo posterior los costos pueden llegar a ser muy elevados.

5.5.1 Objetivos de las Pruebas:

Planificar las pruebas necesarias en cada iteración, incluyendo las pruebas de integración y las pruebas de sistema. Las pruebas de integración son necesarias para cada construcción dentro de la iteración, mientras que las pruebas de sistema son necesarias sólo al final de la iteración. Cabe señalar las pruebas de aceptación las cuales se realizan con el cliente al final de cada iteración.

Diseñar e implementar las pruebas creando los casos de prueba que especifican qué probar, creando los procedimientos de prueba que especifican cómo realizar las pruebas.

Realizar las diferentes pruebas y manejar los resultados de cada prueba sistemáticamente. Las construcciones que detectan defectos son probadas de nuevo y posiblemente devueltas a otro flujo de trabajo, como diseño o implementación, de forma que los defectos importantes puedan ser arreglados.

El resultado final de las pruebas especifica:

Casos de Prueba: especifican qué probar en el sistema

Procedimientos de Prueba: especifican cómo realizar los casos de prueba

Existen dos principales métodos para probar un producto de software, estos son:

Prueba de Caja Blanca.

Prueba de Caja Negra.

En el flujo de trabajo de la prueba verificamos el resultado de la implementación probando cada construcción, incluyendo tanto construcciones internas como intermedias, así como las versiones finales del sistema a ser entregadas a terceros.

5.5.2 Prueba de Caja Blanca.

Este método de casos de prueba se basa en un estudio exhaustivo de los detalles procedimentales, con esto se pretende probar todos los caminos lógicos de cada modulo abarcando bucles y decisiones lógicas entre otras. En la realidad, esto es difícil de probar, ya que por muy pequeño que sea el programa, éste consta de un gran número de caminos lógicos.

Existen distintas técnicas de prueba de caja blanca que se detallan a continuación:

● Prueba de estructura de control: Son variantes que amplían la cobertura de las pruebas y mejoran la calidad de la prueba de caja blanca.

- Prueba de condiciones: Se definen casos de prueba que ejecuten las condiciones lógicas contenidas en un programa. Existen condiciones simples (una variable lógica o una expresión relacional) y condiciones compuestas (formada por dos o más condiciones simples, operadores lógicos y paréntesis). Si una condición es incorrecta, entonces al menos uno de los componentes está incorrecto, por lo que podemos encontrar errores en operador lógico, paréntesis lógico, operador relacional y expresión aritmética.

Prueba de flujo de datos: Selecciona caminos de pruebas de un programa de acuerdo con la ubicación de definiciones y los usos de las variables del programa. Es útil para seleccionar caminos de prueba de un programa que contengan sentencias condicionales o de bucles anidadas. Su efectividad se encuentra en la protección contra errores. Sin embargo la selección de caminos de prueba es más difícil.

- Prueba de bucles: Se basa en la realización de pruebas a los bucles de un programa. Se pueden definir 4 clases diferentes de bucles:
  * Bucles simples
  * Bucles anidados
  * Bucles concatenados
  * Bucles no estructurados

- Prueba de camino básico: Esta técnica define un conjunto básico de caminos de ejecución, en donde los casos de pruebas garantizan la ejecución de por lo menos una vez, cada sentencia del programa. Utiliza una notación de grafo de flujo para representar el flujo de control, en donde los nodos del grafo representan una o más sentencias y las aristas representan el flujo de control.

5.5.3 Prueba de Caja Negra.

Este método de prueba intenta encontrar errores en funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos externa, errores de rendimiento, errores de inicialización y de terminación.

Se centran en los requisitos funcionales del software, para lo cual se utiliza un conjunto de condiciones de entrada.

En este método se tienen las siguientes técnicas:

- Partición equivalente: Divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de pruebas. Consta de una clase de equivalencia, la cual representa un conjunto de estados válidos o no válidos para condiciones de entrada.

Rango, una clase de equivalencia válida y dos inválidas.
Valor, una clase de equivalencia válida y dos inválidas.
Conjunto, una clase de equivalencia válida y una inválida.
Lógica, una clase de equivalencia válida y una inválida.

- Análisis de valores limites: Complementa la técnica de partición equivalente y la selección de los casos de pruebas se lleva a cabo en los extremos de las clases.

Si una condición de entrada especifica un rango delimitado por los valores a y b, se diseñan casos de prueba con valores justo por encima y debajo de a y b.
Si una condición de entrada especifica un número de valores, se diseñan casos de pruebas para los valores máximos y mínimos.

Los puntos anteriores se aplican a las condiciones de salida.

● Técnica de grafo causa-efecto: El intento de traducir un determinado procedimiento en un lenguaje natural a un algoritmo basado en software conduce a errores, para ello esta técnica proporciona una representación de las condiciones lógicas y sus correspondientes acciones. Se listan las causas (condiciones de entrada) para un componente y los efectos (acciones), asignando un identificador a cada uno de ellos.

● Pruebas de comparación: Consiste en verificar los resultados de aplicaciones que se construyen en forma paralela y por separado para que tengan concordancia.

5.5.4 Selección de los Criterios para el Plan de Pruebas

Después de haber descrito cada uno de estos métodos de prueba, ya sea de caja blanca o caja negra, se está en condición de seleccionar la estrategia, el enfoque y la técnica que se van a utilizar para la realización de las pruebas del software.

Se llevará acabo una estrategia de tipo **bottom up** ajustada a las pruebas orientadas a objetos. La estrategia es comenzar con las pruebas unitarias a cada modulo, esto es centrar el proceso de verificación en la menor unidad del diseño de software, luego se progresa hacia las pruebas de integración, estas consisten en construir la estructura del programa mientras que al mismo tiempo se llevan a cabo pruebas para detectar errores asociados con la interacción. El objetivo es tomar los módulos probados mediante las pruebas unitarias y construir una estructura de programa que este de acuerdo con lo que dicta el diseño para luego finalizar con las pruebas de validación.

Se ha elegido el enfoque de **caja negra**, ya que éste se basa en los requerimientos funcionales del sistema, y se considera que para el sistema a desarrollar las pruebas principales que se deben llevar a cabo para detectar los posibles errores que puede tener el programa, son las pruebas de lo resultados que arroja el sistema dada cierta información.

Para esto se ha decidido utilizar la técnica de **causa-efecto**, que refleja que con información entregada al sistema, éste entrega los resultados esperados. Para ello, se debe probar con datos válidos, inválidos o incorrectos, y así verificar que se está procesando la información

correctamente. Para complementar el trabajo de esta técnica también se ha elegido la utilización de **particiones de equivalencia** que serán utilizadas con mayor énfasis en módulos que se distingue una clara división de clases de equivalencia, como por ejemplo al ingresar las notas al libro de clases.

5.5.5 Casos de Prueba

Tabla 5-1 Caso de Prueba 1

Datos de Prueba	Resultado Esperado	Resultado Obtenido
Login: ‘’ Password: ‘’	Mensaje de alerta “Debe ingresar un Login y una Password”.	Mensaje de alerta “Debe ingresar un Login y una Password”.

Tabla 5-2 Caso de Prueba 2

Datos de Prueba	Resultado Esperado	Resultado Obtenido
Login: ‘administrador’ Password: ‘’	Mensaje de alerta “Debe ingresar una Password”.	Mensaje de alerta “Debe ingresar una Password”.

Tabla 5-3 Caso de Prueba 3

Datos de Prueba	Resultado Esperado	Resultado Obtenido
Login: ‘administrador’ Password: ‘724635’	Mensaje de alerta “Usuario inexistente”, se redirecciona a la pagina de autentificación.	Se redirecciona a la pagina de autentificación.

Tabla 5-4 Caso de Prueba 4

Datos de Prueba	Resultado Esperado	Resultado Obtenido
Login: 'administrador' Password: '123456'	Iniciar la sesión del Administrador, accediendo a la página principal de éste.	Se inicia la sesión Administrador, se accede a la página principal de éste.

Tabla 5-5 Caso de Prueba 5

Datos de Prueba	Resultado Esperado	Resultado Obtenido
Rut: '15099460-8' Nombre usuario: 'Karen' Apellido Paterno: 'Perez' Apellido Materno: 'Cotapo' Ingrese Login: 'percot' Repita Login: 'percot' Ingrese Password: '6789' Repita Password: '6789' Tipo usuario: 'Secretaria'	Mensaje de alerta "El usuario fue ingresado exitosamente". Se redirecciona a la pagina principal del administrador .	Mensaje de alerta "El usuario fue ingresado exitosamente". Se redirecciona a la pagina principal del administrador.

Tabla 5-6 Caso de Prueba 6

Datos de Prueba	Resultado Esperado	Resultado Obtenido
Rut: '14.326.542-5'	Mensaje de alerta "El rut es incorrecto" vuelve al formulario.	Mensaje de alerta "El rut es incorrecto" vuelve al formulario.
Rut: 'holamundo'	Mensaje de alerta "El valor ingresado no corresponde a un rut valido".	Mensaje de alerta "El valor ingresado no corresponde a un rut valido".

Tabla 5-7 Caso de Prueba 7

Datos de Prueba	Resultado Esperado	Resultado Obtenido
Rut: '15.099.460-8' Nombre usuario: 'Karen' Apellido Paterno: 'Perez' Apellido Materno: 'Cotapo' Ingrese Login: '' Repita Login: '' Ingrese Password: '' Repita Password: '' Tipo usuario: ''	Mensaje de alerta "Los campos con asterisco deben ser llenados", vuelve al formulario	Mensaje de alerta "Los campos con asterisco deben ser llenados", vuelve al formulario

Tabla 5-8 Caso de Prueba 8

Datos de Prueba	Resultado Esperado	Resultado Obtenido
Rut: '15.099.460-8' Nombre usuario: 'Karen' Apellido Paterno: 'Perez' Apellido Materno: 'Cotapo' Ingrese Login: 'karen' Repita Login: 'kasren' Ingrese Password: '6789' Repita Password: '6789' Tipo usuario: 'secretaria'	Mensaje de alerta "Los login no coinciden", vuelve al formulario.	Mensaje de alerta "Los login no coinciden", vuelve al formulario.
Rut: '15.099.460-8' Nombre usuario: 'Karen' Apellido Paterno:	Mensaje de alerta "Las password no coinciden", vuelve al formulario.	Mensaje de alerta "Las password no coinciden", vuelve al formulario.

'Perez' Apellido Materno: 'Cotapo' Ingrese Login: 'karen' Repita Login: 'karen' Ingrese Password: '6789' Repita Password: '66789' Tipo usuario: 'secretaria'		

Tabla 5-9 Caso de Prueba 9

Datos de Prueba	Resultado Esperado	Resultado Obtenido
Rut: '15.099.460-8' Nombre usuario: 'Karen' Apellido Paterno: 'Perez' Apellido Materno: 'Cotapo' Ingrese Login: 'karen'	Mensaje de alerta "El usuario ya existe", se redirecciona a la pagina principal de administrador.	Mensaje de alerta "El usuario ya existe", se redirecciona a la pagina principal de administrador.

Repita Login: ‘karen’ Ingrese Password: ‘6789’ Repita Password: ‘6789’ Tipo usuario: ‘secretaria’		

Tabla 5-10 Caso de Prueba 10

Datos de Prueba	Resultado Esperado	Resultado Obtenido
Rut: ‘152222222’	Mensaje de alerta “El rut es incorrecto” vuelve al formulario.	Mensaje de alerta “El rut es incorrecto” vuelve al formulario.
Rut: ‘holatierra’	Mensaje de alerta “El valor ingresado no corresponde a un rut valido” vuelve al formulario.	Mensaje de alerta “El valor ingresado no corresponde a un rut valido” vuelve al formulario.

Tabla 5-11 Caso de Prueba 11

Datos de Prueba	Resultado Esperado	Resultado Obtenido
Rut: ‘56185232’	Mensaje de alerta “Esta a punto de eliminar a un usuario… ¿desea continuar?” (usuario confirma) Mensaje de alerta “El usuario	Mensaje de alerta “Esta a punto de eliminar a un usuario… ¿desea continuar?” (usuario confirma) Mensaje de alerta “No fue

	no se puede eliminar ya que no existe", vuelve al formulario.	posible eliminar al usuario", vuelve al menú principal de administrador.
Rut: 'holavenus'	Mensaje de alerta "El valor ingresado no corresponde a un rut valido".	Mensaje de alerta "El valor ingresado no corresponde a un rut valido".

Tabla 5-12 Caso de Prueba 12

Datos de Prueba	Resultado Esperado	Resultado Obtenido
Rut: '15099460-8'	Mensaje de alerta "Esta a punto de eliminar a un usuario… ¿desea continuar?" (usuario confirma) Mensaje de alerta "El usuario fue eliminado con éxito", vuelve al menú principal de administrador.	Mensaje de alerta "Esta a punto de eliminar a un usuario… ¿desea continuar?" (usuario confirma) Mensaje de alerta "El usuario fue eliminado con éxito", vuelve al menú principal de administrador.

Tabla 5-13 Caso de Prueba 13

Datos de Prueba	Resultado Esperado	Resultado Obtenido
Login: '' Password: ''	Mensaje de alerta "Debe ingresar un Login y una Password".	Mensaje de alerta "Debe ingresar un Login y una Password".

Tabla 5-14 Caso de Prueba 14

Datos de Prueba	Resultado Esperado	Resultado Obtenido
Login: ‘secretaria’ Password: ‘’	Mensaje de alerta “Debe ingresar una Password”.	Mensaje de alerta “Debe ingresar una Password”.

Tabla 5-15 Caso de Prueba 15

Datos de Prueba	Resultado Esperado	Resultado Obtenido
Login: ‘secretaria’ Password: ‘2233’	Mensaje de alerta “Usuario inexistente”, se redirecciona a la pagina de autentificación.	Se redirecciona a la pagina de autentificación.

Tabla 5-16 Caso de Prueba 16

Datos de Prueba	Resultado Esperado	Resultado Obtenido
Login: ‘secretaria’ Password: ‘6789’	Se inicia la sesión de la secretaria, accediendo a la página principal de esta.	Se inicia la sesión de la secretaria, se accede a la página principal de esta.

Tabla 5-17 Caso de Prueba 17

Datos de Prueba	Resultado Esperado	Resultado Obtenido
Rut: ‘3.952.461-9’ Nombres: ‘Marta Andrea’ Nº Mat: ‘1’ Apell Pat: ‘Vega’	Mensaje de alerta “El alumno ha sido ingresado exitosamente” vuelve al menú principal de Secretaria.	Redirecciona a la pagina principal de la secretaria.

Apell Mat: ‘Jara’ Fecha Nac: ‘12/09/1990’ Procedencia: ‘American College’ Provincia: ‘Valparaíso’ Comuna: ‘Villa Alemana’ Direccion: ‘AV. Tercera 0532’ Fono: ‘724635’ Fecha Mat: ‘27/02/1996’ Rut Apod: ‘83.579.200-5’ Tipo Curso: ‘Basica’ Curso Ingreso: ‘Primero’ Estado: ‘Activo’ Fecha Retiro: ‘’ Prevision: ‘Fonasa’ Fono Emerg: ‘531016’ Sexo: ‘Masculino’ Chile solidario: ‘on’ Indígena: ‘on’		

Junaeb: ‘on’ Situación: ‘on’ PIE: ‘on’		

Tabla 5-18 Caso de Prueba 18

Datos de Prueba	Resultado Esperado	Resultado Obtenido
Rut: ‘3.952.461-9’ Nombres: ‘Marta Andrea’ Nº Mat: ‘’ Apell Pat: ‘’ Apell Mat: ‘’ Fecha Nac: ‘12/09/1990’ Procedencia: ‘American College’ Provincia: ‘Valparaíso’ Comuna: ‘Villa Alemana’ Direccion: ‘AV. Tercera 0532’ Fono: ‘724635’ Fecha Mat: ‘27/02/1996’	Mensaje de alerta “Los campos con asterisco deben ser llenados”, vuelve al formulario	Mensaje de alerta “Los campos con asterisco deben ser llenados”, vuelve al formulario

Rut Apod: ‘83.579.200-5’ Tipo Curso: ‘Basica’ Curso Ingreso: ‘Primero’ Estado: ‘Activo’ Fecha Retiro: ‘’ Prevision: ‘Fonasa’ Fono Emerg: ‘531016’ Sexo: ‘Masculino’ Chile solidario: ‘on’ Indígena: ‘on’ Junaeb: ‘on’ Situación: ‘on’ PIE: ‘on’		

Capítulo 6: Conclusiones

En un proyecto informático uno de los factores decisivos que influyen en el éxito de las aplicaciones es el estudio de las necesidades, requerimientos y definición de claros objetivos. Esto se consigue con el apoyo de una correcta metodología analítica y de diseño. Actualmente UML es una metodología estándar que permite cubrir estos objetivos.

Según el ciclo de desarrollo de un proyecto el análisis y diseño es el proceso previo a la codificación de un sistema. En el análisis se determinan los requerimientos de lo que el sistema debe hacer; en el diseño se bosquejan los detalles de la solución. Para esto, se recopilan y analizan los elementos para entender el problema y representarlo de alguna manera, plasmando las ideas, objetivos, metas y actores que intervienen en el proyecto. El diseño es un espejo del análisis que ya se aterriza en la arquitectura, plataforma y componentes, es en el análisis y diseño de sistemas que es utilizada la metodología de lenguaje de modelado unificado (UML), técnica de recopilación de información y de especificación de requerimientos que, cómo ya se ha mencionado antes, esta en gran auge. Los distintos conceptos y métodos de análisis y diseño asociados a la especificación UML están relacionadas con las fases del Proceso Unificado de Desarrollo de Software: iterativo, incremental y centrado en la arquitectura.

En una organización o Empresa, el análisis y diseño de sistemas, es el proceso de estudiar su situación con la finalidad de observar como trabaja, evaluar y decidir si es necesario realizar una mejora, es en este contexto en donde nuestro proyecto se adapta a un modelo de trabajo de la organización para la cual se agregará como un componente que mejorará el desempeño de éste. A partir de este trabajo hemos rescatado, y luego de los sucesos imprevistos al trabajar en un proyecto académico implantado en una organización, lo complicado que se puede tornar el desarrollo del proyecto, por todos los agentes externos que influyen, es decir, trabajar bajo estándares u otros factores que no se veían considerados en un proyecto académico, trabajar diariamente en forma ardua, y en fin evaluar todo cambio posible en el sistema, nos demostró que al llevar a cabo un proyecto de

este tamaño es una tarea muy difícil de completar para dos personas. Sin embargo al tener bien definidos los requerimientos funcionales, técnicos y de metodologías de trabajo, se organiza así la forma de trabajo. Esto es una experiencia que enriquece a la hora de pensar en el futuro laboral.

Específicamente en la etapa del análisis, se logra traducir las necesidades del cliente en un modelo de sistema que utiliza uno más de los componentes: software, hardware, personas, base de datos, documentación y procedimientos.

Dentro del análisis y desarrollo de sistemas se puede determinar lo que esta funcionando bien, y lo que hay que cambiar en una empresa, para así alcanzar los objetivos y las metas de la misma. Para alcanzar dichas metas se debe realizar una planificación, la cual requiere de un estudio completo de cada una de las áreas que integran la empresa.

Una vez concluido el estudio de factibilidad, es necesario la definición de la metodología, si bien las metodologías estructuradas para el desarrollo de sistemas de información han marcado la pauta para la construcción de software, la metodología orientada a objetos es nueva para esta actividad que ha demostrado ser de gran utilidad para la solución de problemas clásicos que se presentan al elaborar software, brindando, entre otros beneficios: software reutilizable, costos y tiempos de desarrollo menores, sistemas de alta calidad, fáciles de modificar y mejorar.

Específicamente en el desarrollo del Sistema Registro Académico para los Establecimientos Educacionales de DAEM Concón, es rescatable el desarrollo bajo UML, y más aún aplicado al desarrollo Web de un proyecto, no hay que confundir la implementación de un Web Site con el desarrollo de una aplicación Web, un "Web Site" es relativamente estático, en cambio, la aplicación Web es mucho más dinámica, dispone de una lógica de negocio que puede reaccionar y alterar su estado a partir de la interacción con un usuario. El uso de UML para desarrollo de aplicaciones Web se justifica en la habilidad de representar la ejecución de la lógica de negocios del sistema. También hay que tener en cuenta que la tarea de los desarrolladores fue muy difícil ya que el manejo de la información de los establecimientos educacionales es muy compleja, por ello ellos debieron estar siempre en contacto con el cliente.

Este proyecto, al ser un prototipo de sistema, deja las puertas abiertas para seguir en plan de mejora, puesto a que se acotó el contenido debido a su complejidad, por lo tanto, este sistema puede ser implementado tanto en los establecimientos educacionales de DAEM como en otros establecimientos educacionales públicos, ya que la forma de trabajo es muy similar.

Capítulo 7: Referencias

Pressman P.: "*Ingeniería de Software un Enfoque Práctico*" Editorial McGraw Hill, 5ª Edición, 2002.

Rumbaugh J., Jacobson I., Booch G.: "*El Proceso Unificado de Desarrollo del Software"* Editorial Pearson Educación S.A. Madrid, 2000.

Rumbaugh J., Jacobson I., Booch G.: "*El Lenguaje de Modelado Unificado"* Editorial Addison Wesley Iberoamericana, Madrid, 1999.

Somerville I.: "*Ingeniería de Software"* Editorial Pearson Educación S.A., 6ª Edición, 2002.

Sun Microsystems: "*Patrones de diseño: capa del negocio y de aplicación*" www.sun.com

Wikipedia.org: "*Modelo Relacional y Modelo de Clases"* www.wikipedia.es

Siglas y Abreviaciones

ACID: Atomicity, Consistency, Isolation and Durability

ADO: ActiveX Data Objects

ANSI: American National Standards Institute

AOO: Análisis Orientado a Objetos

BASIC: Beginner´s All-purpose Symbolic Instruction Code

CASE: Computer Aided Software Engineering

CGI: Common Gateway Interface

CRC: Corrección de Redundancia Cíclica

DAEM: Departamento de Administración de Educación Municipal

DAO: Database Access Object

DBA: Database Administrator

DMO: Distributed Management Objects

GUI: Graphic User Interface

HTML: Hypertext Markup Language

IDE: Integrated Development Environment

IIOP: Inter-ORB Protocol

JDBC: Java Database Connectivity

JDK: Java Development Kit

JVM: Java Virtual Machine

L4G: Lenguajes de Programación

LDAP: Lightweight Directory Access Protocol

MINEDUC: Ministerio de Educación

MVC: Modelo Vista Controlador

ODBC: Open DataBase Connectivity

ODS: Open Data Services

OO: Orientación a Objetos

OODBMS: Objected Oriented Database Management System

PC: Personal Computer

PHP: Hyper Text Preprocessor

RAM: Random Access Memory

RDBMS: Relational Database Management System

RECH: Registro de Estudiantes de Chile

RUP: Rational Unified Process

SABD: Sistema Administrador de Base de Datos

SQL: Structured Query Language

T4G: Técnicas de Cuarta Generación

TCP: Transmission Control Protocol

UML: Unified Modeling Languages

WYSIWYG: What you see is what you get

XML: Extensible Markup Language

Anexo A: Diagramas del Modelo del Negocio

A1 Diagrama de Casos de Uso del Negocio

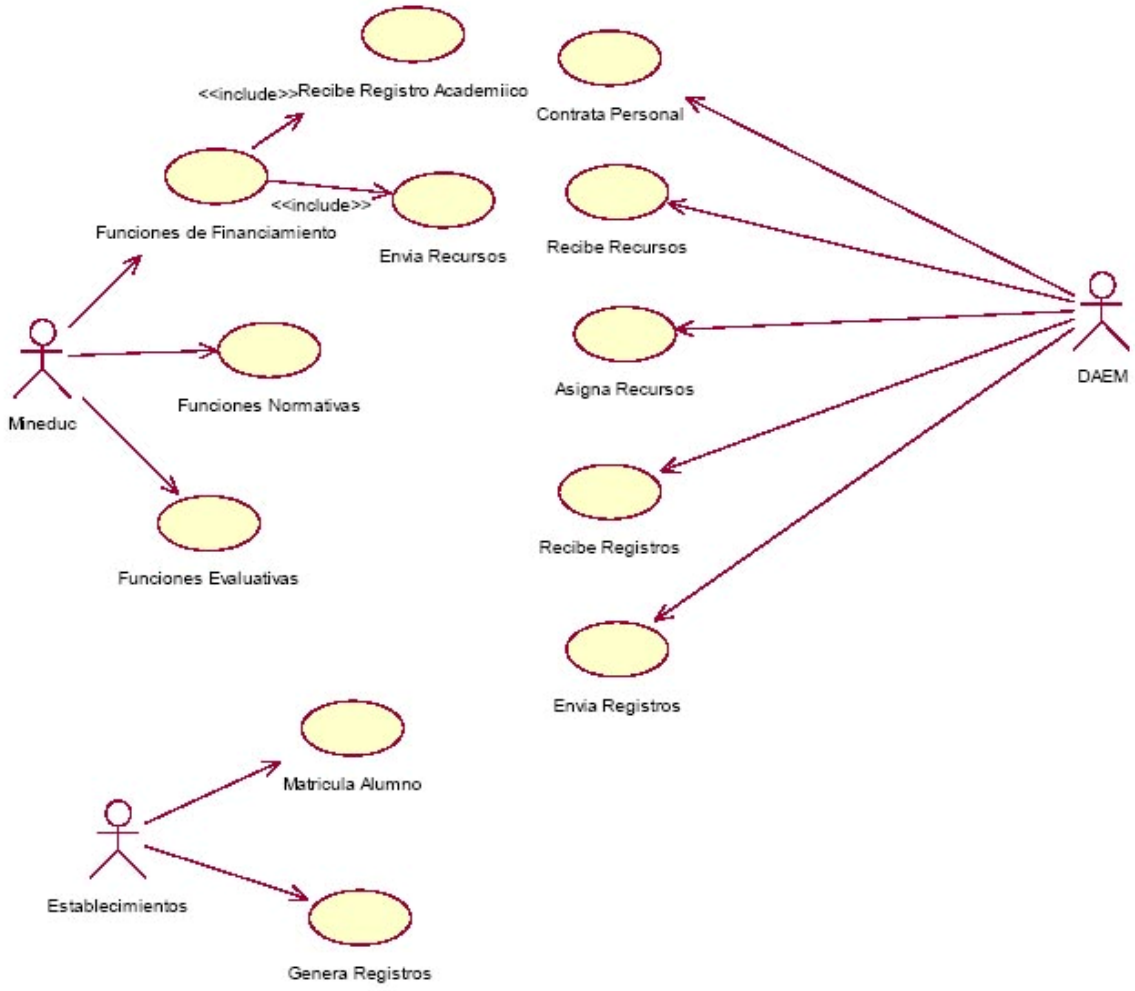

Ilustración A-1 Diagrama de Casos de Uso del Negocio

A2 Diagrama de Clases del Negocio

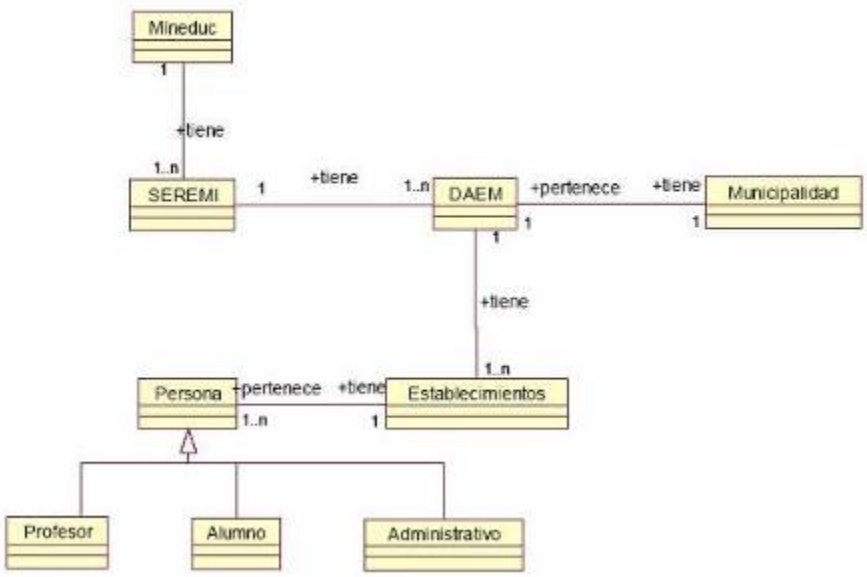

Ilustración A-2 Diagrama de Clases del Negocio

Anexo B: Diagramas del Modelo del Domino

B1 Diagrama de Clases del Dominio

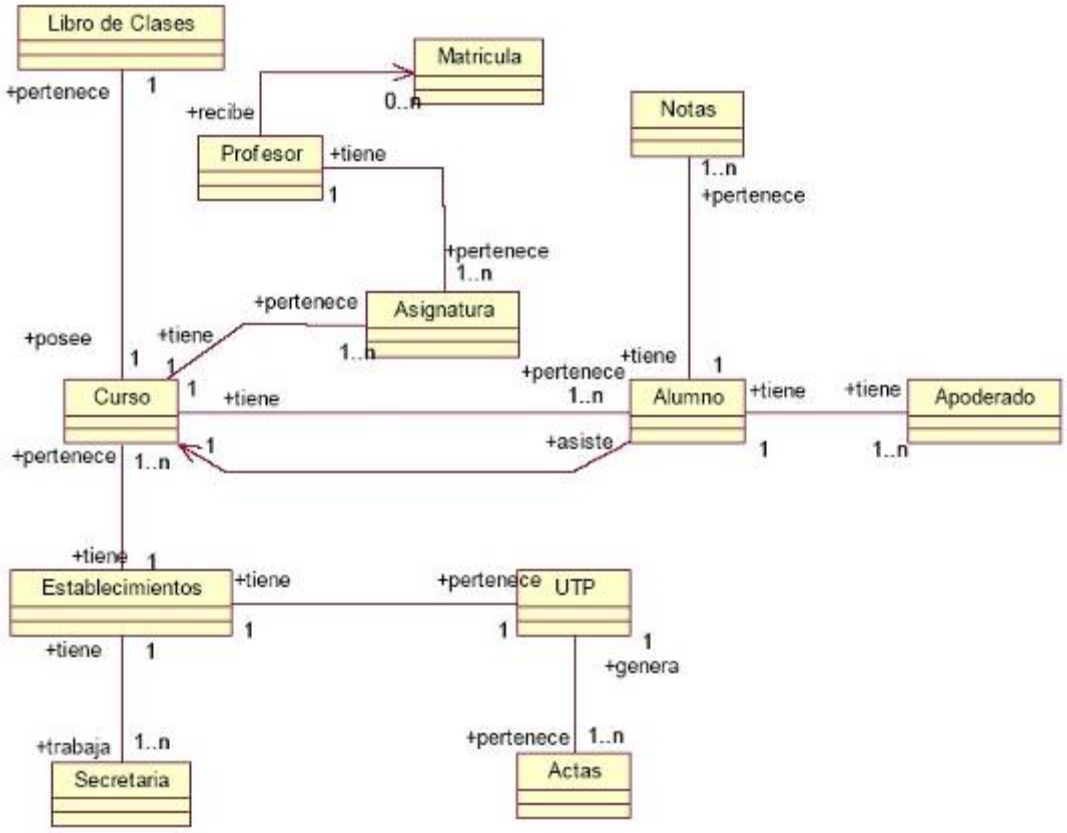

Ilustración B-1 Diagrama de Clases del Dominio

Anexo C: Especificación de Casos de Uso

Anexo D: Diagramas de Análisis de los Casos de Uso

D1 Asignar Profesor a Asignatura

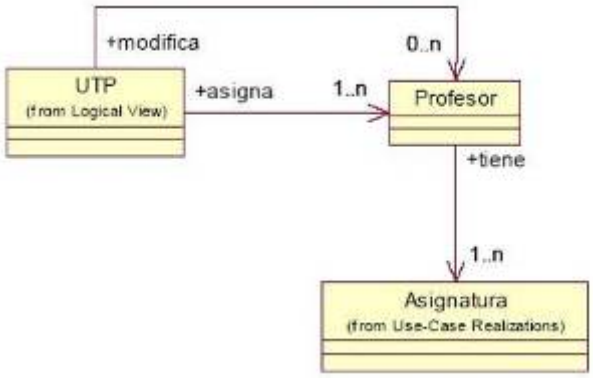

Ilustración D-1 Diagrama de Clases Asignar Profesor a Asignatura

D2 Asignar Profesor Jefe a Curso

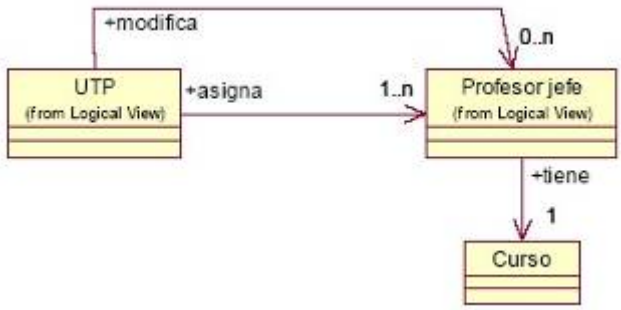

Ilustración D-2 Diagrama de Clases Asignar Profesor Jefe a Curso

D3 Cerrar Periodo

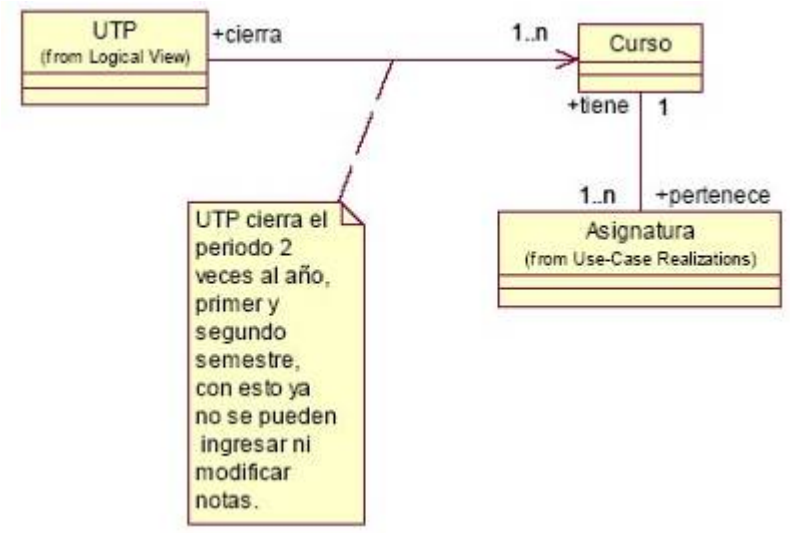

Ilustración D-3 Diagrama de Clases Cerrar Periodo

D4 Cerrar Libro de Clases

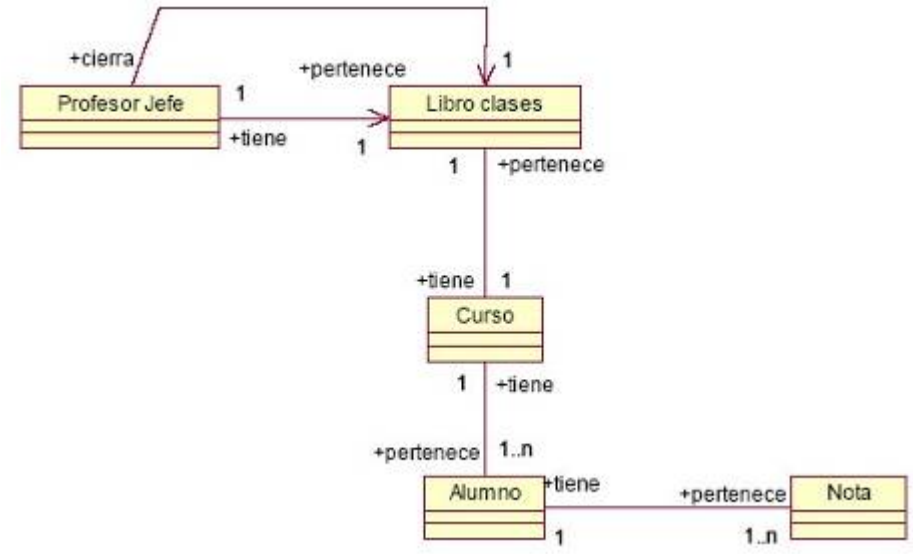

Ilustración D-4 Diagrama de Clases Cerrar Libro de Clases

D5 Generar Certificados

Ilustración D-5 Diagrama de Clases Generar Certificados

D6 Generar Registros

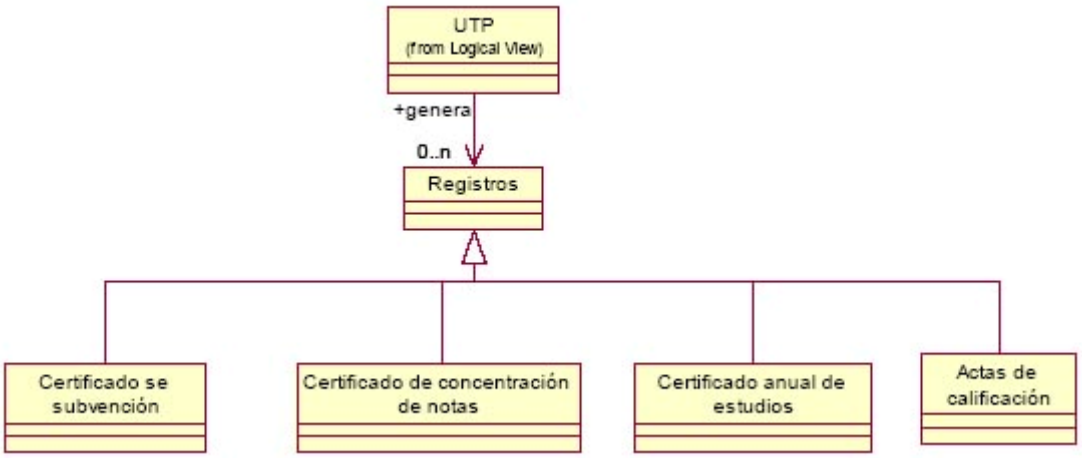

Ilustración D-6 Diagrama de Clases Generar Registros

D7 Ingresar/Modificar Asistencia

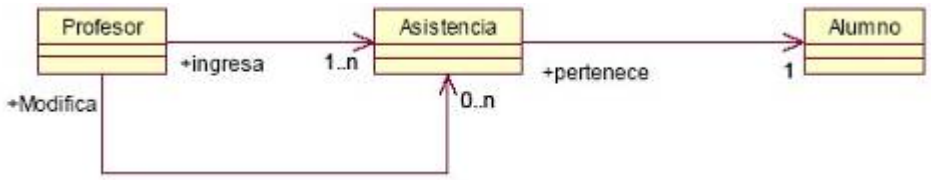

Ilustración D-7 Diagrama de Clases Ingresar/Modificar Asistencia

D8 Ingresar Ficha Alumno/Profesor

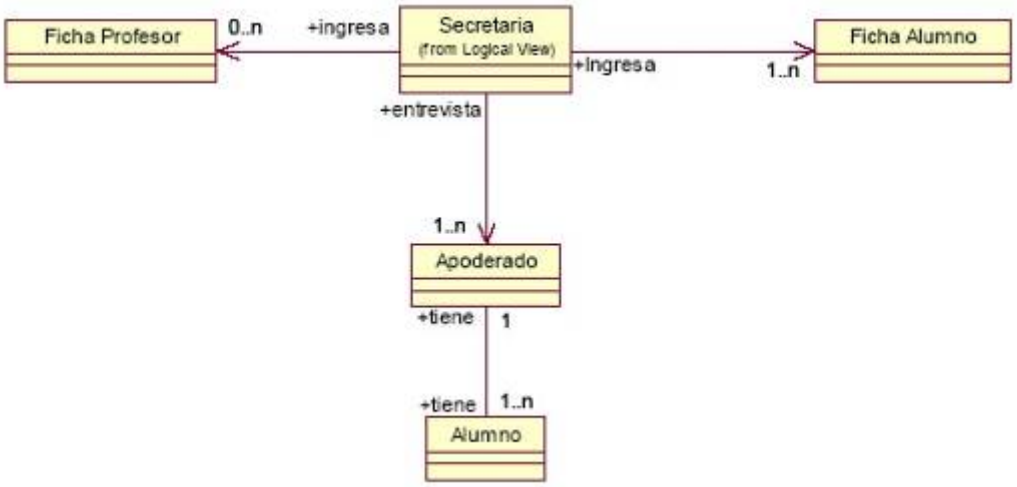

Ilustración D-8 Diagrama de Clases Ingresar Ficha Alumno/Profesor

D9 Ingresar/Modificar Notas

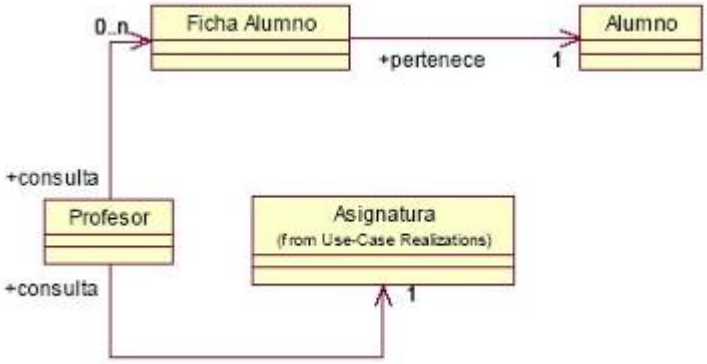

Ilustración D-9 Diagrama de Clases Ingresar/Modificar Notas

D10 Generar Certificados

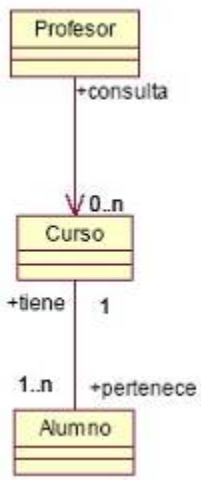

Ilustración D-10 Diagrama de Clases Listar Alumnos por Curso

D11 Listar Asistencia

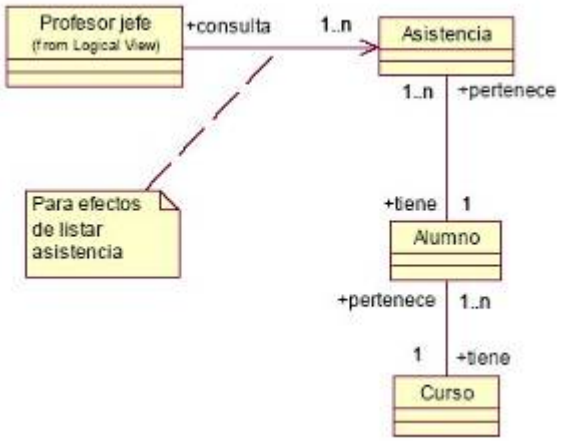

Ilustración D-11 Diagrama de Clases Listar Asistencia

D12 Modificar Ficha

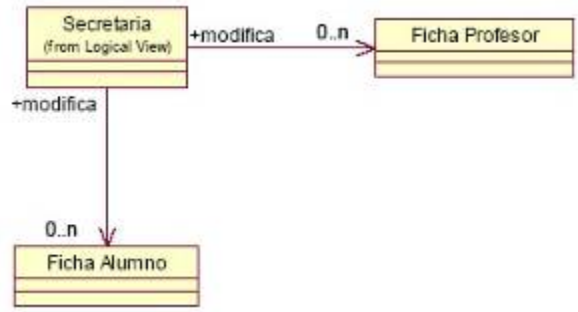

Ilustración D-12 Diagrama de Clases Modificar Ficha

D13 Mostrar Datos Profesor

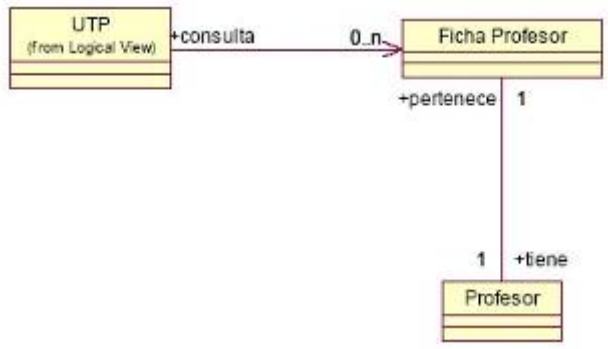

Ilustración D-13 Diagrama de Clases Mostrar Datos de Profesor

D14 Generar Informe de Notas

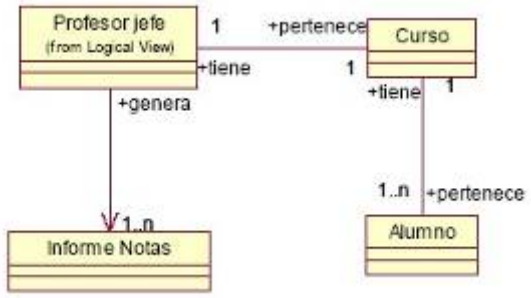

Ilustración D-14 Diagrama de Clases Generar Informe de Notas

D15 Listar Alumnos

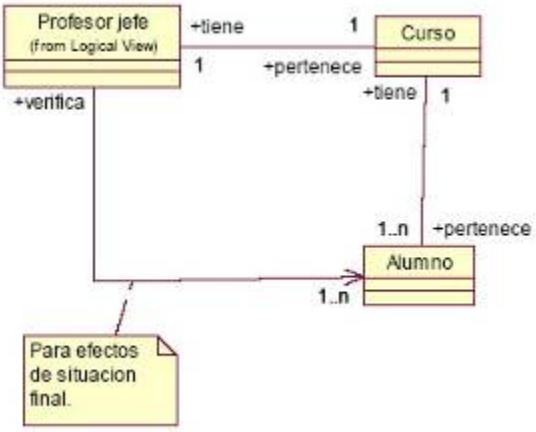

Ilustración D-15 Diagrama de Clases Listar Alumnos

Anexo E: Diagrama de Secuencia

E1 Ingresar Notas a Asignatura

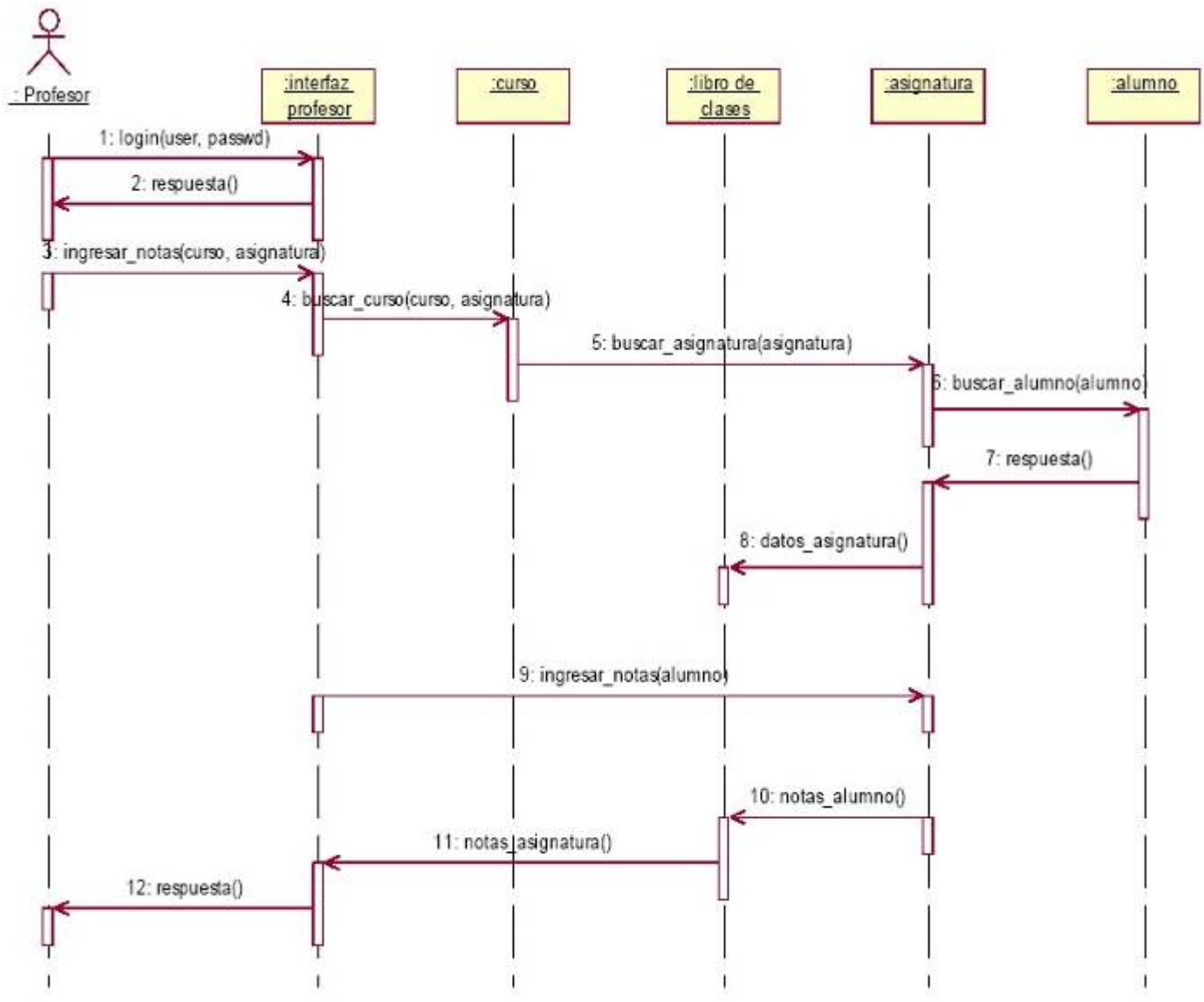

Ilustración E-1 Diagrama de Secuencia Ingresar Notas a Asignatura

E2 Listar Alumnos por Curso

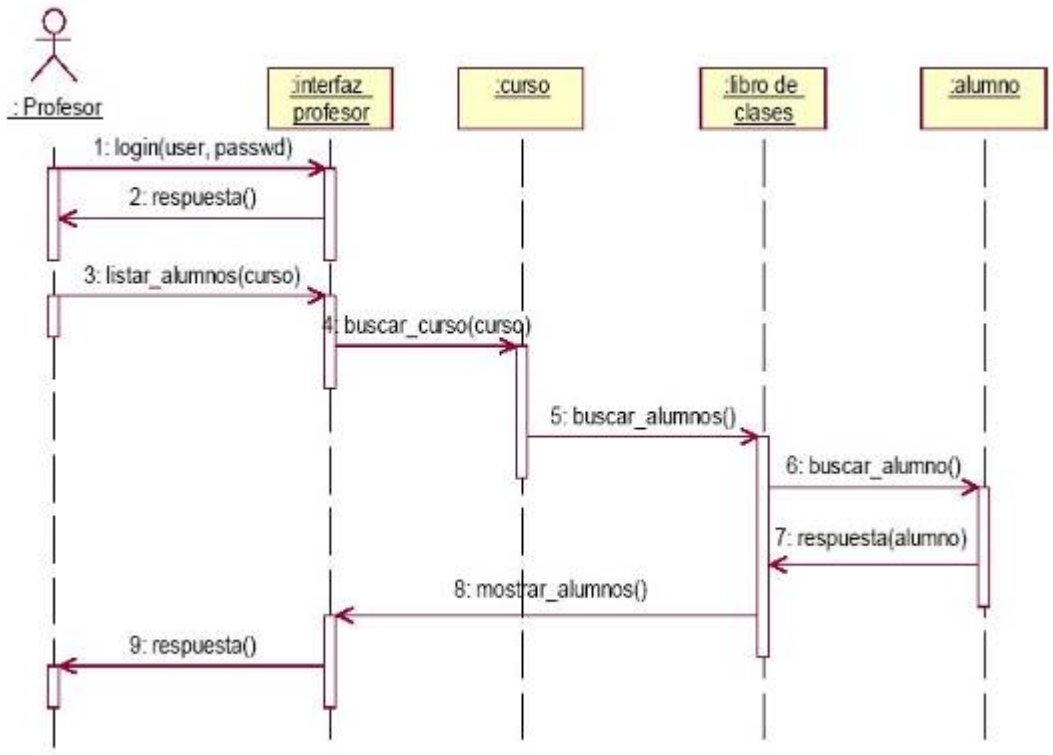

Ilustración E-2 Diagrama de Secuencia Listar Alumnos por Curso

E3 Listar Notas Asignatura

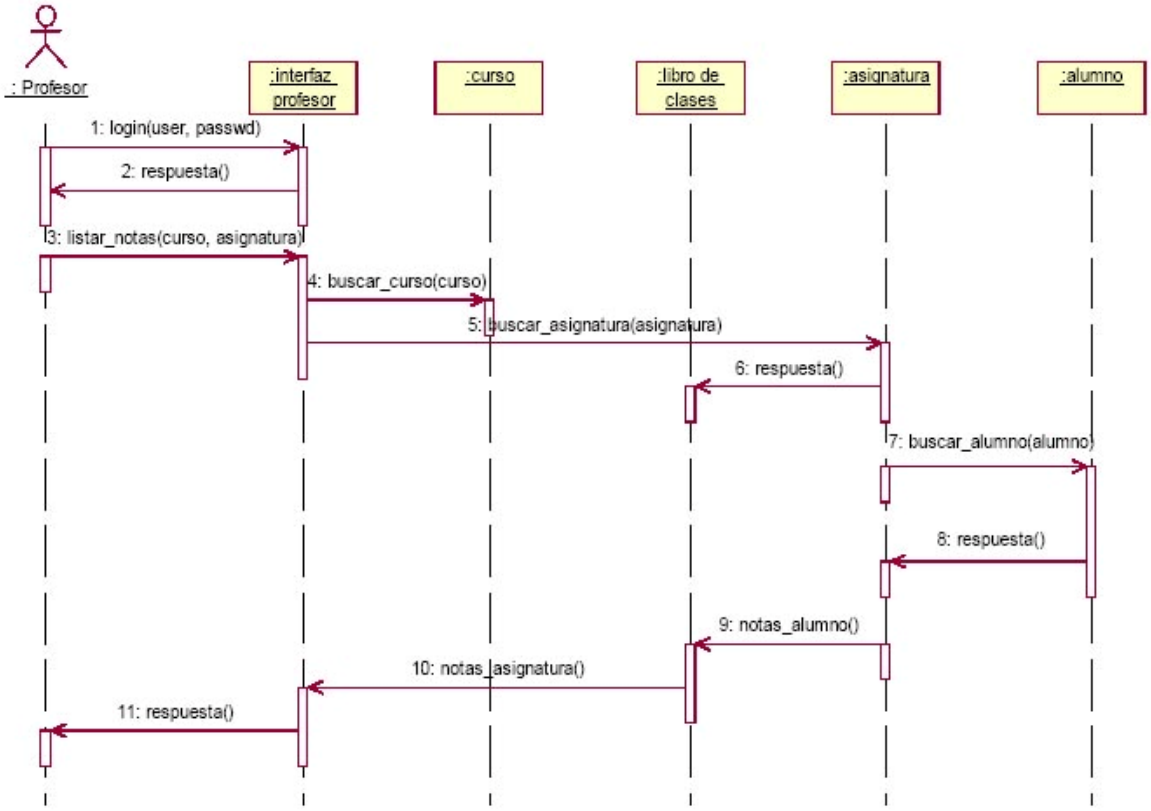

Ilustración E-3 Diagrama de Secuencia Listar Notas Asignatura

E4 Modificar Asistencia

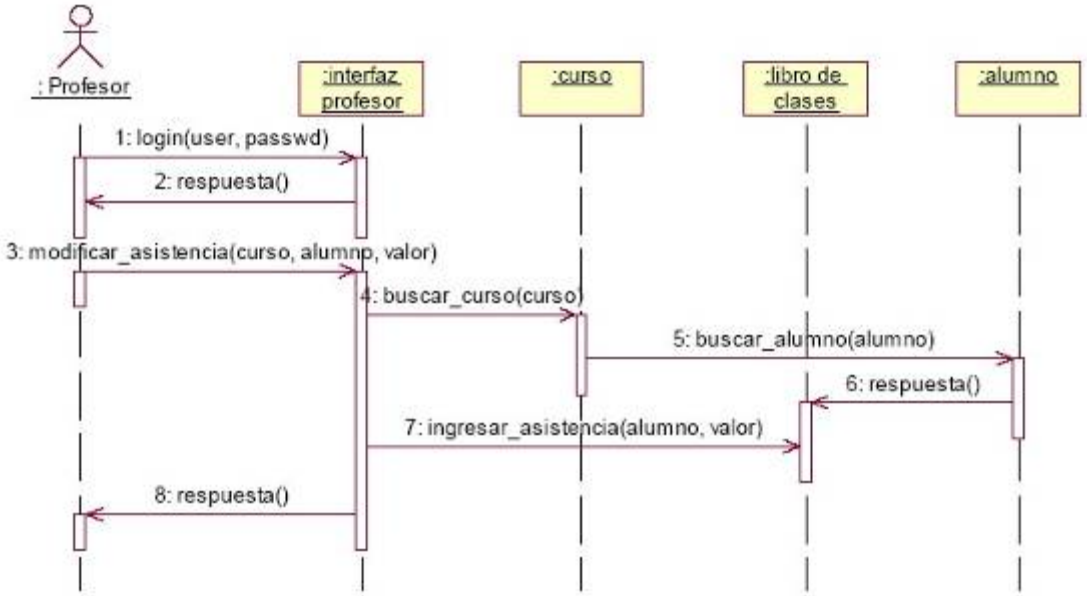

Ilustración E-4 Diagrama de Secuencia Modificar Asistencia

E5 Listar Alumnos por Curso

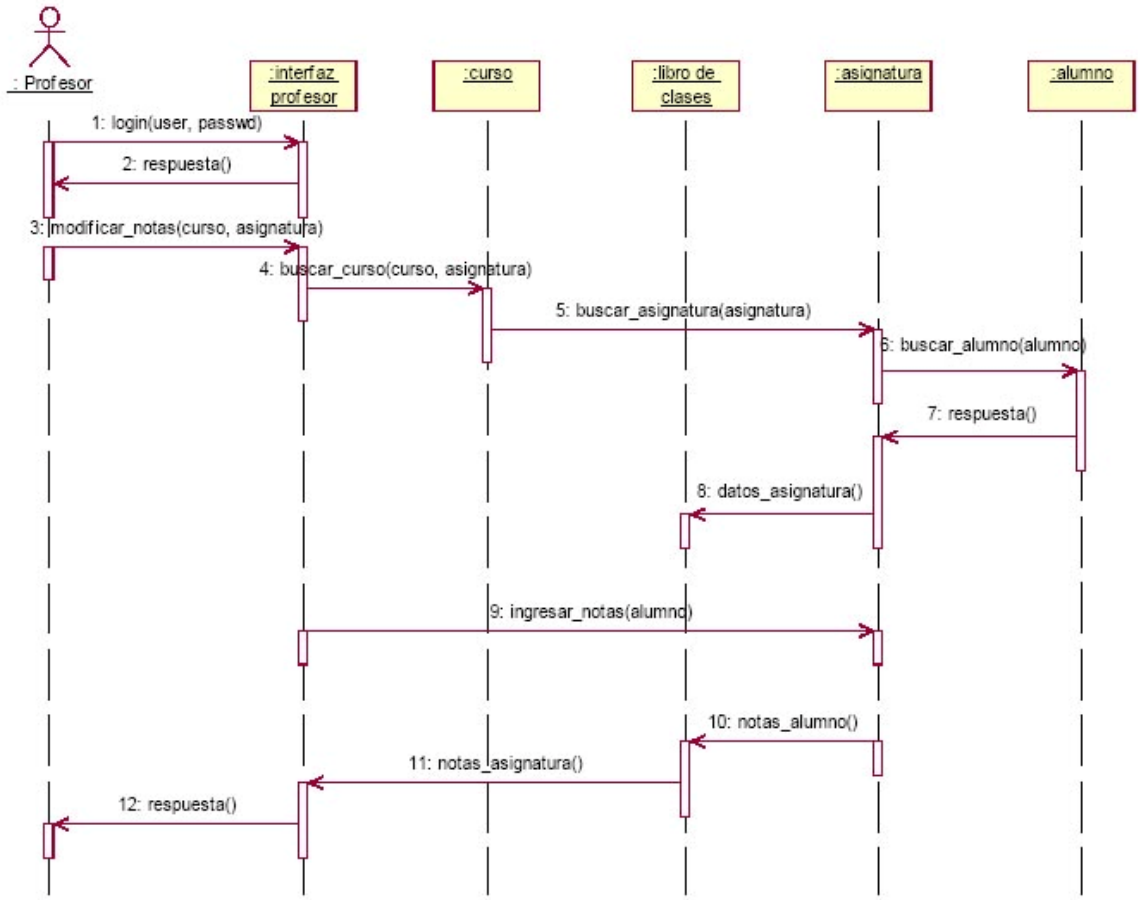

Ilustración E-5 Diagrama de Secuencia Modificar Notas Asignatura

E6 Cierre Libro de Clases

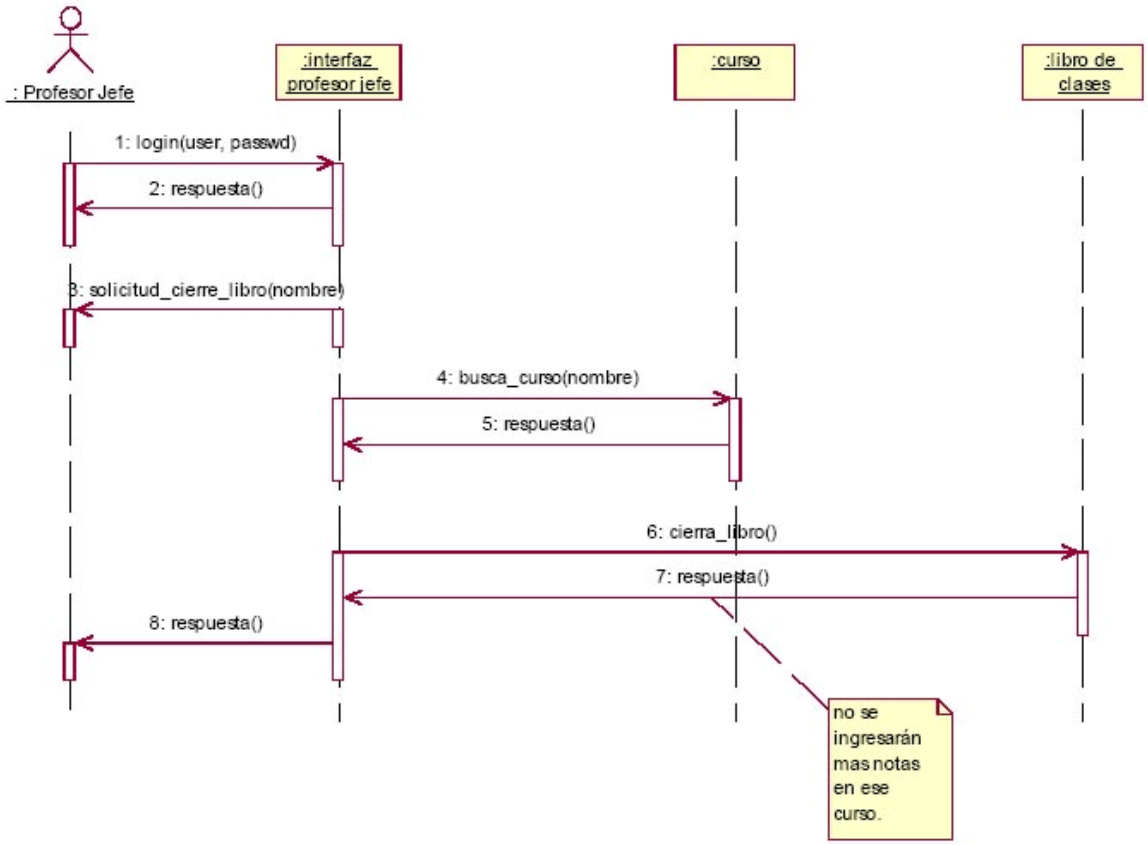

Ilustración E-6 Diagrama de Secuencia Cierre Libro de Clases

E7 Generar Informe de Notas

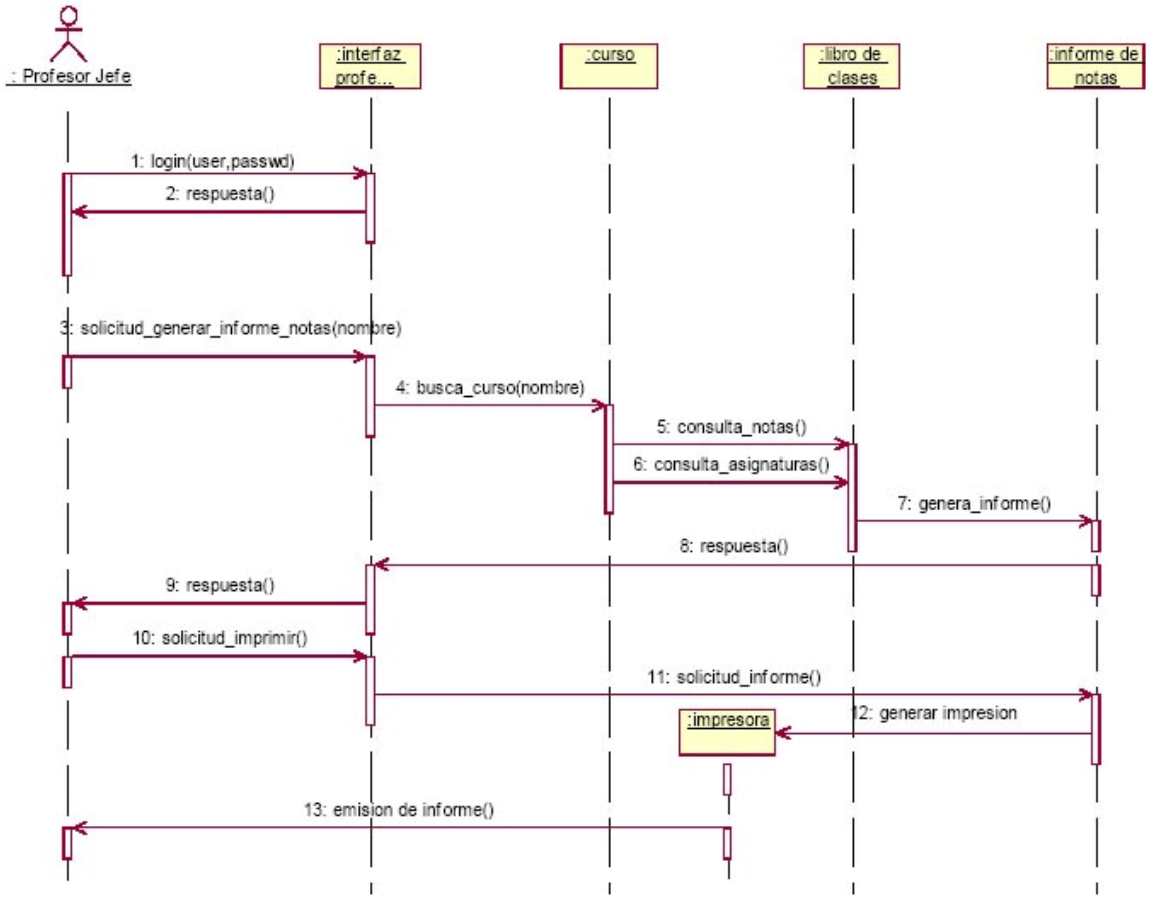

Ilustración E-7 Diagrama de Secuencia Generar Informe de Notas

E8 Generar Certificado

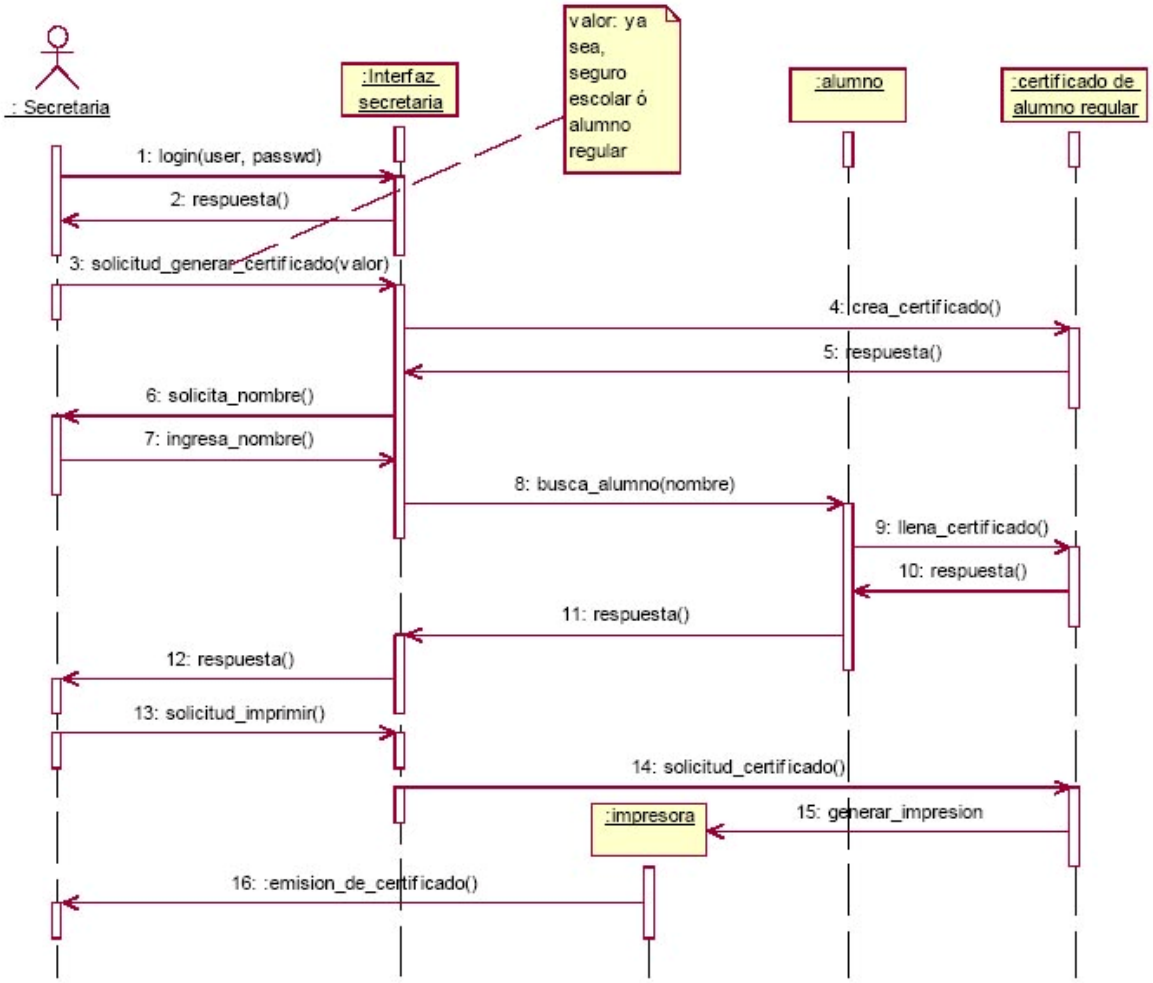

Ilustración E-8 Diagrama de Secuencia Generar Certificado

E9 Ingresar Ficha Alumno

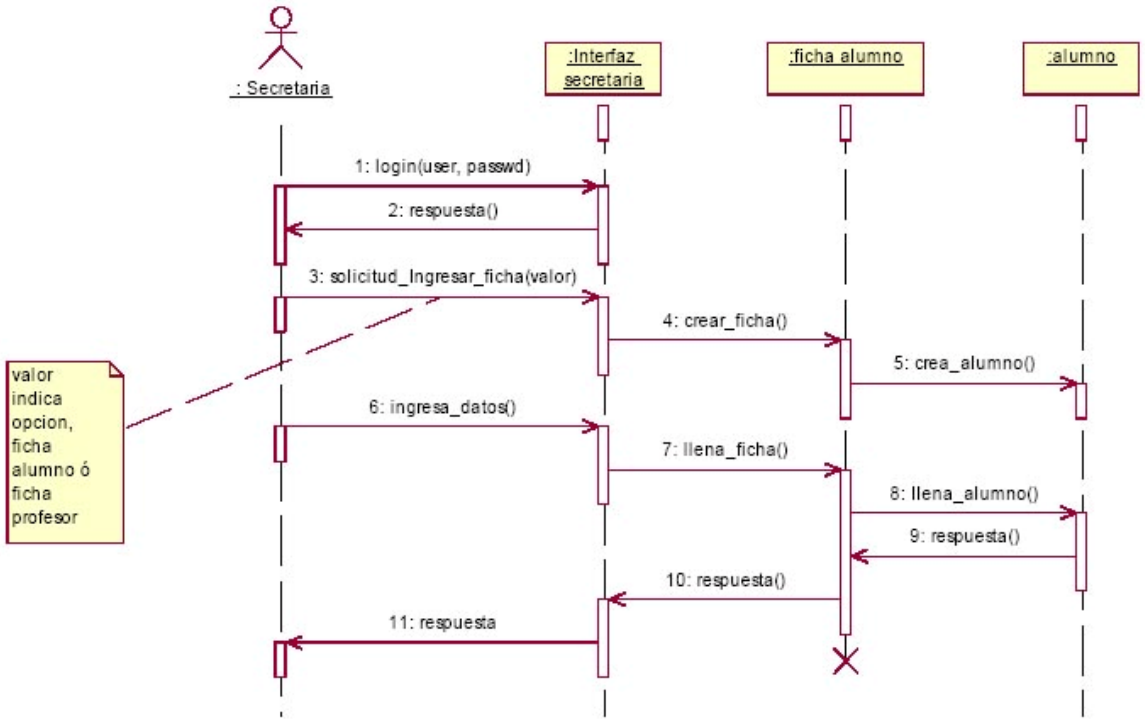

Ilustración E-9 Diagrama de Secuencia Ingresar Ficha Alumno

E10 Ingresar Ficha Profesor

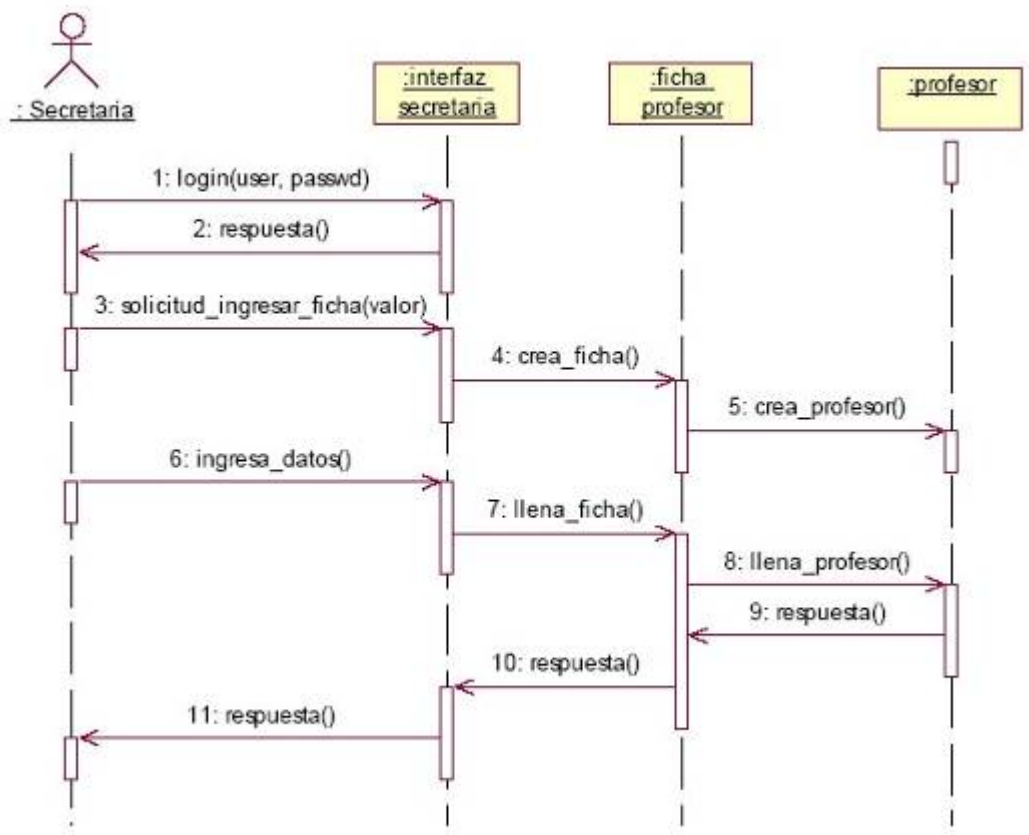

Ilustración E-10 Diagrama de Secuencia Ingresar Ficha Profesor

E11 Asignar Profesor a Asignatura

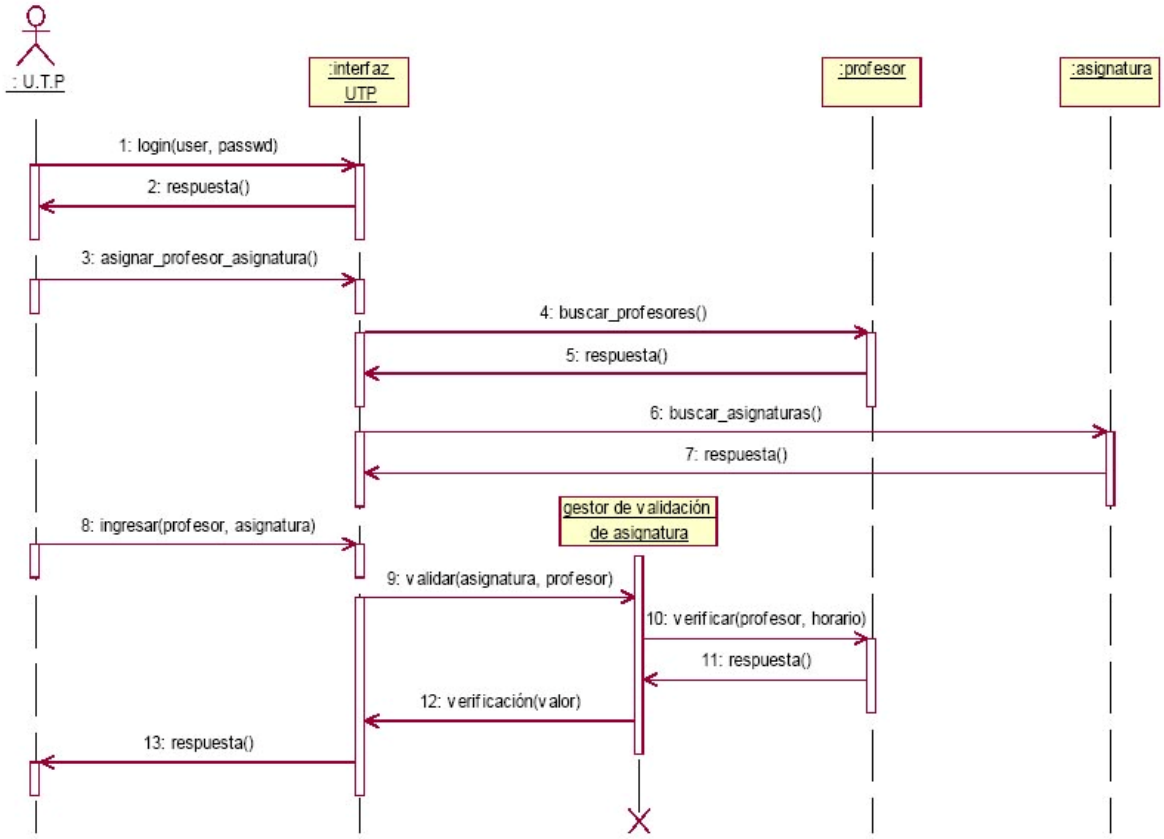

Ilustración E-11 Diagrama de Secuencia Asignar Profesor a Asignatura

E12 Asignar Profesor Jefe a Curso

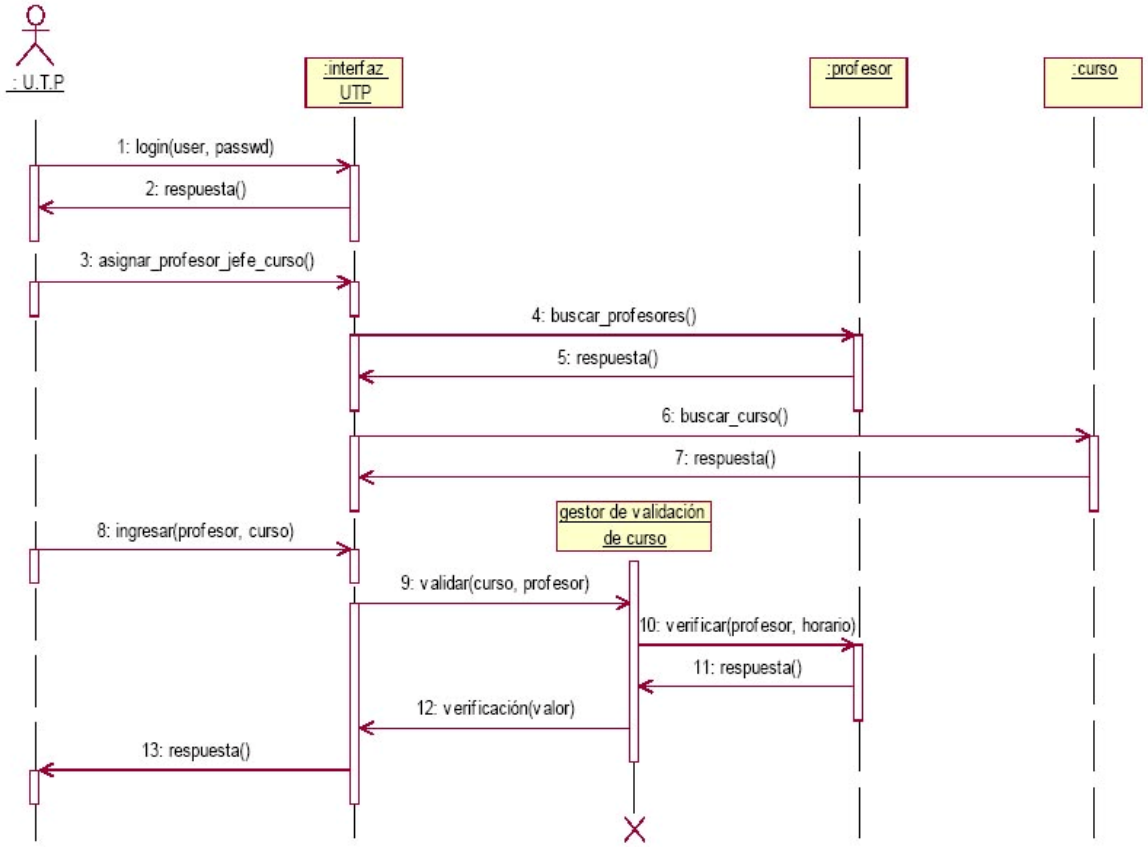

Ilustración E-12 Diagrama de Secuencia Asignar Profesor Jefe a Curso

E13 Cerrar Periodo

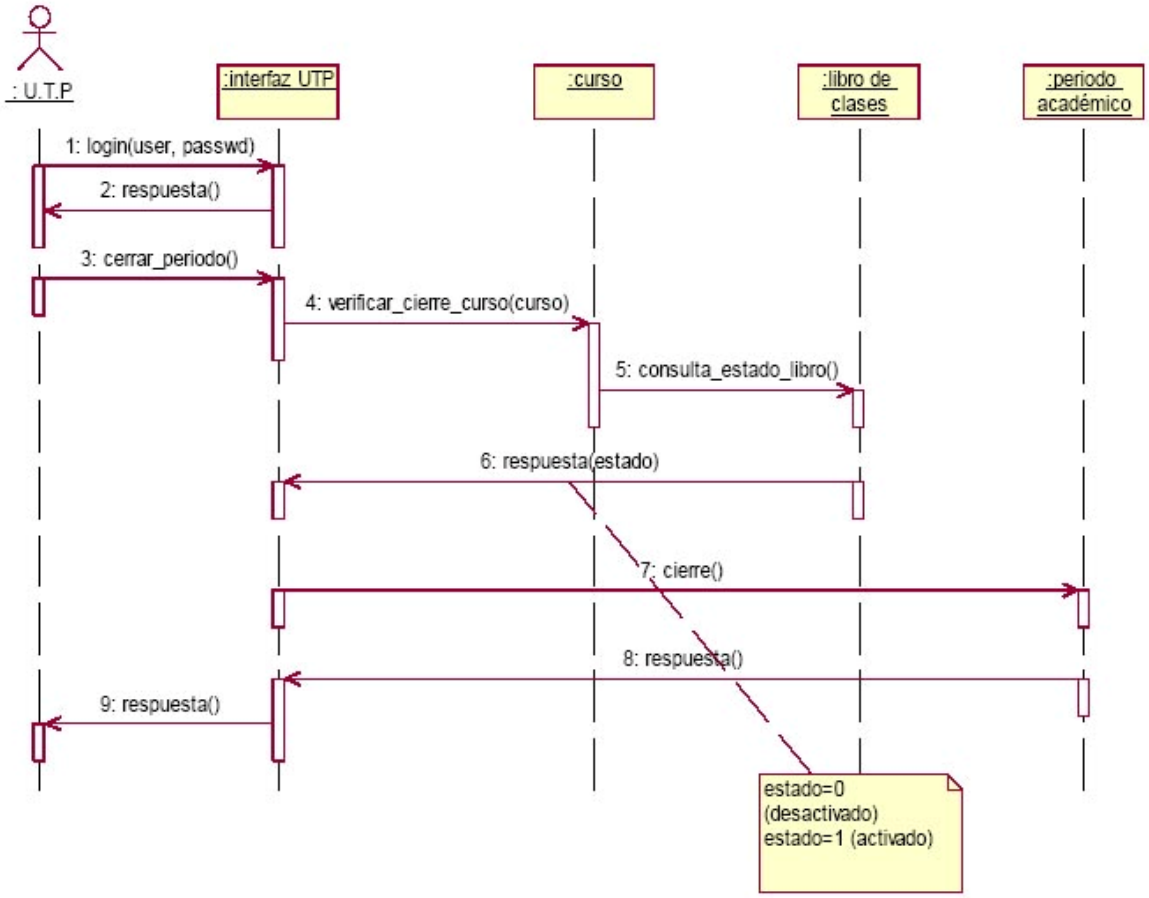

Ilustración E-13 Diagrama de Secuencia Cerrar Periodo

E14 Generar Actas

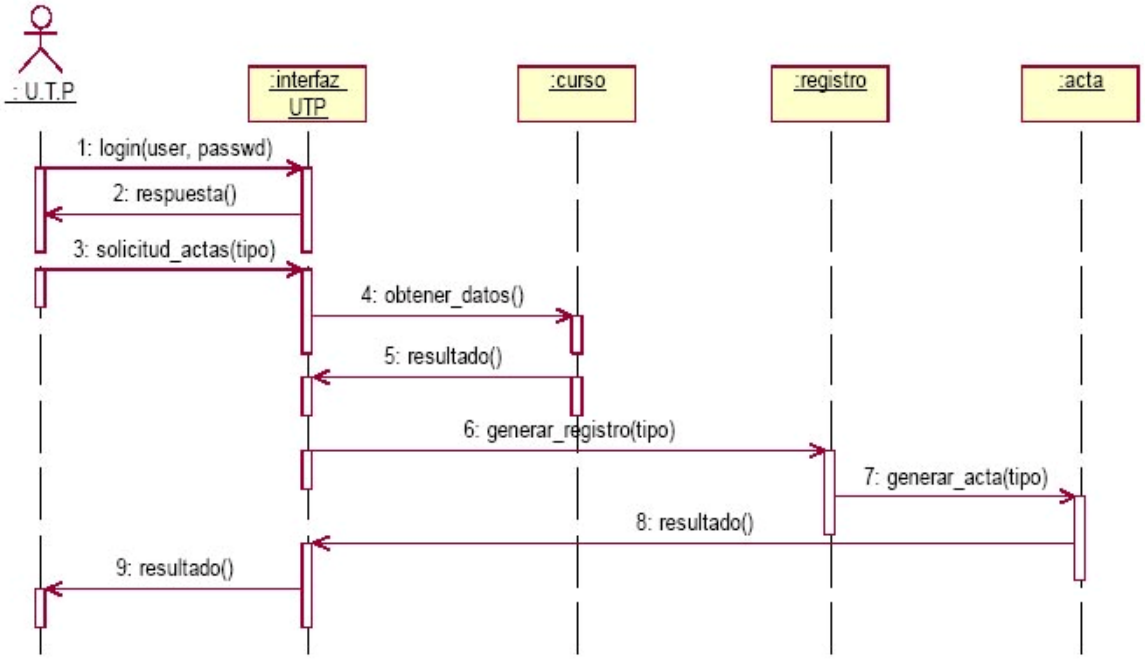

Ilustración E-14 Diagrama de Secuencia Generar Actas