Universidad de las Ciencias Informáticas

Facultad-2

Título: Sistema Automatizado para la Gestión de las Actividades en la Residencia Estudiantil.

Trabajo de Diploma

Para optar por el título de Ingeniero Informático

Autor(es): Lesnier Manuel González Zayas.
Carlos Guillén Chang.

Tutor(es): Ing. Maidelis Milanés Luque.

Co-tutor: Ing. Dayron Cruz Iñigo

Consultante: Ing. Yahima Vigo Valdés

Asesor: Ing. Temis Betancourt Villavicencio

Ciudad de la Habana, Junio 2008.

"Año 50 de la Revolución"

DECLARACION DE AUTORIA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Lesnier Manuel González Zayas

_____

Firma del Autor

Carlos Guillén Chang

_____

Firma del Autor

Maidelis Milanés Luque

_____

Firma del Tutor

OPINIÓN DEL USUARIO DEL TRABAJO DE DIPLOMA

El Trabajo de Diploma, titulado Sistema para la gestión de los proyectos productivos, fue realizado en la Universidad de las Ciencias Informáticas. Esta entidad considera que, en correspondencia con los objetivos trazados, el trabajo realizado le satisface

- ☐ Totalmente
- ☐ Parcialmente en un ____ %

Los resultados de este Trabajo de Diploma le reportan a esta entidad los beneficios siguientes (cuantificar):

_____

_____

_____

_____

_____

_____

Como resultado de la implantación de este trabajo se reportará un efecto económico que asciende a:
Y para que así conste, se firma la presente a los ____ días del mes de _____ del año _____.

_____	_____
Representante de la entidad	Cargo
_____	_____
Firma	Cuño

OPINIÓN DEL TUTOR DEL TRABAJO DE DIPLOMA

Título: Sistema Automatizado para la Gestión de las Actividades de la Residencia (SAGAR).

Autores: Lesnier Manuel González Zayas.
Carlos Guillén Chang.

El tutor del presente Trabajo de Diploma considera que durante su ejecución los estudiantes mostraron las cualidades que a continuación se detallan.

Por todo lo anteriormente expresado considero que los estudiantes están aptos para ejercer como Ingeniero en Ciencias Informáticas y propongo que se le otorgue al Trabajo de Diploma la calificación de _____.

_____ _____
Ing. Maidelis Milanés Luque Fecha

AGRADECIMIENTOS:

Con estas líneas pretendemos hacer llegar nuestro más sincero agradecimiento a todas las personas que de una manera u otra nos dieron su apoyo y estuvieron al tanto de nuestro desempeño como universitarios. Quisieramos citar en primer lugar la insustituible figura de nuestro Comandante en Jefe Fidel Castro Ruz, como creador de este gran proyecto del cual formamos parte y que hoy vemos hecho realidad, agradecer a nuestros padres y familiares que a pesar de la lejanía nos dieron el aliento y el consejo adecuado hasta la culminacion de nuestros estudios. Agadecemos a todos nuestros amigos, compañeros de aula, profesores y trabajadores que en su momento cumplieron el papel de nuestros familiares y padres, haciéndonos cada día mejores como estudiantes y como persona.

Lesnier Manuel González Zayas.

Carlos Guillén Chang.

Dedicatoria

RESUMEN

La Universidad de las Ciencias Informáticas emerge como una de las instituciones que mayor expectativa asegura en el orden económico, político y social de nuestro país. Este proyecto, concebido como la primera universidad surgida al calor de la Batalla de Ideas, tiene características que la distinguen como única de su tipo en el sistema educacional cubano. Caracterizada por su dinamismo, hace de la comunidad universitaria un ente intranquilo y enérgico; por su flexibilidad es totalmente adaptable al cambio; por su diversidad se convierte en un termómetro de disímiles indicadores; y por su extensión y concurrencia de procesos es una fiel muestra de una ciudad del futuro.

Hacer de la Universidad de las Ciencias Informáticas una ciudad digital, una ciudad donde todos y cada uno de los procesos estén sustentados sobre una infraestructura tecnológica capaz de responder a las necesidades de los que aquí conviven, es una tarea en la que directivos, trabajadores y estudiantes estamos inmersos desde sus inicios en el 2002.

Nuestra Universidad cuenta con áreas de residencia dentro de las cuales se encuentran distribuidos los estudiantes por facultad, donde estos realizan varias actividades importantes como son: la guardia estudiantil y el servicio de Cuartelería, por mencionar algunas de ellas

Debido a la gran cantidad de estudiantes que componen un área de residencia se hace necesario un sistema que facilite la distribución de la misma, y que a su vez brinde la posibilidad de planificar las actividades mencionadas.

PALABRAS CLAVES

Sistema, Residencia, Software, Automatización.

TABLA DE CONTENIDOS

INTRODUCCION

La residencia estudiantil de nuestra facultad cuenta con un total de 7 edificios, los cuales poseen diferentes capacidades para dar alojamiento (beca) y la atención necesaria a todos los estudiantes de la misma. Ellos se dividen en edificios de hembras y edificios de varones. Cada edificio tiene un trabajador responsable del mismo, el cual es el encargado de velar por el cumplimiento de todas las actividades de los estudiantes en la beca y emitir una evaluación de los mismos.

Pero contamos con una **situación problémica** y es que en estos momentos existe ineficiencia en la planificación de la cuartelaría y la guardia, y no existe una vía que facilite la distribución de los estudiantes en la residencia. Todas estas actividades se realizan actualmente de forma manual.
De tal situación se deriva el siguiente **problema científico** ¿Cómo desarrollar un sistema automatizado para gestionar los procesos de la residencia estudiantil?
Definiéndose en la investigación como nuestro **objeto de estudio** la residencia estudiantil de la Universidad de las Ciencias Informáticas (UCI).
Tomando como **campo de acción** los procesos que se realizan en la residencia estudiantil de la facultad 2
Con el fin de resolver el problema planteado se tiene como **objetivo general** Desarrollar un sistema que permita el funcionamiento automatizado de los procesos en la residencia estudiantil, como son:
La distribución de la beca
El plan de guardia.
El plan de servicio de la Cuartelería
Generar un Reporte del estudiante
Se tiene las siguiente **pregunta científica** ¿ Porqué es importante desarrollar la aplicación en una plataforma de software libre ?
¿Qué se conoce como software libre?

El software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. De modo más preciso, se refiere a cuatro libertades de los usuarios del software:

- La libertad de usar el programa, con cualquier propósito.
- La libertad de estudiar cómo funciona el programa, y adaptarlo a tus necesidades. El acceso al código fuente es una condición previa para esto.

- La libertad de distribuir copias, con lo que se puede ayudar a un compañero o entidad.
- La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie. El acceso al código fuente es un requisito previo para esto.

La migración a software libre rompe las barreras que el software propietario mantiene sobre el desarrollo de la comunidad internacional, sobre todo en países pobres, y amplía los campos de conocimientos de la sociedad, aumentando las posibilidades de desarrollar y crear nuevas herramientas que resuelvan los problemas fundamentales de la sociedad.

La licencia que rige este tipo de software, llamada GPL (GNU General Public License, Licencia Pública General), se creó para proteger la integridad del software libre y evitar que nadie restrinja o utilice su circulación.

Todo programa o código sujeto a esta licencia permite:

- Descargar, utilizar y copiar libremente dicho programa, siempre que se incluya el copyright correspondiente y todo el texto correspondiente a la licencia.

- Incluir modificaciones y mejoras en dicho software y distribuirlas libremente, siempre que se especifique qué cambios se han hecho en el software original y la fecha de realización.

- Dicho software debe distribuirse libremente, sin costo alguno. Como mucho, se permite cobrar una cuota que costee los gastos de reproducción (por ejemplo, el CD en el que se graba).

La UCI como instituto para la formación de un profesional integral, conformador de un espacio para el estudio, investigación y desarrollo, debe estar a la vanguardia en la ciencia y la tecnología. Por esto, recibe gran importancia el desarrollo del software libre, herramienta necesaria para un proceso de aprendizaje con vista hacia el futuro, con proyección, con evolución y revolución permanente.

Para cumplir el objetivo de la investigación, se llevaron a cabo las siguientes **tareas investigativas:** Efectuar entrevistas al cliente (Vice Decano Extensión) para obtener el conocimiento de cómo se realizan las actividades en la actualidad y confeccionar los primeros modelos de estos procesos.

Estudiar las normas establecidas para la confección de modelos de análisis y diseño de sistemas, teniendo en cuenta las características del sistema a diseñar y los requerimientos establecidos por el cliente.

Estudiar la situación actual del problema en cuestión y comparar con otras aplicaciones que den solución a tareas semejantes, y lograr acogerlas al medio de trabajo de la Residencia.
Estudiar y dominar con solidez los elementos fundamentales de ingeniería y gestión de software, y programación a utilizar para desarrollar un sistema automatizado

Para lograr una mejor organización del contenido a tratar, se decidió que el documento estuviera estructurado en capítulos, los cuales se describen a continuación:

- ***CAPITULO 1. FUNDAMENTACION TEORICA****:* Se describen los principales aspectos y conceptos de relevancia que han sido objeto de análisis a lo largo de la investigación. Se explica el por qué se hace necesario el uso de cada herramienta, metodologías y lenguajes de programación para el sistema que se desea implementar.

- ***CAPITULO 2. CARACTERISTICAS DEL SISTEMA****:* Se aborda el objeto de estudio, problema y situación problémica, se detalla el objeto de automatización, la información que se maneja, se realiza una descripción general de la propuesta de sistema para el desarrollo de la aplicación Web. Se describen los requisitos funcionales y no funcionales con los que debe contar el sistema así como elementos clases del Negocio.

- ***CAPITULO 3. ANALISIS Y DISEÑO DEL SISTEMA****:* Aborda el flujo de trabajo Análisis y Diseño. Se definen las clases del análisis y el Modelo de clases de análisis. En el diseño se realizan los modelos de clases del diseño para cada caso de uso, se definen los diagramas de interacción por cada realización de casos de uso, se hace un modelo de las clases persistentes y la posterior creación del modelo de Datos (Base de Datos).

- ***CAPITULO 4. IMPLEMENTACION****:* Se aborda todo lo referente al flujo de trabajo Implementación en su desarrollo para esta aplicación, para ello se expone el diagrama de despliegue que corresponde a la aplicación y descrito en los capítulos anteriores. Quedan reflejados los diagramas de componentes que forman la base de la programación de este sistema. También se definen

elementos de las pruebas realizadas al sistema a través de su ciclo de vida y la final realización del mismo.

CAPITULO 5. ESTIMACION: Se aborda el estudio de la factibilidad del proyecto, debido a esto se profundiza en el conocimiento del análisis del costo, el esfuerzo y los beneficios que este reportará.

CAPÍTULO 1: FUNDAMENTACION TEORICA

1.1.Introducción

En este capítulo se explicará como es el actual funcionamiento de las actividades de la residencia de la facultad 2 de la Universidad de las Ciencias Informáticas (UCI), para un mejor entendimiento del problema a resolver, se realizará un estudio de todo lo referente al estado del arte del tema a desarrollar, así como las tendencias, tecnologías, herramientas, lenguajes y metodologías que serán utilizadas en el desarrollo de este proyecto, se pondrán ejemplos de otros sistemas que realicen actividades semejantes (si existen) y su desarrollo en la actualidad.

1.2.Trabajo Actual

En la residencia estudiantil de la facultad 2 de la Universidad de las Ciencias Informáticas (UCI) se lleva un control manual de todas las actividades por parte del Vice Decanato de Extensión Universitaria, así como por parte del responsable de cada edificio. El trabajo de planificación de guardia y de Cuartelería resulta en muchas ocasiones engorroso debido a que son muchos estudiantes y profesores, y hay que tener control de que no se repitan y de que les llegue a tiempo el plan trabajo a cumplir por estos. Debido a la gran carga de trabajo se emplea mucho tiempo a estas planificaciones, y prácticamente no terminan de hacer las de un mes y ya tienen que comenzar las de otro.

1.3.Sistemas existentes.

El sistema de Gestión de Información de la Facultad 8 (SGIF 8), es un sistema para gestionar toda la información referente a la facultad 8, posee 6 módulos, de estos uno es para gestionar gran parte de la información no docente que se lleva a cabo en esta facultad y vinculadas a la residencia estudiantil como son:

- Ubicar a los estudiantes en los apartamentos de los edificios con los que dispone la facultad.
- Conocer en qué edificio, paso de escalera y apartamento de la residencia se encuentra ubicado un estudiante de la facultad o un grupo de estudiantes.
- Emitir partes de guardia.
- Conocer cómo se ha comportado la guardia estudiantil de una brigada específica hasta el momento.
- Consultar los partes de la guardia estudiantil de cualquier día.

El sistema ha sido desarrollado utilizando una arquitectura cliente-servidor.

1.4 Metodología.

Metodología, del griego (metà "mas allà" odòs "camino" logos "estudio"). Se refiere a los métodos de investigación que se sigue para alcanzar una gama de objetivos en una ciencia. Aun cuando el término puede ser aplicado a las artes cuando es necesario efectuar una observación o análisis más riguroso o explicar una forma de interpretar la obra de arte. En resumen son el conjunto de métodos que se rigen en una investigación científica o en una exposición doctrinal. [1]

Metodología: Conjunto de métodos empleados para el desarrollo de sistemas automatizados. [2]

Las metodologías proporcionan:

- Guías para estimar costos.
- Manejo del proyecto en las tareas y entregas
- Medidas y métricas
- Formas definidas y dirección en las entregas de la construcción.
- Políticas y procedimientos para garantizar la calidad del software.
- Descripciones de los roles y programas de entrenamiento detallados.
- Ejemplos totalmente trabajados.
- Ejercicios de entrenamiento.
- Técnicas para adaptar el método.
- Técnicas definidas.[2]

1.4.1Metodología orientada a objetos.

1.4.1.1 RUP

(Rational Unified Process) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos. [3]

Se escoge RUP para llevar a cabo este proyecto por no ser un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

Principales características.

- Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo).
- Pretende implementar las mejores prácticas en Ingeniería de Software.
- Desarrollo iterativo.
- Administración de requisitos.
- Uso de arquitectura basada en componentes.
- Control de cambios.
- Modelado visual del software.
- Verificación de la calidad del software. [3]

Durante el ciclo de vida de *RUP* presenta tres características fundamentales:

Dirigidos por casos de uso: Los casos de uso representan los requisitos funcionales, guían el diseño, implementación y prueba de un sistema; es decir, guían el proceso de desarrollo. El proceso de desarrollo sigue un hilo, avanza a través de una serie de flujos de trabajo que parten de los casos de uso.

Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción.

Iterativo e incremental: RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto.

RUP divide el proceso de desarrollo de un producto en cuatro fases:

- **Inicio** (Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema).
- **Elaboración** (Se especifican en detalle la mayoría de los casos de uso y se diseña la arquitectura).
- **Construcción** (Es la línea base de la arquitectura)
- **Transición**

En cada una de estas fases se desarrollan los flujos de trabajos definidos por RUP, que son los siguientes:

- **Modelamiento del negocio:** Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- **Requerimientos:** Define qué es lo que el sistema debe hacer, se identifican las funcionalidades y restricciones que se imponen.
- **Análisis y Diseño:** Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.
- **Implementación:** Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- **Prueba** (Testeo): Busca los defectos a los largo del ciclo de vida.
- **Instalación:** Produce *release* del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.
- **Administración del proyecto:** Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- **Administración de configuración y cambios:** Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, etc.
- **Ambiente:** Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado para software que permite la modelación de sistemas orientados a objetos. RUP lo utiliza para la visualización, especificación, construcción y documentación de los artefactos que se generan en el proceso de desarrollo de un software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

El UML cuenta con varios tipos de modelos, los cuales muestran diferentes aspectos de las entidades representadas [4].

Tipos de modelo

- Funcional: Muestra la funcionalidad del sistema desde el punto de vista del usuario, incluye:
  - Diagramas de caso de uso
- Objetos: Muestra la estructura y la subestructura del sistema usando objetos, atributos, operaciones y asociaciones, incluye:
  - Diagramas de clase
- Dinámico: Muestra el comportamiento interno del sistema, incluye:
  - Diagramas de secuencia
  - Diagramas de actividad
  - Diagramas de estado [4]

- Los ***Diagramas de Estructura*** enfatizan en los elementos que deben existir en el sistema modelado:
- Diagrama de clases
- Diagrama de componentes
- Diagrama de objetos
- Diagrama de estructura compuesta (UML 2.0)
- Diagrama de despliegue
- Diagrama de paquetes
- Los ***Diagramas de Comportamiento*** enfatizan en lo que debe suceder en el sistema modelado:
- Diagrama de actividades
- Diagrama de casos de uso
- Diagrama de estados
- Los ***Diagramas de Interacción*** son un subtipo de diagramas de comportamiento, que enfatiza

sobre el flujo de control y de datos entre los elementos del sistema modelado:

- Diagrama de secuencia
- Diagrama de colaboración
- Diagrama de tiempos (UML 2.0)
- Diagrama de vista de interacción (UML 2.0) [5]

1.4.1.2 Microsoft Solution Framework (MSF).

MSF en una metodología de *software* flexible, se puede adaptar a proyectos de distintos tamaños y complejidades [6].

MSF (*Microsoft Solution Framework*) provee un conjunto de modelos, principios y lineamientos para diseñar y desarrollar soluciones empresariales de manera que todos los elementos de un proyecto (como: la gente, procesos, y herramientas) puedan ser administrados apropiadamente, provee prácticas probadas para planear, diseñar, desarrollar e implementar soluciones empresariales exitosamente [6].

MSF tiene las siguientes **características**: [7]

Adaptable: es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.

Escalable: puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas a más.

Flexible: es utilizada en el ambiente de desarrollo de cualquier cliente.

Tecnología Agnóstica: porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

Consta de las siguientes fases: [6]

- **Visión:** (*envisioning*) descripción general de las metas y restricciones del proyecto. Aquí se identifican las tareas y entregables.

- **Planeación:** el equipo determina que desarrollar y planea como crear la solución. El equipo prepara la especificación funcional, crea el diseño de la solución y prepara los planeas de trabajo, estimaciones de costos, y agenda el cumplimiento de los entregables.
- **Desarrollo:** creación de código que implementa la solución y su documentación.
- **Estabilización:** el equipo integra, carga y realiza pruebas de la solución.
- **Implementación:** el equipo implementa la solución tecnológica y sus componentes, estabiliza la implementación, transfiere el proyecto a producción y soporte, y obtiene la aprobación final del cliente sobre el proyecto.

1.4.1.3 Extreme programing (XP).

Es una de las metodologías de desarrollo de *software* para proyectos de corto plazo, corto equipo y cuyo plazo de entrega era ayer. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto [7].

Características de XP:

Pruebas Unitarias: se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro, podamos hacer pruebas de las fallas que pudieran ocurrir. Es como si nos adelantáramos a obtener los posibles errores.

Refabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.

Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa [7].

1.4.1.4 Selección de la metodología a utilizar.

Después de un análisis del las metodologías antes mencionadas y de ver sus características se ha seleccionado la metodología RUP como apoyo en el desarrollo de la aplicación, ya que se basa fundamentalmente en la documentación del software, utiliza UML para visualizar, especificar, construir

y documentar los artefactos que se generan en el proceso de desarrollo de un software, y de esta manera organiza y simplifica el trabajo de una forma muy eficiente.

1.5.Herramientas Case.

Las Herramientas CASE (***C****omputer* ***A****ided* ***S****oftware* ***E****ngineering*, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, calculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. [8]

Objetivos

- Mejorar la productividad en el desarrollo y mantenimiento del software.
- Aumentar la calidad del software.
- Mejorar el tiempo y coste de desarrollo y mantenimiento de los sistemas informáticos.
- Mejorar la planificación de un proyecto.
- Aumentar la biblioteca de conocimiento informático de una empresa ayudando a la búsqueda de soluciones para los requisitos.
- Automatizar, desarrollo del software, documentación, generación de código, pruebas de errores y gestión del proyecto.
- Ayuda a la reutilización del software, portabilidad y estandarización de la documentación.
- Gestión global en todas las fases de desarrollo de software con una misma herramienta.
- Facilitar el uso de las distintas metodologías propias de la ingeniería del software. [8]

1.5.1 Rational Rose.

Herramienta CASE que da soporte al modelado visual con UML ofreciendo distintas perspectivas del sistema. Da soporte al Proceso Unificado de Rational (RUP):

- Modelado de Negocio
- Captura de Requisitos (parcial)
- Análisis y Diseño (Completo)

- Implementación (como ayuda)
- Control de Cambio y gestión de configuración [9]

Las vistas de Rose son las siguientes:

a) **La Vista de Casos de Uso**, *Use Case View*, que es la vista en la que se presenta el comportamiento deseado del sistema: en ella se encontrarían los modelos relacionados con la captura de requisitos, en esta vista se ubicarían el modelo del negocio, el modelo conceptual, el modelo de casos de uso del sistema y los diagramas de secuencia del sistema.

b) **La Vista Lógica***, Logical View*, Están presente los modelos que muestran el vocabulario y la funcionalidad (estructura y comportamiento) del sistema, a través de un conjunto de colaboraciones que realizan los casos de uso de la vista de casos de uso (colaboraciones que se modelan mediante diagramas de clases y diagramas de interacción: secuencia y colaboración).

c) **La Vista de Componentes**, *Component View*, en la que se representa la implementación del sistema mediante componentes, la organización modular del software. Esta vista está relacionada con la gestión de la configuración del software. Los paquetes en esta vista se organizan en niveles. Un componente está relacionado con un archivo de software y un lenguaje de programación. Las clases de la vista lógica se asignarían a los componentes de la vista de componentes.

d) **La Vista de Despliegue**, *Deployment View*, en la que se modela la distribución o despliegue de los componentes a los nodos de procesamiento del sistema. Muestra la topología, distribución e instalación del sistema.

1.6.Sistemas Gestor de Bases de datos.

Sistema Gestor de Base de Datos (SGBD) es un conjunto de programas que permiten crear mantener una base de datos, asegurando su integridad, confidencialidad y seguridad.
Los SGBD deben cumplir:

- Definir una base de Batos: especificar tipos, estructuras y restricciones de datos.
- Construir la Base de Datos: guardar los datos en algún medio controlado por el mismo SGBD.
- Manipular la BD: Realizar consultas, actualizarla y generar informes.

Características de un SGBD:

- Control de la redundancia: La redundancia de datos tienen varios efectos negativos (duplicar los datos al actualizar, desperdicia espacio en disco, puede provocar inconsistencia de datos) aunque a veces es deseable por cuestiones de rendimiento.
- Restricción de los accesos no autorizados: cada usuario ha de tener unos permisos de accesos y autorización.
- Cumplimiento de las restricciones de integridad: el SGBD ha de ofrecer recursos para definir y garantizar el cumplimiento de las restricciones de integridad.

Ventajas del SGBD:

- ✓ Facilidad de manejo de grandes volúmenes de información.
- ✓ Gran velocidad en muy poco tiempo.
- ✓ Independencia del tratamiento de información.
- ✓ Seguridad de la información (acceso a usuarios autorizados), protección de información, de modificaciones, inclusiones, consulta.
- ✓ No hay duplicidad de información, comprobación de información en el momento de introducir la misma.
- ✓ Integridad referencial el terminar los registros.[10]

Desventajas del SGBD:

- ✓ El costo de actualización del hardware y software son muy elevados.
- ✓ El Costo (salario o remuneración) del administrador de la base de datos es grande.
- ✓ El mal diseño de esta puede originar problemas a futuro.
- ✓ Un mal adiestramiento a los usuarios puede originar problemas a futuro.
- ✓ Si no se encuentra un manual del sistema no se podrán hacer relaciones con facilidad.
- ✓ Generan campos vacíos en exceso.
- ✓ El mal diseño de seguridad genera problemas en esta.[10]

1.6.1 PostgreSQL(8.2).

PostgreSQL es el servidor de base de datos relacional orientada a objetos de software libre más potente que existe, liberado bajo la licencia BSD. [11]

Soporta:

- *Querys* complejos, incluyendo *subselects*.
- Integridad referencial *(Foreign Keys*).
- *Triggers*.
- Vistas (*Views*).
- Integridad Transaccional (ACID).
- Control de versionado concurrente (MVCC)[11]

Ventajas de PostgreSQL(8.2).

- ✓ Posee una gran escalabilidad. Es capaz de ajustarse al número de Unidades Centrales de Procesamiento (CPUs) y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta.
- ✓ Implementa el uso de *rollback*, subconsultas y transacciones, haciendo su funcionamiento mucho más eficaz.
- ✓ Tiene la capacidad de comprobar la integridad referencial, así como también la de almacenar procedimientos en la propia base de datos.
- ✓ PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos [12].

Desventajas de PostgreSQL(8.2):

- ✓ Tiene un límite de 8K por fila, aunque se puede aumentar a 32K, con una disminución considerable del rendimiento.
- ✓ Es lento [12].

1.6.2 MySQL

MySQL es un sistema de gestión de bases de datos relacional, licenciado bajo la GPL de la GNU. Su diseño multihilo le permite soportar una gran carga de forma muy eficiente. MySQL fue creada por la empresa sueca MySQL AB, que mantiene el copyright del código fuente del servidor SQL, así como también de la marca.

Aunque MySQL es software libre, MySQL AB distribuye una versión comercial de MySQL, que no se diferencia de la versión libre más que en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de no ser así, se vulneraría la licencia GPL.

Este gestor de bases de datos es, probablemente, el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración [13].

Características de MySQL

Las principales características de este gestor de bases de datos son las siguientes:

1. Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo.
2. Soporta gran cantidad de tipos de datos para las columnas.
3. Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc).
4. Gran portabilidad entre sistemas.
5. Soporta hasta 32 índices por tabla.
6. Gestión de usuarios y *passwords*, manteniendo un muy buen nivel de seguridad en los datos [13].

1.6.3 ORACLE

Oracle es un sistema de gestión de base de datos relacional (o RDBMS por el acrónimo en inglés de Relational Data Base Management System), fabricado por Oracle *Corporation*.

Se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando su:

- Soporte de transacciones.
- Estabilidad.
- Escalabilidad.
- Es multiplataforma.

1.6.4 Fundamentación del SGBD a escoger:

Una vez analizados los sistemas gestores de bases de datos antes mencionados viendo sus ventajas y desventajas, y principales características la aplicación se utilizará PostgreSQL(8.2) como SGBD, ya que posee gran escalabilidad, soporta una gran cantidad de peticiones simultaneas de forma correcta, Implementa el uso de *rollback's*, subconsultas y transacciones, haciendo su funcionamiento mucho más eficaz, tiene la capacidad de comprobar la integridad referencial, así como también la de

almacenar procedimientos en la propia base de datos, equiparándolo con los gestores de bases de datos de alto nivel.

MySQL carece de soporte para transacciones, rollback's y subconsultas, no posee integridad referencial, no es viable para su uso con grandes bases de datos, a las que se acceda continuamente, ya que no implementa una buena escalabilidad.

ORACLE presenta desventajas en la seguridad, además de ser muy caro.

1.7.Lenguajes de programación para la Web.

1.7.1 PHP (Personal Home Page).

Es un lenguaje de programación usado normalmente para la creación de páginas Web dinámicas. PHP es un acrónimo recursivo que significa "**P**HP **H**ypertext **P**re-processor" (inicialmente PHP Tools, o, ***P****ersonal* ***H****ome* ***P****age Tools*), y se trata de un lenguaje interpretado. Últimamente también puede ser utilizado para la creación de otro tipo de programas incluyendo aplicaciones con interfaz gráfica [14].

Ventajas del PHP.

- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos.
- Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos.
- Biblioteca nativa de funciones sumamente amplia e incluida.
- No requiere definición de tipos de variables.
- Tiene manejo de excepciones. [14]

1.7.2 Practical Extraction and Report Language (Perl)

Lenguaje interpretado que tiene varias utilidades, pero está principalmente orientado a la búsqueda, extracción y formateado de ficheros de tipo texto. También es muy usado para manejo y gestión de procesos (estado de procesos, conteo y extracción de parámetros característicos, etc.).

Es una combinación de las características de los lenguajes más usados por los programadores de sistemas, como son los *Shell* del sistema operativo UNIX, los utilidad (que incluye un lenguaje

interpretado propio) AWK para formateo y tratamiento de texto e incluso características de Pascal, aunque su potencia se basa en la similitud con las mejores características del lenguaje estructurado C [15].

Ventajas del PERL.

- Construcción de pequeños programas que pueden ser usados como filtros para obtener información de ficheros, realizar búsquedas.
- Se puede utilizar en varios entornos, como puede ser Windows 95, OS/2, sin realizar cambios de código, siendo únicamente necesario la introducción del interprete PERL correspondiente a cada sistema operativo.
- Es uno de los lenguajes mas utilizados en la programación de CGI scripts, que son guiones o scripts que utilizan el interface CGI (Common Gateway Interface), para intercambio de información entre aplicaciones externas y servicios de información.
- El mantenimiento y depuración de un programa en PERL es mucho más sencillo que la de cualquier programa en C. [15]

1.7.3 ASP.NET

Herramienta de desarrollo web comercializado por Microsoft. Es usado por programadores para construir sitios web domésticos, aplicaciones web y servicios XML. Forma parte de la plataforma .NET de Microsoft y es la tecnología sucesora de la tecnología Active Server Pages (ASP). [16]

Ventajas:

- Mejor rendimiento.
- Compatibilidad con herramientas de primer nivel.
- Eficacia y flexibilidad.
- Simplicidad.
- Facilidad de uso.
- Escalabilidad y disponibilidad.
- Posibilidad de personalización y extensibilidad.
- Seguridad. [17]

1.7.4 Java

Es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 1990. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel como punteros. [18].

1.7.5 Fundamentación del lenguaje a escoger.

Tabla 0–1 Comparación de los lenguajes de programación PHP, Java, ASP.NET y Perl.

Lenguajes de Programación / Características	PHP	Java	ASP.NET	Perl
Licencia:	Distribución libre.	*Sun Microsystems*	*Microsoft*.	Artística.
Multiplataforma:	Sí	Sí	Con previo pago y con el software adecuado.	Sí
Gestión de memoria:	Automática.	Automática.	Algunos defectos por solucionar.	Automática.
Compatibilidad con *PostgreSQL*:	Sí	Sí	Requiere un controlador para la conexión con las BD.	Sí
Orientación a objetos:	En cierta medida.	Sí	Sí	Sí
Precio:	Código libre.	No	Código propietario.	Código libre.

Al analizar la tabla1 y ver las ventajas que ofrece cada lenguaje de programación, Perl y Java superan a los demás lenguajes, pero no existe ningún compilador para estos lenguajes, Perl presenta como desventaja además que consume muchos recursos en la maquina, y se torna lento en lenguajes de bajo nivel, java por su parte presenta alta complejidad tecnológica.

Se decide PHP como lenguaje para desarrollar la aplicación Web debido que se pueden hacer grandes cosas con pocas líneas de código, viene acompañado por una excelente biblioteca de funciones que

permite realizar cualquier labor (acceso a base de datos, encriptación, envío de correo, gestión de un e-*commerce*, XML, creación de PDF), es multiplataforma, funciona en todas las plataformas que soporten apache, es software libre. Se puede obtener en la Web y su código esta disponible bajo la licencia GPL.

1.8. Arquitectura

La arquitectura de Software es la estructura jerarquica de los componentes del programa (modulos), la manera en que los componentes interactuan y la estructura de datos que van a utilizar los mismos.[27]

1.8.1 Arquitectura Cliente-Servidor: en Capas

La arquitectura cliente-servidor es una nueva tendencia en el desarrollo de redes locales, que tiene como objetivo optimizar el uso tanto del hardware como del software a través de la separación de funciones: el cliente que maneja la porción de la aplicación y el servidor que administra los procesos de almacenamiento y recuperación de los datos.

Puede presentarse como uno a varios clientes y uno o más servidores, junto con un sistema operativo y una plataforma de comunicación para formar un sistema cooperativo que permita la computación distribuida, el análisis y la presentación de datos. Un único servidor típicamente sirve a una multitud de clientes, ahorrando a cada uno de ellos el problema de tener la información instalada y almacenada localmente.

Características de la arquitectura Cliente/Servidor:

El servidor presenta una interfaz única y bien definida a todos sus clientes.

El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.

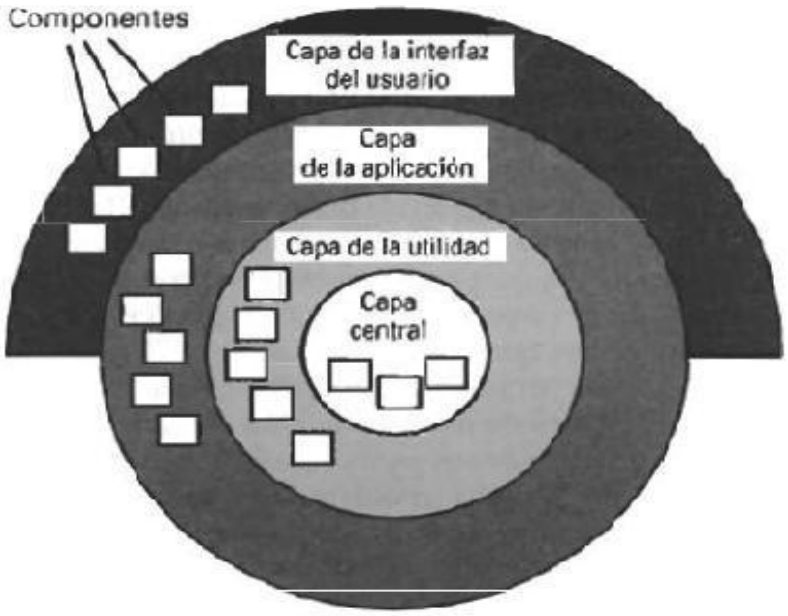
El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.

Los cambios en el servidor no afectan al cliente.[28]

1.8.2. Estilo Arquitectonico:

Arquitectura Estratificada

Se crean diferentes capas y cada una realiza operaciones que progresivamente se aproximan mas al cuadro de instrucciones de la maquina. En la capa mas externa los componentes sirven a las operaciones de interfaz usuario. En la capa interna los componentes realizan operaciones de interfaz del sistema. Las capas intermedias proporcionan servicios de utilidad y funciones del software de aplicaciones.[27]



1.8.3 Patrón de arquitectura de Software

Modelo Vista Controlador (MVC)

Modelo Vista Controlador (MVC) es un patrón de arquitectura 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, el modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio y el controlador es el responsable de recibir los eventos de entrada desde la vista.[6]

Descripción del patrón

Modelo: Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos; por ejemplo, no permitiendo comprar un número de unidades negativo, calculando si hoy es el cumpleaños del usuario o los totales, impuestos o importes en un carrito de la compra.

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

Controlador: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista[19]

1.9.Frameworks que implementan el Modelo Vista Controlador (MCV)

1.9.1 CakePHP

CakePHP es un marco de desarrollo rápido para PHP que proporciona una arquitectura extensible para el desarrollo, mantenimiento y despliegue de aplicaciones. Usando comúnmente conocidos como patrones de diseño MVC y ORM en el marco del Convenio sobre la configuración de paradigma, CakePHP reduce los costes de desarrollo y ayuda a los desarrolladores escribir menos código [20]

CakePHP es un framework (entorno de trabajo) libre y de código abierto para el desarrollo en PHP. Es una estructura de librerías, clases y una infraestructura run-time (en tiempo de ejecución) para programadores de aplicaciones web originalmente inspirado en el framework Ruby On Rails

Características CakePHP:

• CakePHP es principalmente el más avanzado framework MVC, con algunos módulos añadidos en la parte superior.

• Se puede manejar la mayoría de material del proyecto a desarrollar, y que incluye el soporte a Ajax y validación de datos.

• También cuenta con un módulo de autenticación de usuario único llamado *'Access Lists'*, que se puede utilizar para dar acceso a los diferentes usuarios de diferentes partes de su sitio web con CakePHP.

• *Scaffolding* de las aplicaciones.

• Componentes de seguridad

1.9.2 Symfony Project

Características:

- Proyecto de Symfony es un framework muy amplio, e incluye un verdadero ORM, de nombre Propel, que es otro proyecto de código abierto y, probablemente, una de las mejores soluciones ORM para PHP.
- Incluye Creole para la capa de abstracción de base de datos y Mojavi para la capa Model-View-Controller.
- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo

1.9.3 CodeIgniter

- CodeIgniter es relativamente un nuevo framework, por los fabricantes de ExpressionEngine, y parece muy prometedor.
- Está inspirado en Ruby on Rails, y que ofrece una gran parte de la misma funcionalidad, como los Scaffolding.
- Tiene una excelente documentación, y se han incluido vídeo tutoriales.
  CodeIgniter es un poderoso PHP marco con una muy pequeña huella, construido para los codificadores de PHP que necesita un simple y elegante conjunto de herramientas para crear todo lo que ofrece aplicaciones web. Si eres un programador que vive en el mundo real de alojamiento compartido y cuentas de clientes con los plazos, y si estás cansado de ponderously gran profundidad y marcos de indocumentados. [26]

CodeIgniter es adecuado para usted si...

- Usted quiere un marco con una pequeña huella.
- Usted necesita un rendimiento excepcional.
- Usted necesita una amplia compatibilidad con el estándar de alojamiento cuentas que se ejecutan una gran variedad de versiones de PHP y configuraciones.
- Usted quiere un marco que requiere la configuración de casi cero.
- Usted quiere un marco que no requiere que usted use la línea de comandos.
- Usted quiere un marco que no requiere que se adhieran a la codificación de normas restrictivas.
- Usted no se ha interesado en gran escala monolítico librerías como PEAR.
- Usted no quiere verse obligado a aprender un lenguaje de plantillas (aunque una plantilla analizador está disponible, opcionalmente, si lo desea uno).
- Usted quiere evitar la complejidad, favoreciendo soluciones simples.[26]

1.9.4 Selección del *Frameworks* a utilizar.

Después de analizar las características particulares seleccionamos *CodeIgniter* como Frameworks para la aplicación pues cuenta con una serie de funcionalidades de gran importancia para el desarrollo

de nuestro sistema, además permite trabajar remoto, y trabaja con servidor apache, además de contar con una buena documentación.

1.10.Servidor Web Apache

El servidor HTTP Apache es un *software* (libre) servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Windows, Macintosh y otras

Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido.

Ventajas:

- Modular.
- Open source.
- Multi-plataforma.
- Extensible.
- Popular (fácil conseguir ayuda/soporte).
- Gratuito.

Apache2.0-Principales Mejoras

- **Nuevo sistema de configuración y compilación**
- El sistema de configuración y compilación ha sido escrito de nuevo desde cero para basarlo en autoconf y libtool. Esto hace que el sistema de configuración de Apache se parezca ahora más al de otros proyectos Open Source.
- **Soporte Multiprotocolo**
- La nueva versión tiene la infraestructura necesaria para servir distintos protocolos. Por ejemplo, se ha escrito el módulo mod_echo.
- **Soporte mejorado para las plataformas que no son tipo Unix**
- La versión 2.0 de Apache es más rápida y más estable en sistemas que no son tipo Unix, tales como BeOS, OS/2 y Windows, que la versión antigua. Con la introducción de módulos de

multiprocesamiento (MPMs) específicos para cada plataforma y del Apache Portable Runtime (APR), estas plataformas tienen ahora implementada su propia API nativa, evitando las capas de emulación POSIX que provocan problemas y un bajo rendimiento.

- **Nueva interfaz de programación (API) de Apache**

- La API para los módulos ha cambiado significativamente en la nueva versión. Muchos de los problemas de ordenación y prioridad de módulos de la versión 1.3 deben haber desaparecido. Apache 2.0 hace automáticamente mucho de lo que es necesario, y la ordenación de módulos se hace ahora por hooks, lo que ofrece una mayor flexibilidad. También se han añadido nuevas llamadas que ofrecen capacidades adicionales sin tener que parchear el núcleo del servidor Apache.

- **Soporte de Ipv6**

- En los sistemas que soportan Ipv6 con la librería Apache Portable Runtime, Apache soporta Ipv6 listening sockets por defecto. Además, las directivas Listen, NameVirtualHost, y VirtualHost soportan direcciones Ipv6 numéricas (por ejemplo, "Listen [2001:db8:1]:8080").

- **Filtros**

- Los módulos de Apache pueden ahora escribirse para que se comporten como filtros que actúan sobre el flujo de contenidos tal y como salen del servidor o tal y como son recibidos por el servidor. Esto permite, por ejemplo, que el resultado de un script CGI sea analizado por las directivas Server Side Include usando el filtro INCLUDES del módulo mod_include. El módulo mod_ext_filter permite que programas externos actúen como filtros casi del mismo modo que los CGIs pueden actuar como handlers.

- **Mensajes de error en diferentes idiomas**

- Los mensajes de error que se envían a los navegadores están ahora disponibles en diferentes idiomas, usando documentos SSI. Estos mensajes pueden personalizarse por el administrador del sitio Web para conseguir un look and feel coherente con el resto de los contenidos.

- **Configuración simplificada**

- Muchas directivas que podían inducir a confusión han sido simplificadas. Las directivas Port y BindAddress han desaparecido; para configurar la dirección IP en la que escucha el servidor

ahora se usa únicamente la directiva Listen; la directiva ServerName especifica el nombre del servidor y el número del puerto solo para redirecionamiento y reconocimento de host virtual.

- **Actualización de la librería de expresiones regulares (regular expresión)**
- Apache 2.0 incluye la Librería de expresiones regulares compatibles de/con Perl (PCRE). Ahora, cuando se evalúan las expresiones tipo, se usa siempre la potente sintaxis de Perl 5.[21]

1.11.Otras herramientas a utilizar.

1.11.1 Macromedia Dreamweaver 8 es un *software* fácil de usar que permite crear páginas web profesionales. [22]

Las funciones de edición visual de *Dreamweaver8* permiten agregar rápidamente diseño y funcionalidad a las páginas, sin la necesidad de programar manualmente el código HTML. [22]

Se puede crear tablas, editar marcos, trabajar con capas, insertar comportamientos Java Script, etc., de una forma muy sencilla y visual. [22]

Además incluye un software de cliente FTP completo, permitiendo entre otras cosas trabajar con mapas visuales de los sitios web, actualizando el sitio web en el servidor sin salir del programa. [22]
Dreamweaver es compatible con todas las principales tecnologías de servidor como, por ejemplo, ColdFusion, PHP, ASP, ASP.NET y JSP. [23]

1.11.2 NuSphere PHPEd.

NuSphere PHPEd es un medio para el desarrollo de carácter profesional, creado precisamente para la confección de bases de datos y aplicaciones Web por medio de lenguaje script, aunque también son soportados los lenguajes populares HTML, CSS, XML, Java, Python y Perl.[24]

NuSphere PHPEd es un muy fuerte editor de código, un sorprendente eliminador de errores PHP, creador y publicador de perfiles. También incluye su propia base de datos, clientes CVS, servicios SOAP, validación HTML, formato de códigos, y soportes varios [24].

Características:

- ✓ Completo sistema de ayuda.
- ✓ Plantillas de documento y de fragmentos de código frecuentes.

- ✓ Código de colores para comandos en PHP, Perl, Java script, SQL, HTML y más.
- ✓ Incluye cliente FTP y servidor Web.

1.12.Conclusiones del capítulo.

El producto de software encontrado como parte de la investigación en curso, no satisface todas las actividades que se realizan en la residencia estudiantil de la facultad, pues no se gestionan las cuartelarías, ni reportes de los estudiantes, que son procesos de suma importancia en esta área.

A partir del estudio detallado de este sistema y al analizar que todos los procesos que posee implementado los realiza con buena calidad y rapidez a la hora de procesar la información, surge la necesidad de mejorar el módulo de residencia perteneciente al SGIF 8, para que se adapte más a las necesidades de nuestra residencia estudiantil.

Para efectuar el diseño de dicho sistema, nos amparamos del Proceso Unificado Racional que hace uso de el lenguaje unificado de modelado UML, que en su conjunto conforman en la actualidad una de las metodologías más utilizadas en el desarrollo de grandes proyectos.

De las tecnologías, técnicas y metodologías que han sido objeto de estudio en este capítulo se seleccionó un grupo de ellas para conformar la propuesta tecnológica y así desarrollar un sistema que permita a los clientes finales un entorno de trabajo amigable y flexible.

- Lenguaje de programación: PHP 5, apoyándose del editor de código NuSphere PHPEd.
- Sistema gestor de base de datos: PostgreSQL (versión 8.2).
- Herramienta Diseño Web: Macromedia Dreamweaver 8.
- Frameworks: CodeIgniter.
- Herramienta Case: Rational Rose.
- Metodología RUP.
- Lenguaje UML.
- Servidor Apache 2.0

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1.Introducción.

En este Capítulo se verá una descripción del negocio del problema a resolver, así como sus distintos casos de uso del negocio y la descripción de cada uno, además de las actividades a seguir por cada actor y trabajador de dicho negocio. Se podrá ver además los diversos casos de uso del sistema y los requerimientos funcionales y no funcionales con los cuales se realizará posteriormente un análisis y diseño para concluir con la implementación, prueba y presentación de la primera versión de este sistema a automatizar, pero esto se conocerá en próximos Capítulos. Teniendo en cuenta la metodología RUP se comienza este trabajo de modelación del negocio en la fase de inicio.

2.2.Objeto de Estudio.

Se necesita crear un sistema para un mejor funcionamiento de las actividades de la residencia y que de solución a los problemas existentes.
El objetivo del sistema es automatizar los procesos de la residencia. En la residencia se realizan varias actividades, que cada una tiene una distribución diferente, y que todas son planificadas por el Vicedecano, es por eso que se requiere de un sistema que automatice y de soporte a cada una de ellas, entre ellas están el Plan de Guardia y el de la Cuartelería, además de la Distribución de la Beca. En la actualidad estas actividades se realizan de forma manual.

2.3.Objeto de automatización.

Serán objeto de automatización los procesos de gestión de cada una de las actividades que se realizan en la residencia. A continuación se verán los procesos más importantes de la residencia que son objeto de automatización:

- Se debe realizar la Distribución de la Beca de manera que los estudiantes queden ubicados en los edificios de acuerdo al sexo.

- Se debe crear un Plan de Guardia Mensual que responda a las necesidades existentes, donde el grupo quede dividido en dos subgrupos, y que cada subgrupo esté dirigido por un profesor, el primer subgrupo debe ser dirigido por el profesor guía.

- Se debe crear un Plan de Cuartelería Mensual donde se planifique el día en que cada estudiante tiene que realizar la misma en su paso de escalera.

- El sistema debe ser capaz de crear reportes de cualquier estudiante ubicado en la residencia y mostrar las aptitudes que posee.

2.4.Información que se maneja.

La información que se maneja en este sistema es bien amplia pero de fácil acceso, pues es necesario conocer los datos completos de los estudiantes de la facultad 2 a ubicar en la residencia; la cantidad de edificios que posee la residencia con sus apartamentos y la capacidad de estos, además de la cantidad de pasos de escaleras; los profesores que dan clases a los grupos de la facultad 2 y resaltar los que sean guías; se necesita conocer las aptitudes que poseen los estudiantes para poder realizar un reporte completo de estos. Con todos estos datos y con el conocimiento de cómo se realizan las actividades a automatizar se lleva a cabo la automatización del sistema.

2.5.Propuesta de sistema.

La propuesta que se hace para dar solución al problema en cuestión es confeccionar una aplicación Web que sea de fácil acceso pero con una fuerte seguridad e integridad.

El sistema debe cumplir con la exigencia de que el cliente es el único encargado de navegar en la aplicación y es por ende, la única persona autorizada a administrar las planificaciones y los cambios que sean necesarios realizar. El sistema debe ser explícito en la información que se muestre, por esto es la importancia de una buena interfaz de usuario con una clara descripción de lo que ahí se muestra. La aplicación debe centrarse en las funcionalidades que se requieren y que estas se ejecuten con la mayor rapidez posible. El sistema debe permitir insertar, modificar y actualizar, así como buscar y guardar toda la información que se maneja. El sistema debe ser específico y mostrar sólo lo que el usuario desee ver.

Este sistema cumple con cada requisito establecido por el cliente, tiene mayor interacción con el usuario, mayor seguridad en el flujo de información, es más atractivo a la vista del usuario y presenta una mejor organización en cada opción que presenta el sistema, facilitando el trabajo del Vicedecano para el buen funcionamiento de la residencia estudiantil.

2.6.Modelo de Negocio.

Descripción del Negocio:

En la actualidad el trabajo del Vicedecano de residencia es muy complejo debido a que la planificación de los procesos que se realizan como la Guardia y la Cuartelería, además de la Distribución de la Beca se realizan de forma manual.

En cuanto a la distribución de la Beca el tiene que conocer la cantidad de edificios que posee su residencia, conocer la cantidad de apartamentos y saber la capacidad de cada uno de ellos, separar los grupos por sexo, y repartir los edificios en hembras y varones, quedando confeccionada la distribución de esta forma.

Para la planificación de la Guardia divide los grupos en dos subgrupos, al primer subgrupo lo ubica con el profesor guía y el segundo subgrupo lo dirige un profesor que le imparte docencia al grupo, tiene que tener profesores de reserva por si se presenta cualquier inconveniente, tiene que mandar la planificación en tiempo y forma para que sea del conocimiento de profesores y estudiantes y la actividad se realice de forma satisfactoria, quedando confeccionado el Plan de Guardia.

Para la planificación de la Cuartelería cuenta con el listado de estudiantes por edificio y los planifica de acuerdo al paso de escalera, comenzando por el primer apartamento y con la condición de que los domingos no se realiza esta actividad, quedando confeccionado el Plan de Cuarteleria.

Actores y Trabajadores del Negocio.

Tabla 2.6.1-*Actores Del Negocio.*

Actores del negocio	Justificación
Vicedecano	Interviene en todas las actividades del negocio, es el máximo responsable de que se lleven a cabo.

Tabla 2.6.2-*Trabajadores Del Negocio*

Trabajadores del negocio	Justificación
Planificador	Interviene en todos los procesos del negocio, es el encargado de planificarlos.

Diagramas de Casos de Uso del Negocio.

Tabla 2.6.3- Diagrama de Caso de Uso del Negocio: Distribuir Beca

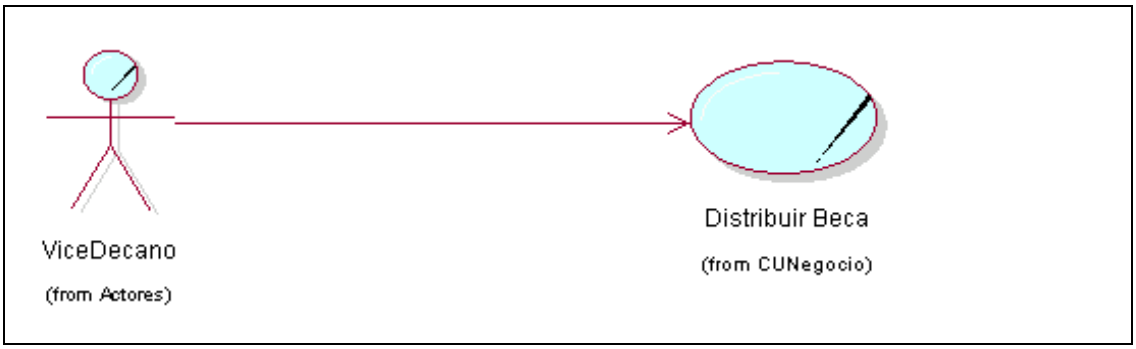


Tabla 2.6.4- Diagrama de Caso de Uso del Negocio: Distribuir Cuartelería

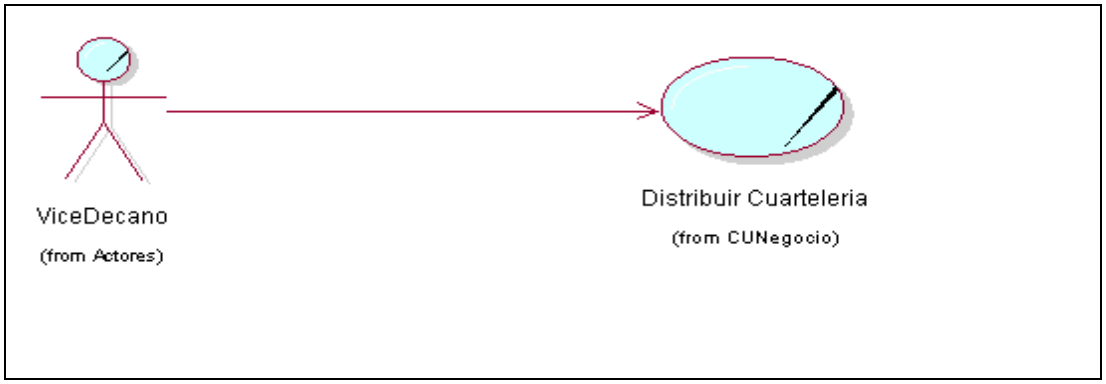


Tabla 2.6.5- Diagrama de Caso de Uso del Negocio: Distribuir Guardia

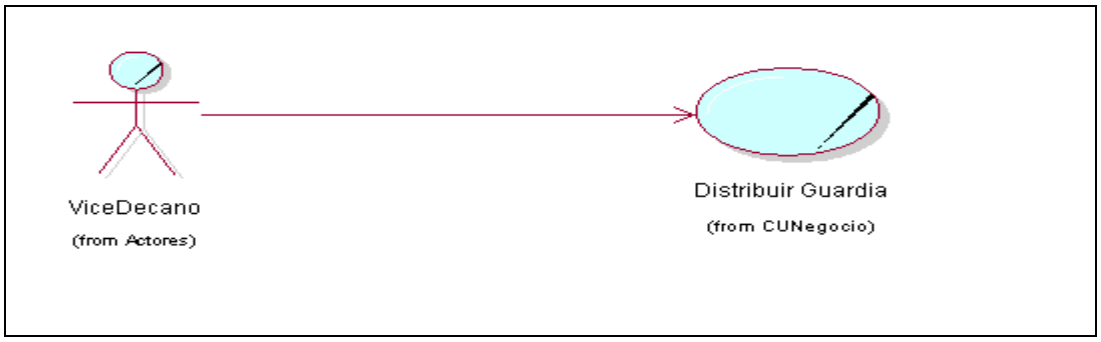
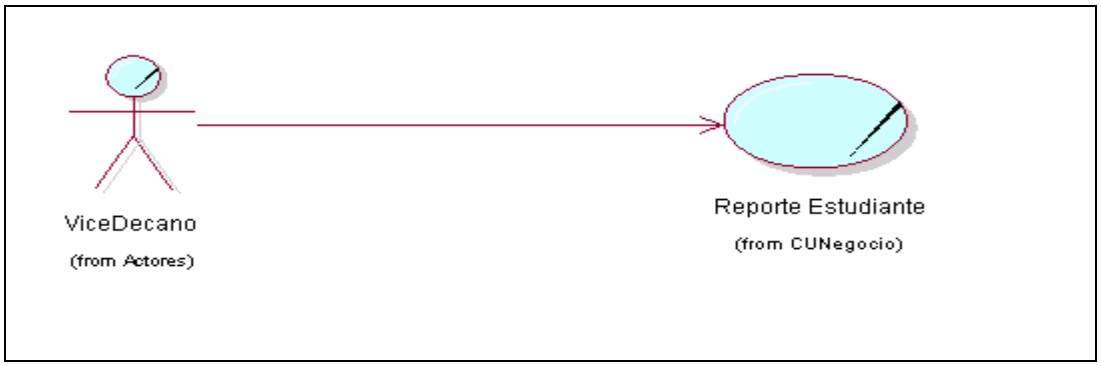


Tabla 2.6.6- Diagrama de Caso de Uso del Negocio: Reporte Estudiante



Descripción de los Casos de Usos del Negocio y Reglas del Negocio. Ver Anexo 1.

Diagrama de Actividades. Ver Anexo 2.

Especificación de los Requisitos de Software.

Entre los **requerimientos funcionales** del sistema tenemos:

R1: Autenticar Usuario (Crítico)

1.1Nombre Usuario

1.2 Contraseña

R2: Importar

2.1 Importar Edificios

2.2 Importar Estudiantes

2.3 Importar Profesores

R3: Gestionar Beca

3.1 Distribuir Beca

3.2 Modificar asignación de la Beca.

3.2.1 Eliminar Asignacion de Beca.

3.3 Mostrar Estado de la Beca

3.4-Gestionar Edificio

3.4.1-Gestionar Apartamento

3.4.1.1 Mostrar Listado Apartamento por Edificio

3.4.1.2 Modificar Apartamento

R5: Gestionar Cuartelaría

5.1 Asignar Cuartelaría a un Estudiante.

5.2 Mostrar Listado de Cuarteleria por Edificio.

R6: Gestionar Guardia

6.1 Asignar Guardia

6.1.1 A un Grupo en una fecha determinada.

6.1.2 Determinar Profesor responsable de Guardia.

6.2 Mostrar Listado de asignación de Guardia.

R7: Gestionar Aptitud

7.1 Insertar Aptitud

7.2 Eliminar Aptitud

7.3 Asignar Aptitudes a un estudiante

R8: GestionarReporteEstudiantes

8.1 Generar reporte de los estudiantes con los siguientes datos: nombre , aptitudes, apartamento, teléfono, grupo, nro_solapín.

Como **Requerimientos no funcionales**, es decir, características o capacidades del sistema, tenemos los siguientes:

Requerimientos de apariencia o interfaz externa

Interfaz de usuario amigable y fácil de entender sin mucho entrenamiento por parte del cliente. Colores suaves e información legible que evite cualquier pérdida por parte de un usuario sin conocimiento de la aplicación.

Requerimientos de Rendimiento

Sistema rápido y eficiente en la búsqueda de datos. Los envíos de reportes y de respuestas del sistema deben ser rápidos y exactos, sin perdida de datos.

Requerimientos de Soporte

Documentación de usuario y una Ayuda con suficiente información.

Presenta manual de usuario fácil de entender para todo tipo de usuario.

Requerimientos de Seguridad

Confidencialidad: Asignar permisos limitados a usuarios teniendo en cuenta el nivel de acceso a la aplicación y al trabajo a realizar en ella.

Integridad: La información manejada por el sistema será objeto de cuidadosa protección contra la corrupción y estados inconsistentes.

Disponibilidad: Significa que los usuarios autorizados se les garantizará el acceso a la información y que los dispositivos o mecanismos utilizados para lograr la seguridad no ocultarán o retrasarán a los usuarios para obtener los datos deseados en un momento dado.

Requerimientos de confiabilidad

Presenta protección contra fallos. El uso del sistema por usuarios autorizados limita el acceso de personal ajeno y protege la información de agentes foráneos o accidentes provocados intencionalmente o por el más uso del sistema.

Requerimientos de Software.

Del lado del Servidor

Servidor Apache 2.0

Gestor de Base Datos Postgree 8.2

PHP 5

Del lado del Cliente

Sistema Operativo Windows XP o Superior

Internet Explorer 6.0

Mozilla Firefox 2.0 o Superior

Definición de los Casos de Uso del Sistema.

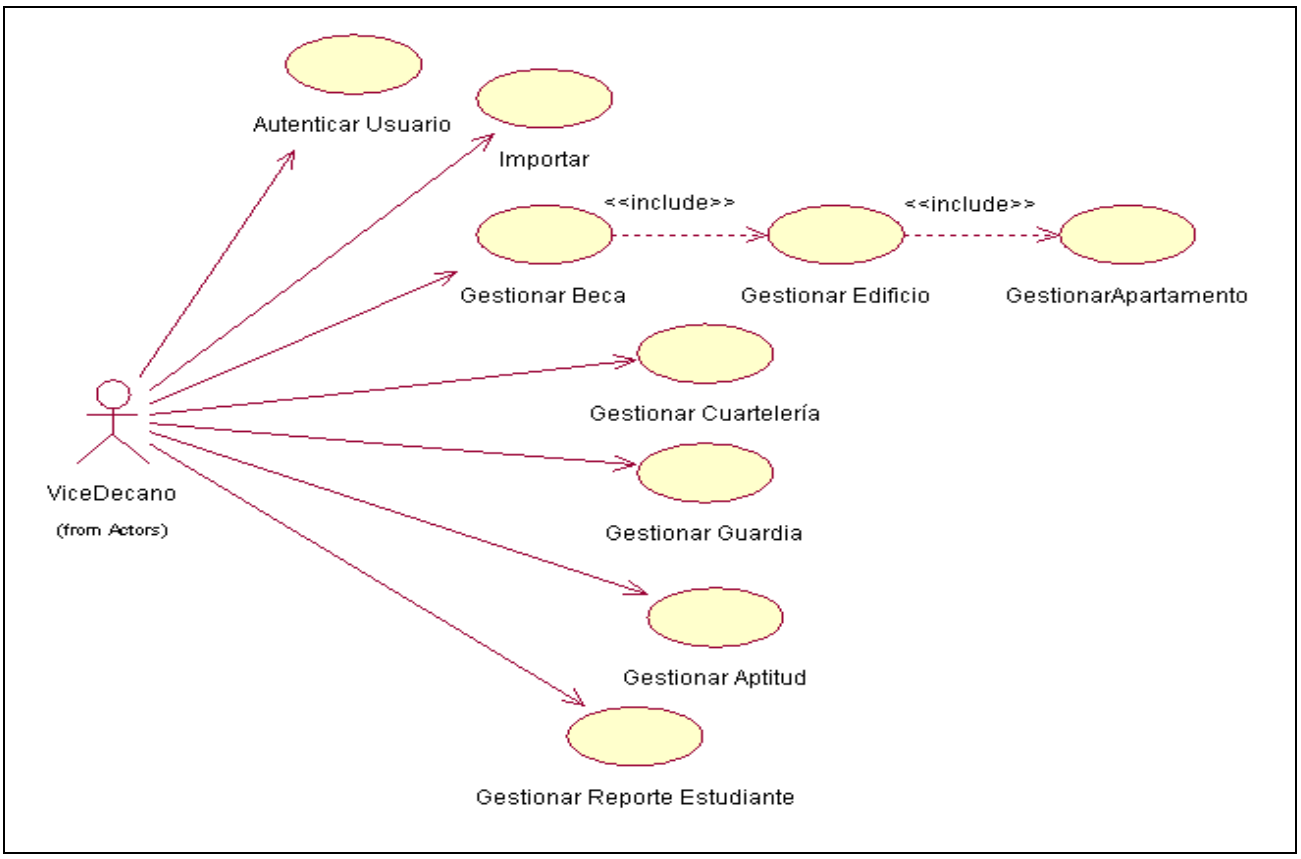
Actores del sistema.

Tabla 2.6.7- *Actor del sistema.*

Actores	Justificación
ViceDecano	Persona con acceso a interactuar con todas las funcionalidades del sistema.

Diagrama de Casos de Uso del Sistema.

Tabla 2.6.8- Diagrama de Casos de Uso del Sistema



Descripción de los Casos de Uso del Sistema. Ver Anexo 3.

Conclusiones

En este capítulo se ha planteado la necesidad de describir las características del sistema y se ha realizado un análisis de los requerimientos funcionales y no funcionales del sistema, facilitando la identificación de los casos de uso del sistema.

CAPÍTULO 3: ANALISIS Y DISEÑO DEL SISTEMA

3.1.Introducción

En este capitulo abordaremos el tema relacionado con el análisis y diseño del sistema, donde realizaremos los distintos tipos de diagramas de clases que se se necesitan para dar solución al problema y explicaremos los tipos de clases que los componen. Se profundiza en los requerimientos funcionales que debe cumplir el sistema.

3.2.Análisis.

Clases de análisis: Se centran en los requisitos funcionales y son evidentes en el dominio del problema porque representan conceptos y relaciones del dominio.

Clases Interfaz: Se utiliza para modelar la interacción de entre los actores y el sistema. permitiendo recibir o presentar información hacia los actores. Representa abstracciones de ventanas, formularios, paneles etc.

Clases de control: Las clases de control coordinan el trabajo de uno o unos pocos casos de uso, coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso, por lo que definen el flujo de contol y las transacciones dentro de un caso de uso delegando el trabajo a otros objetos.

Clases de entidad: Estas son las clases encargadas de modelar la información que posee larga vida y que es a menudo persistente.

3.2.1.Diagramas de clases de análisis.

Los diagramas de clases del análisis : Estos muestran las interaciones de estos tipos de clases para cada caso de uso, permitiendo una mejor comprensión. Un Diagrama de clases del análisis es un artefacto en el que se representan los conceptos en un dominio del problema. Representa las cosas del mundo real, no de la implementación automatizada de estas cosas

Tabla 3.2.1.1-Autenticar Usuario

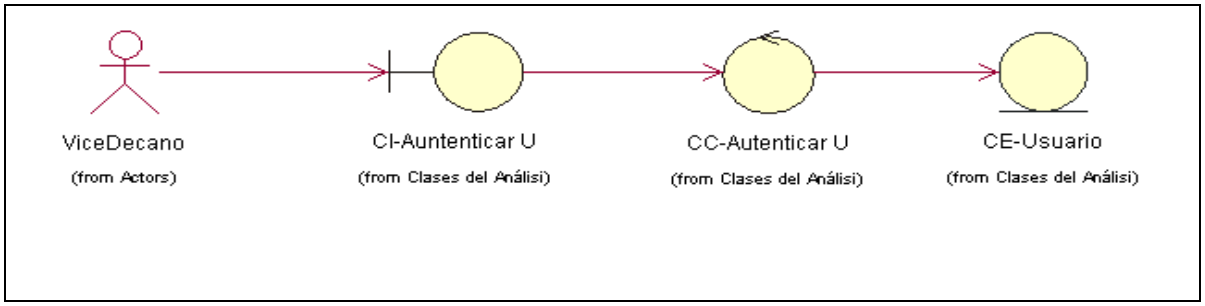


Tabla 3.2.1.2-GestionarBeca(Modificación)

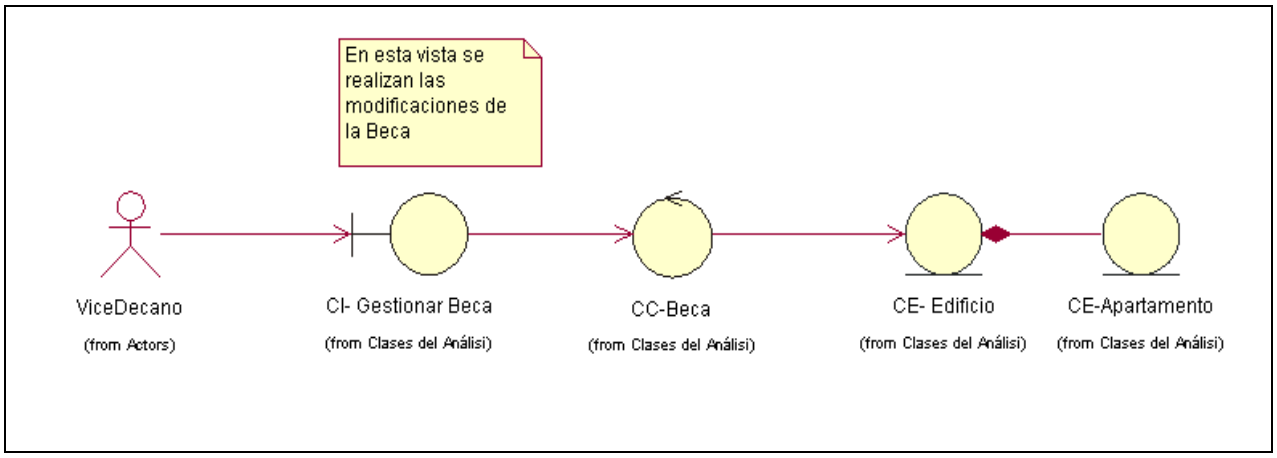


Tabla 3.2.1.3-Gestionar Beca

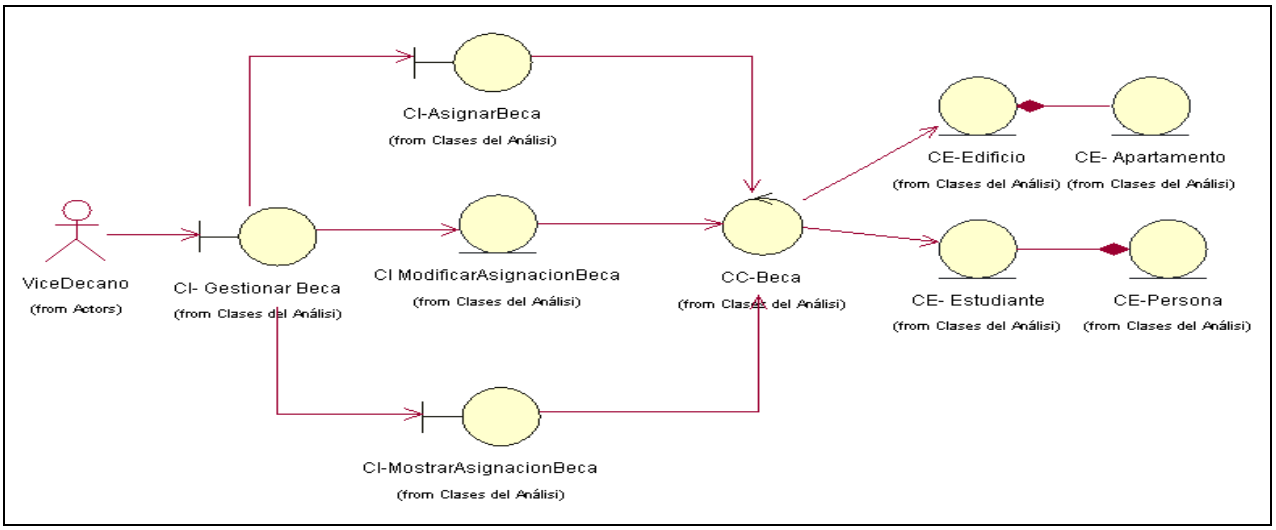


Tabla 3.2.1.4-Gestionar Cuartelería

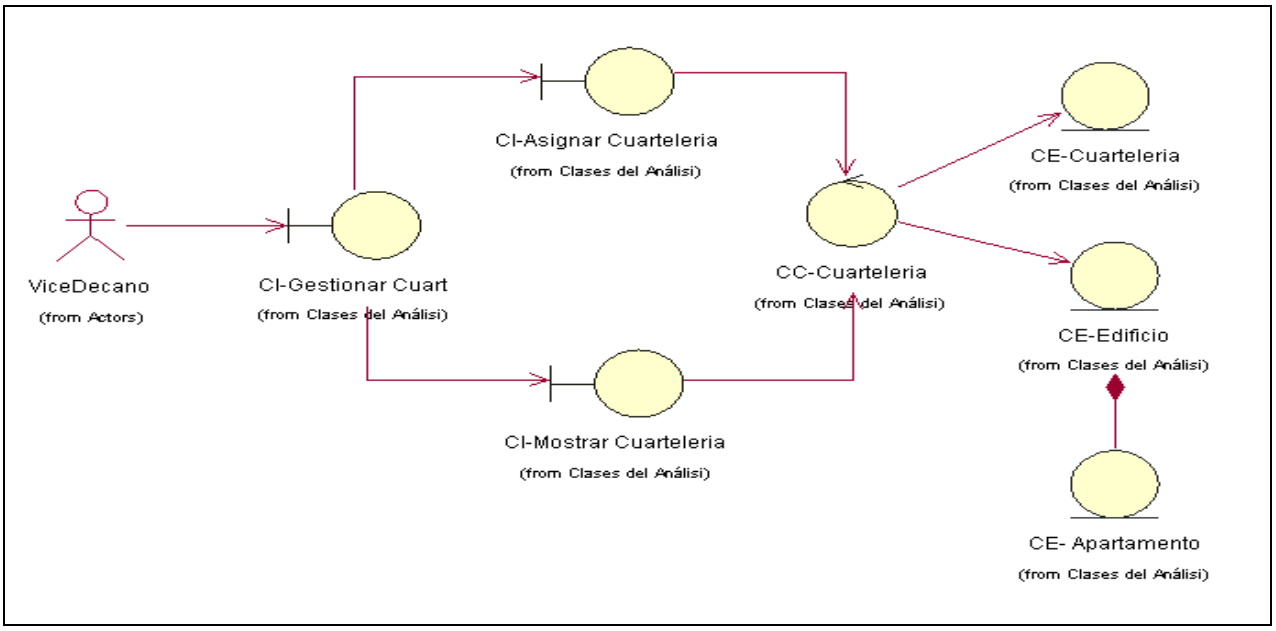


Tabla 3.2.1.5-Gestionar Guardia

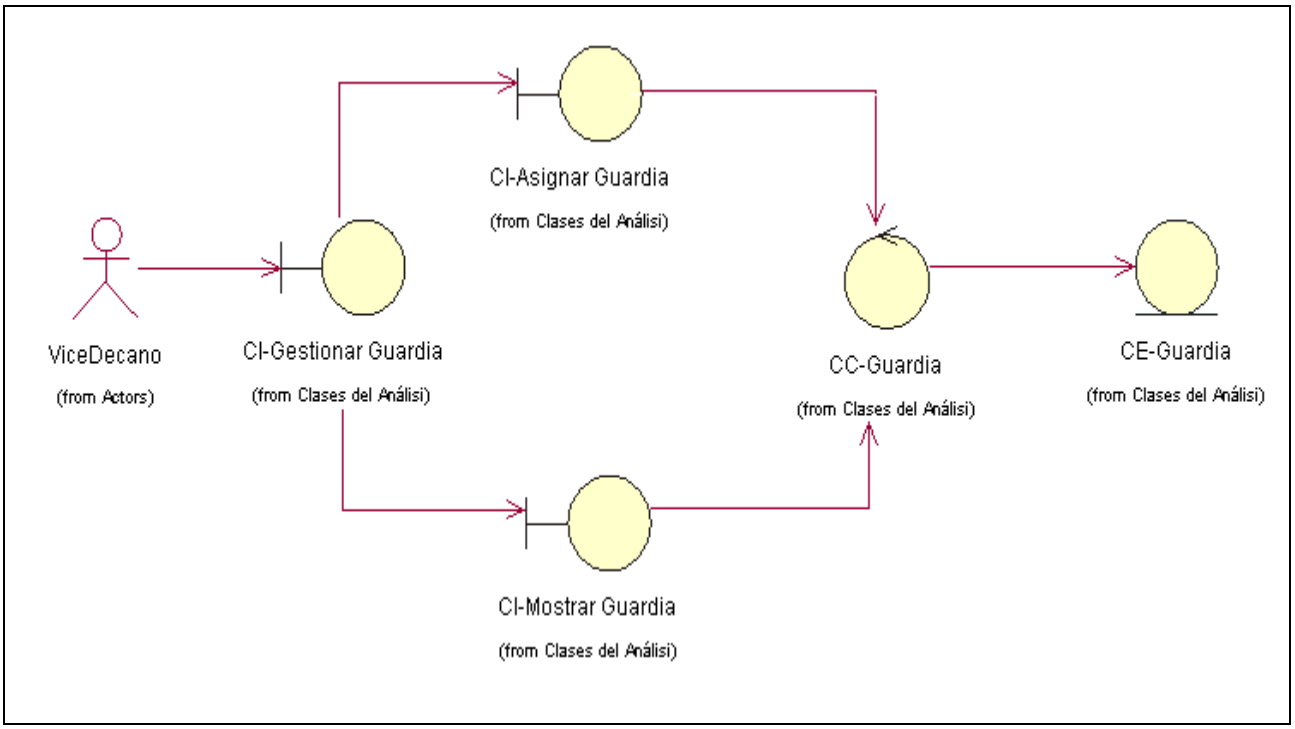


Tabla 3.2.1.6-GestionarReporteEstudiante

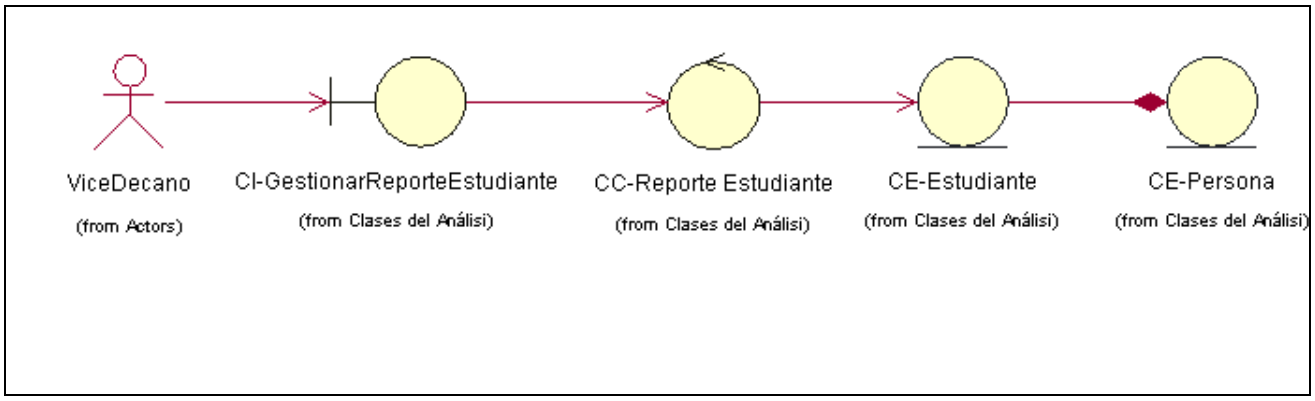
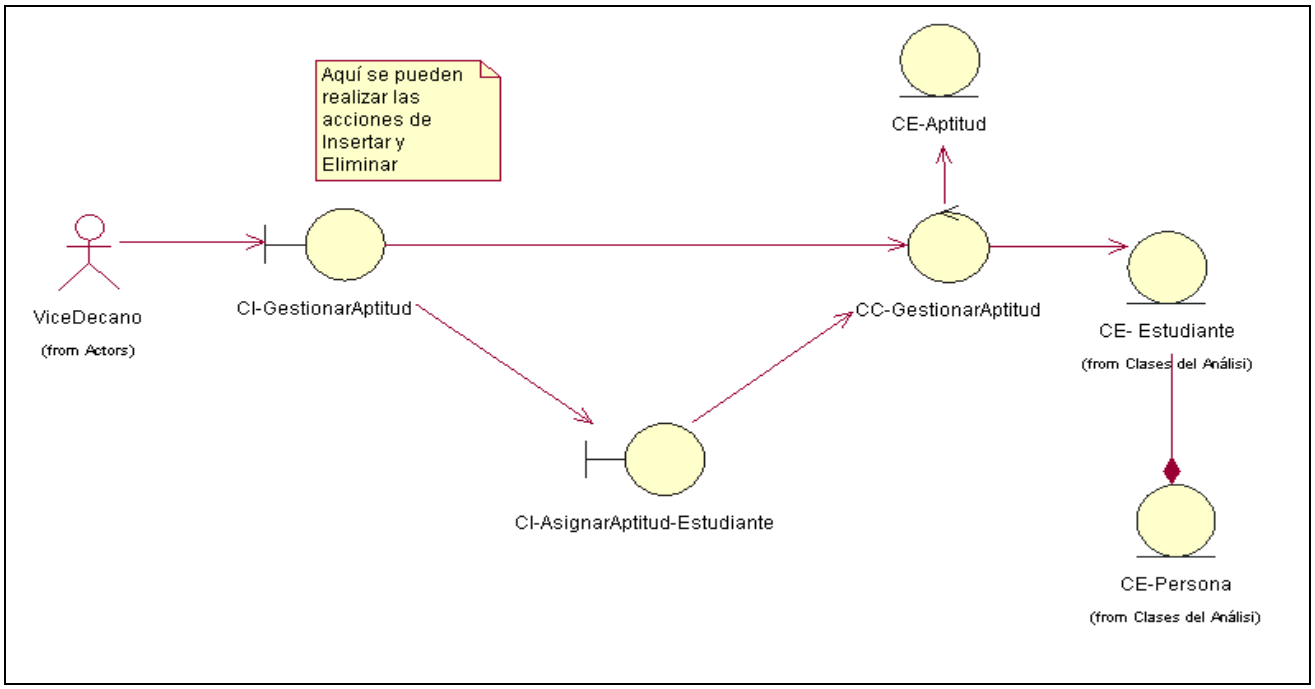


Tabla 3.2.1.7-GestionarAptitud



3.3.Diseño.

El diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción. En el diseño modelamos el sistema y encontramos su forma(incluida la

arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. El diseño impone una estructura del sistema que debemos mantener lo mas fielmente posible.

3.3.1.Diagrama de clases del diseño.

Diagrama de clases del diseño: En el diagrama de clases podemos ver las diferentes clases que estructuran el sistema asi como sus interacciones. En el diagrama se puede ver las colaboraciones que existen entre cada página donde cada una representa a una clase. Los diagramas de clase fueron definidos a partir de cada caso de uso.

Tabla 3.3.1.1-Autenticar Usuario

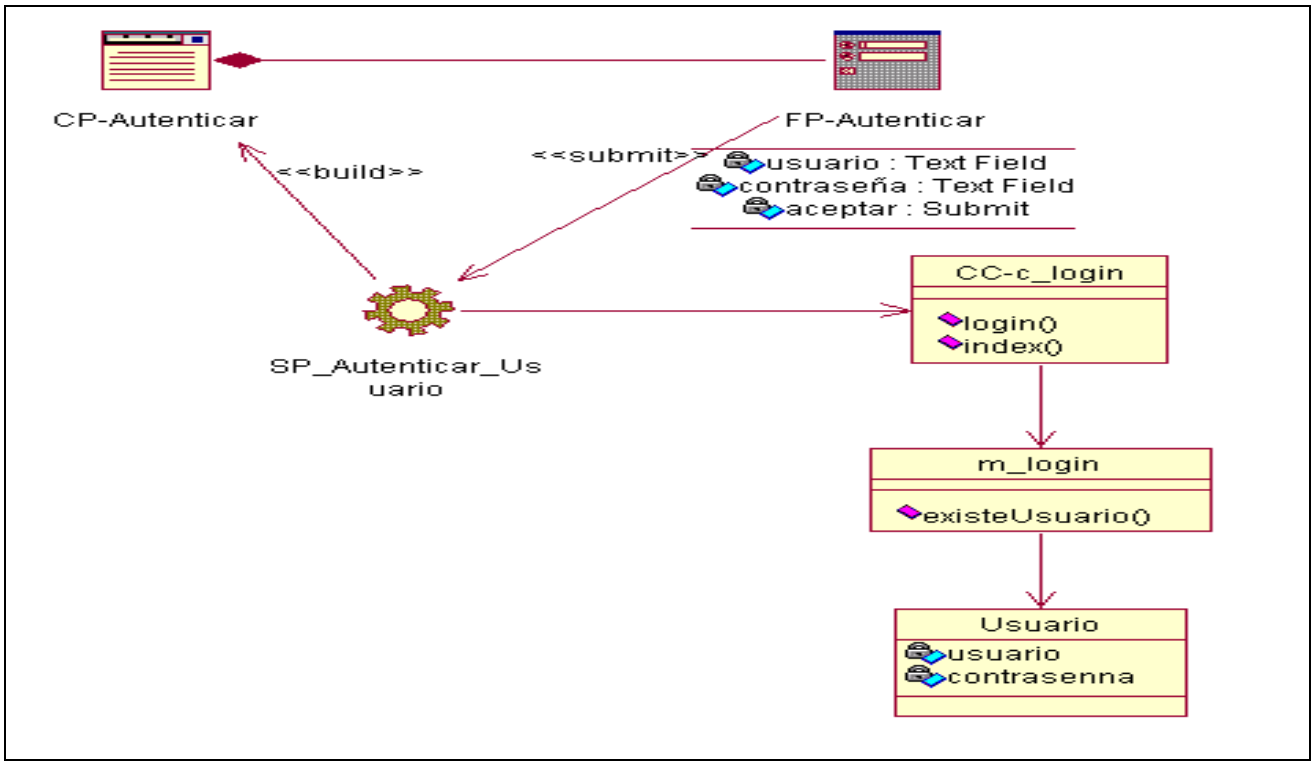


Tabla 3.3.1.2-GestionarBeca(Modificación)

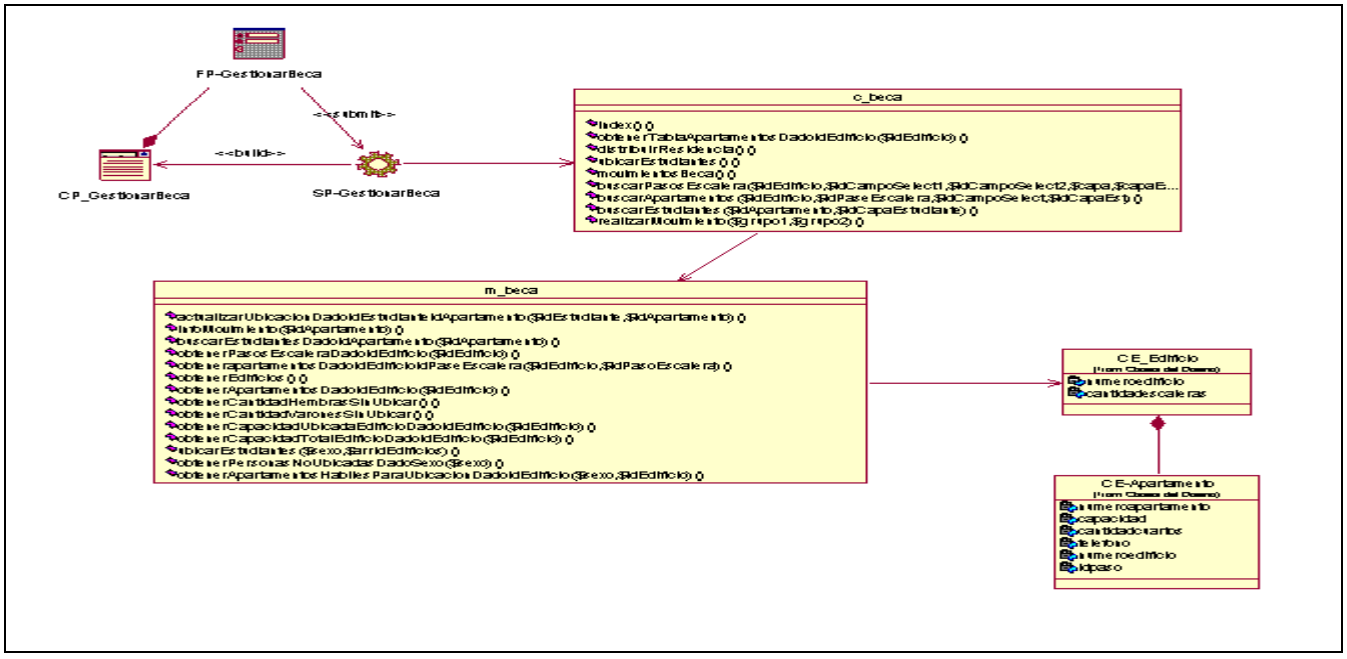


Tabla 3.3.1.3-GestionarBeca.

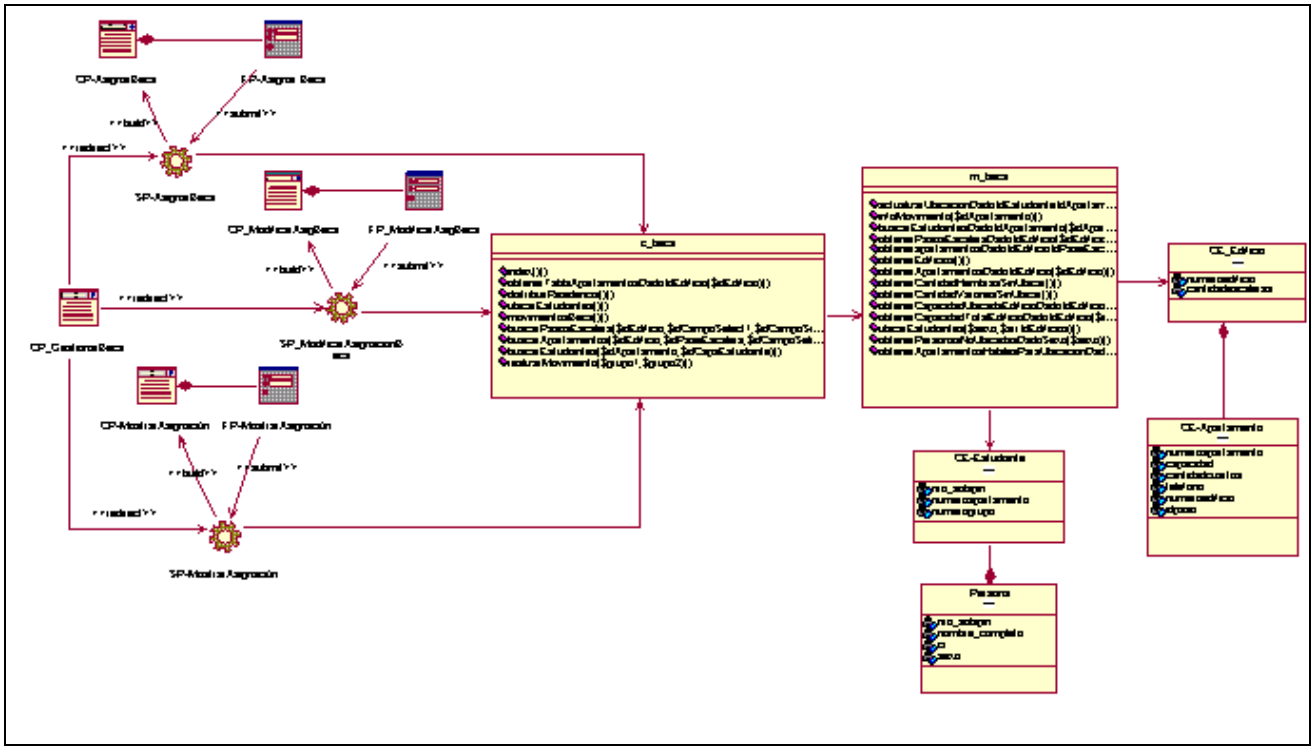


Tabla 3.3.1.4-GestionarCuartelería.

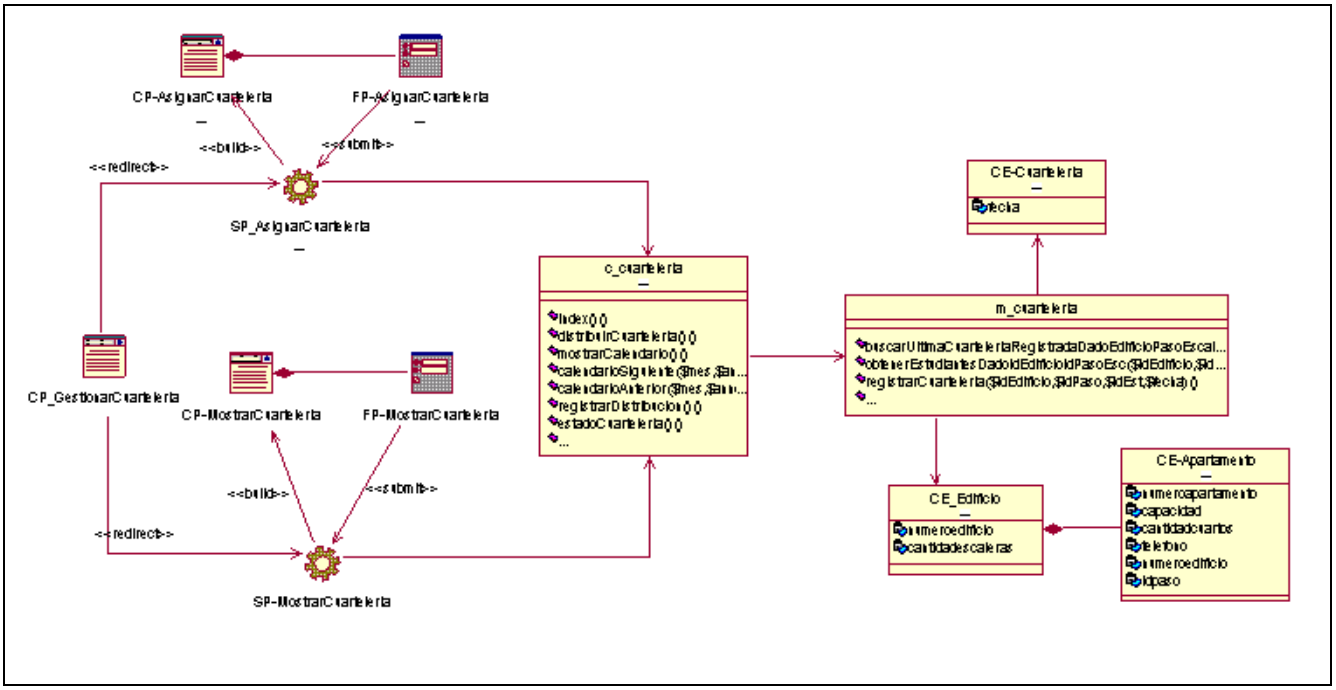


Tabla 3.3.1.5-GestionarGuardia.

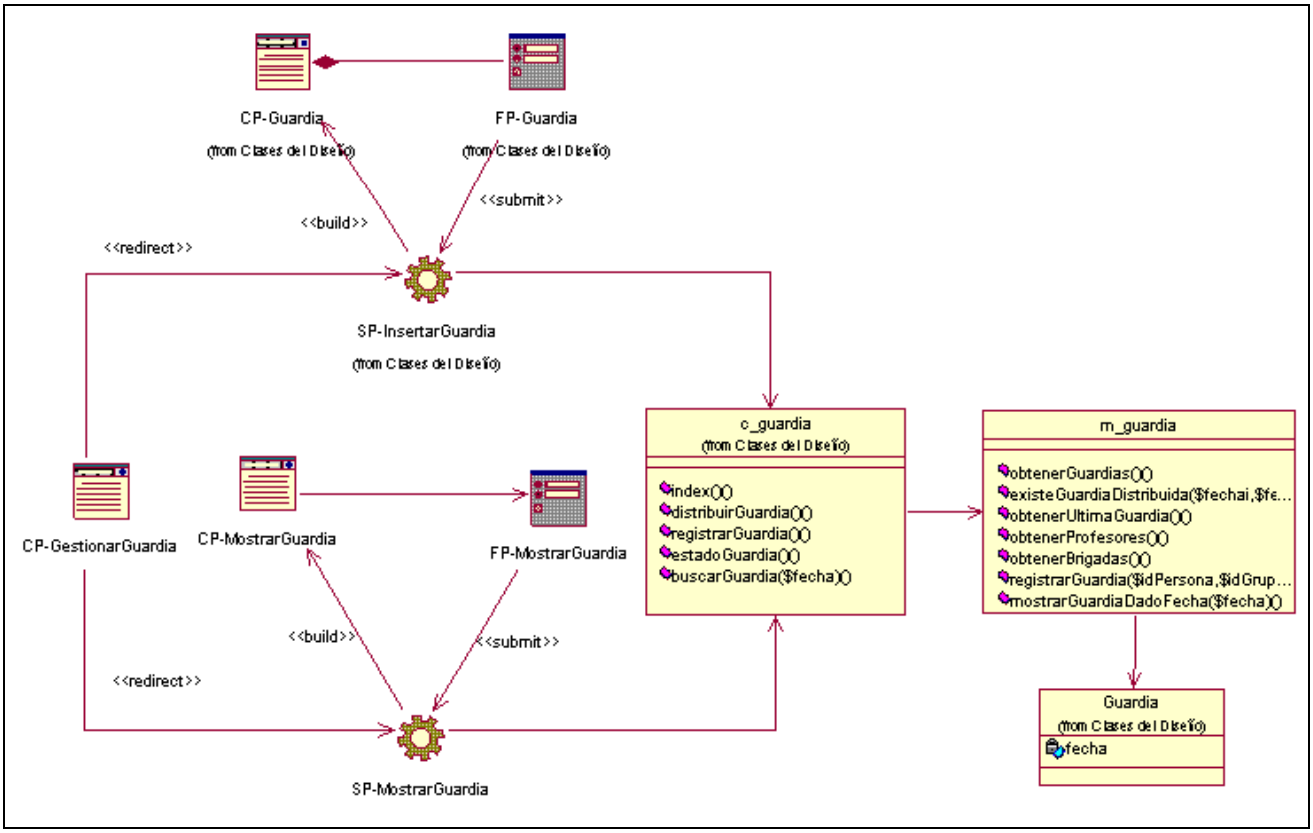
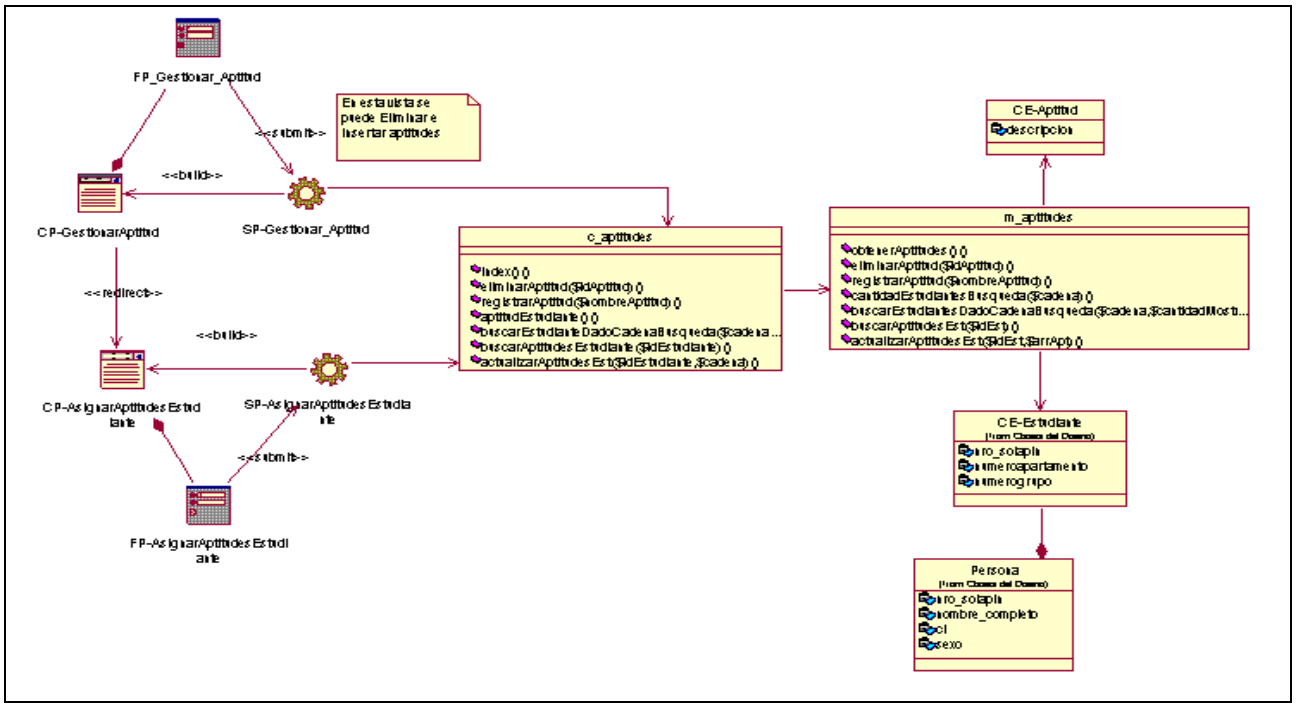
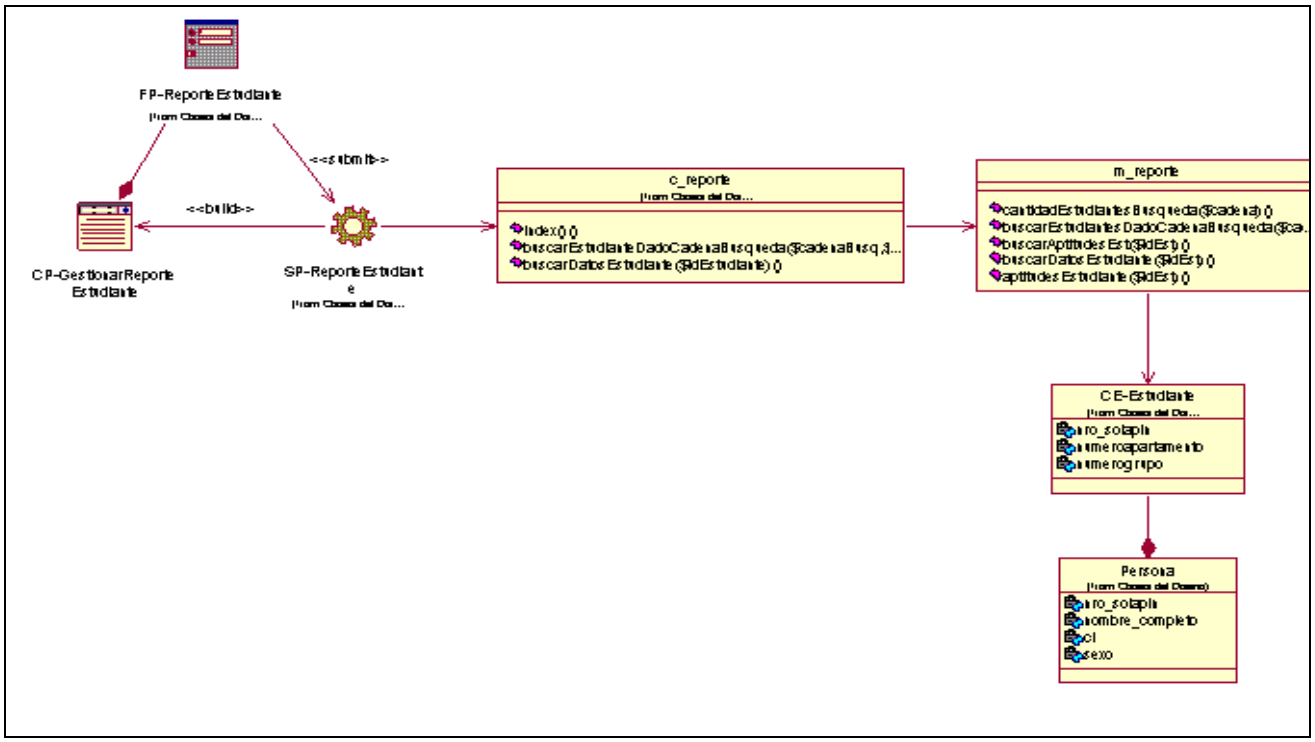


Tabla 3.3.1.6-GestionarAptitud



.

Tabla 3.3.1.7-GestionarReporteEstudiante



3.4.Diagramas de interacción. Poner de Anexo. VER ANEXO 4.

Diagramas de interacción:

Los diagramas de interacción muestran la secuencia de acciones de un caso de uso en específico, permiten modelar los aspectos dinámicos del sistema. Un diagrama de interacción consiste en un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos. Los tipos de diagramas de interacción son:

- **Diagramas de secuencia:** En estos se destaca el orden temporal de los mensajes como principal diferencia con respecto a los de colaboración

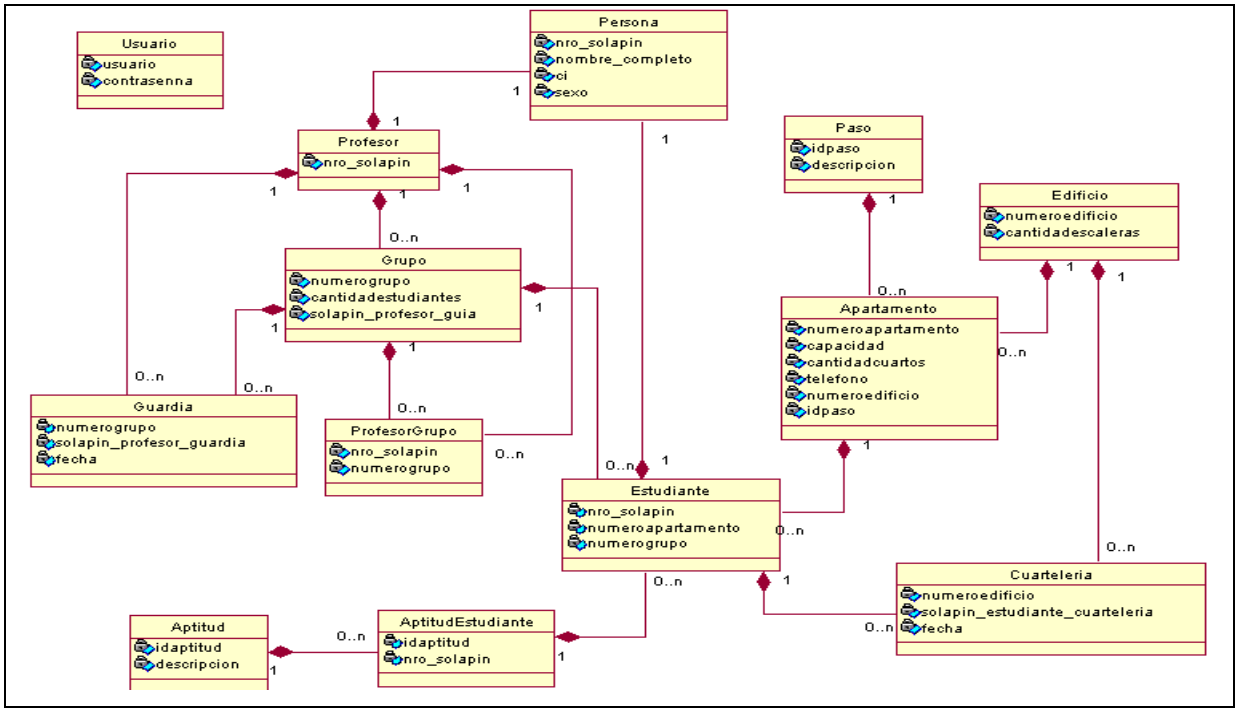
- **Diagramas de colaboración:** En estos se destaca la organización estructural de los objetos que envían y reciben mensajes.

3.5.Diseño de la base de datos.

Definición de Bases de Datos.- Un conjunto de información almacenada en memoria auxiliar que permite acceso directo y un conjunto de programas que manipulan esos datos

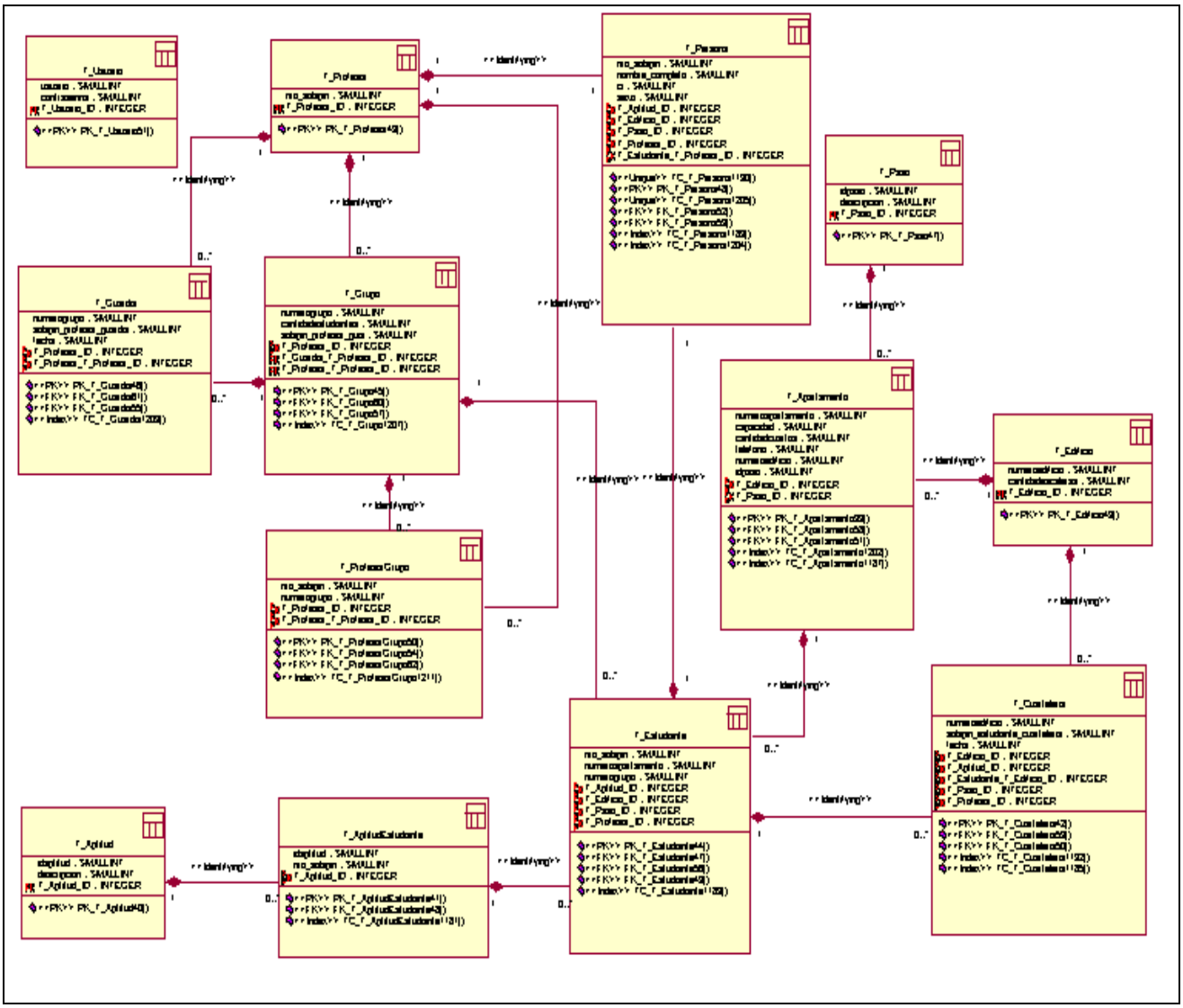
Base de Datos es un conjunto exhaustivo no redundante de datos estructurados organizados independientemente de su utilización y su implementación en máquina accesibles en tiempo real y compatibles con usuarios concurrentes con necesidad de información diferente y no predicable en tiempo.

Tabla 3.5.1-Modelo lógico de datos (diagrama de clases persistentes).



Modelo de datos: Un modelo de datos describe como se representan estos de una forma abstracta en un sistema gestor de base de datos. Estos contienen objetos, atributos y relaciones.

Tabla 3.5.2-Modelo físico de datos (modelo de datos).



Conclusiones

En este capítulo hemos abordado los diagramas del análisis y del diseño así como el modelo de datos, permitiendo una mejor comprensión de nuestro sistema.

CAPÍTULO 4: IMPLEMENTACION Y PRUEBA

4.1.Introducción

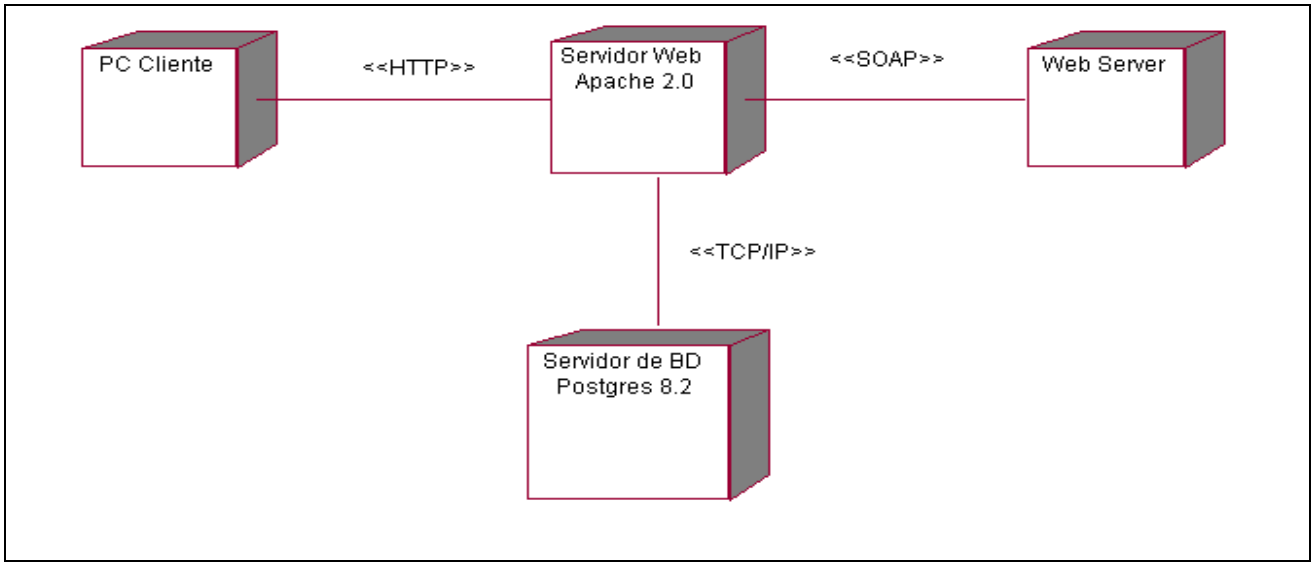
En este capitulo se conocerán los distintos diagramas de implementación que representan los elementos físicos del sistema y el despliegue realizado por el proyecto.

4.2.Diagrama de despliegue.

El modelo de despliegue muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los links de comunicación entre ellos, y las instancias de los componentes y objetos que residen en ellos. Se compone de los siguientes elementos:

- **Nodos:** Elementos de procesamiento con al menos un procesador, memoria, y posiblemente otros dispositivos
- **Dispositivos**: Nodos estereotipados sin capacidad de procesamiento en el nivel de abstracción que se modela
  - **Conectores**: Expresa el tipo de conector o protocolo utilizado entre el resto de los elementos del modelo.

Tabla 4.2.1-Diagrama de Despliegue



4.3.Diagrama de componentes.

El diagrama de componentes describe los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable

Tabla 4.3.1-Login(Autenticarse)

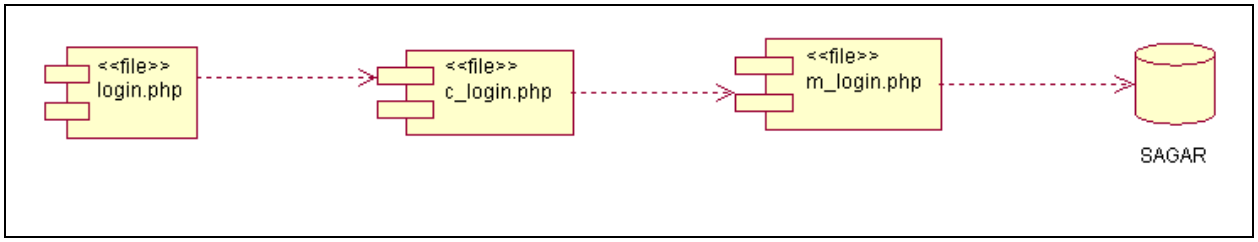


Tabla 4.3.2-Prototipo

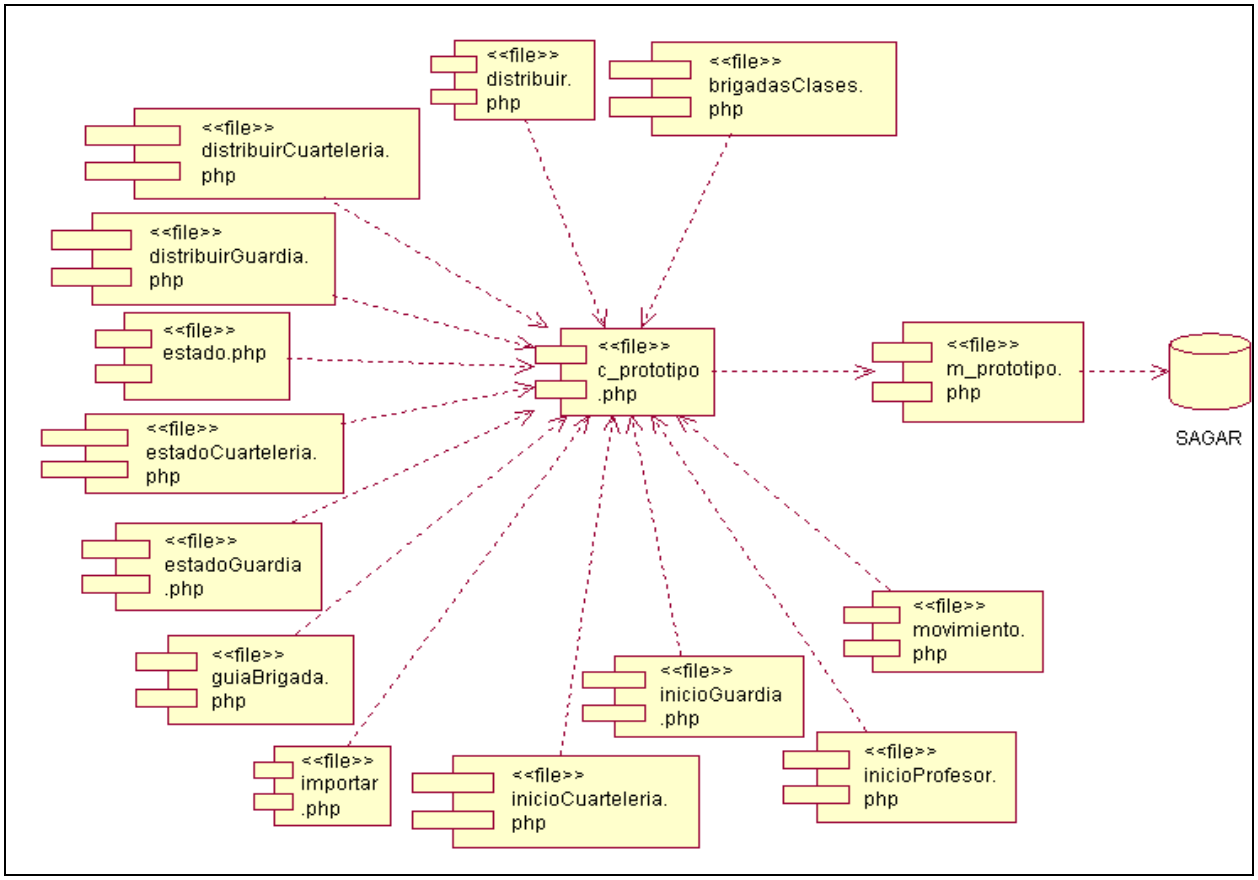


Tabla 4.3.3-Beca

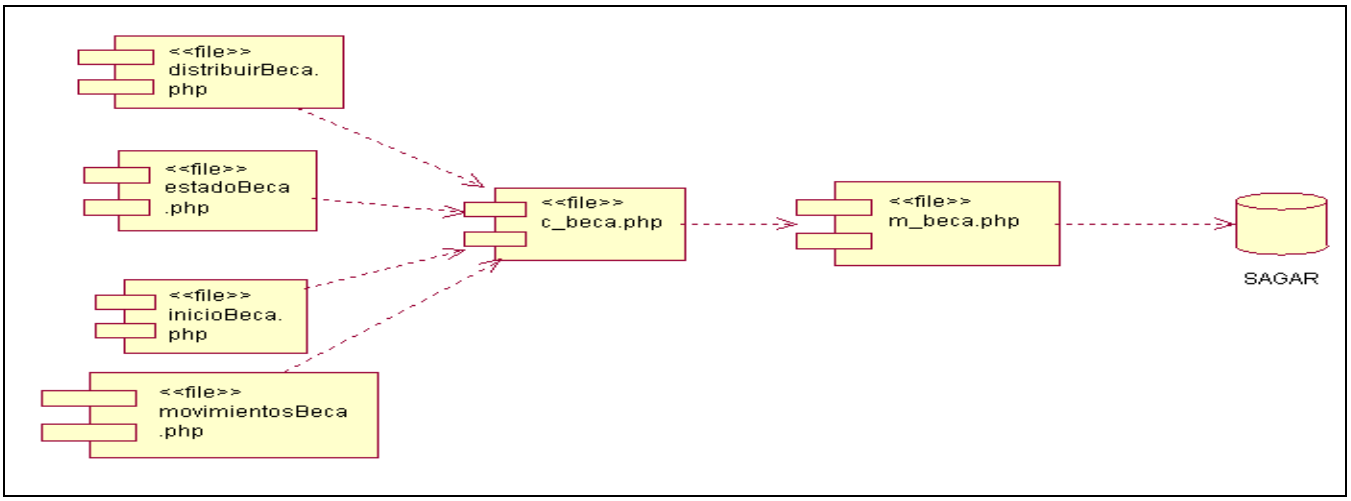


Tabla 4.3.4-Cuartelería

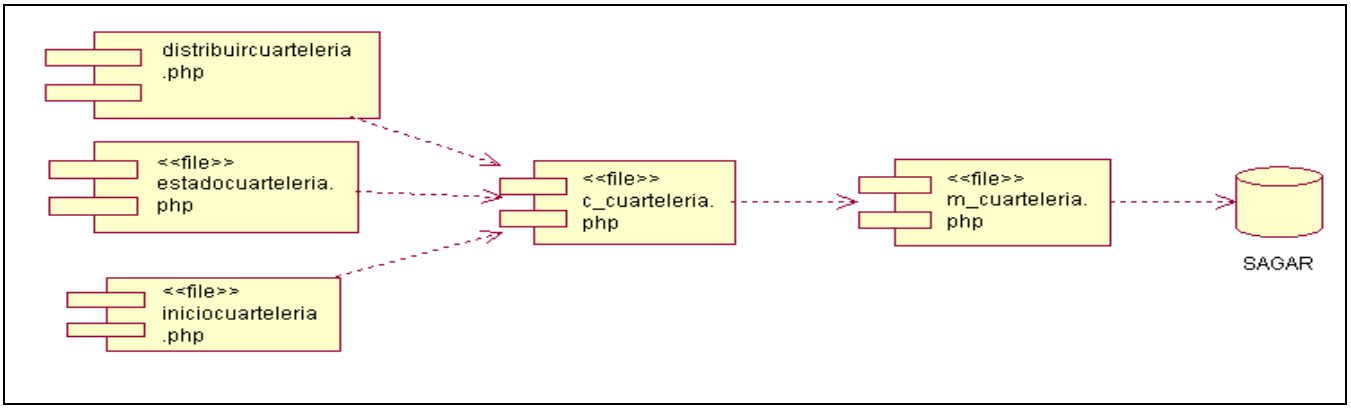


Tabla 4.3.5-Guardia

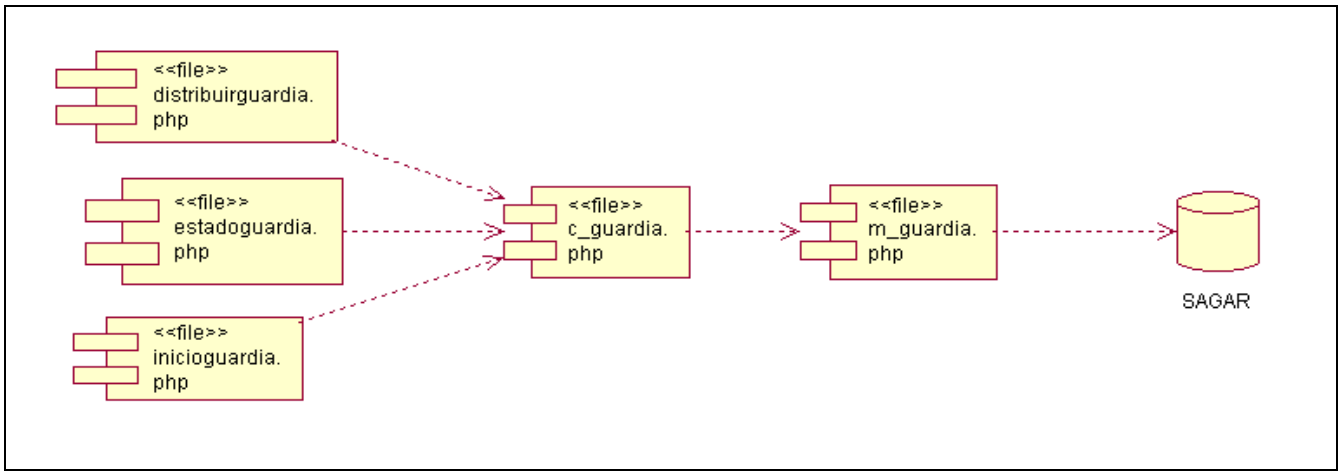


Tabla 4.3.6-Aptitud

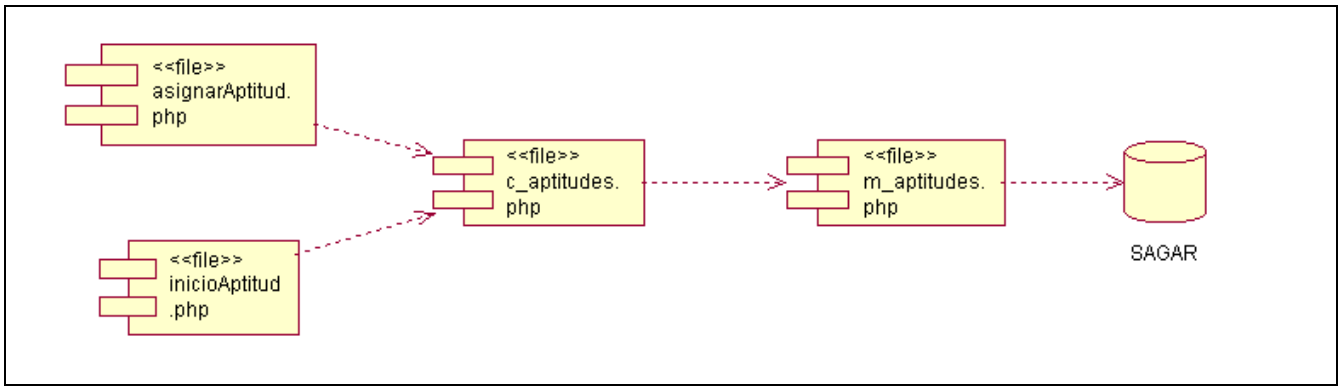
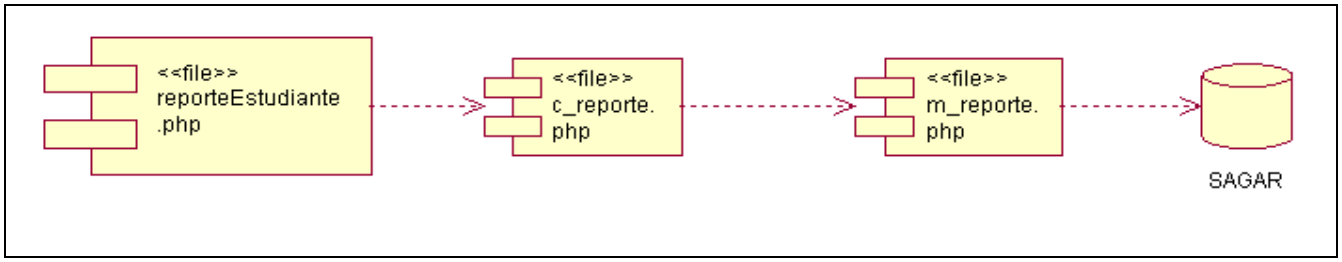


Tabla 4.3.6-Reporte



Conclusiones

En este capítulo se desarrollaron los diagramas de componentes del sistema, los cuales representan la separación de un sistema en componentes físicos. Además se desarrolló el diagrama de despliegue que muestra la configuración de los nodos de procesamiento.

CAPÍTULO 5: ESTUDIO DE FACTIBILIDAD

5.1.Introducción

Uno de los principales objetivos del estudio de la factibilidad es disminuir el riesgo que implica todo el proceso investigativo y evitar las pérdidas de recursos que afectan la entidad donde se lleva a cabo el proyecto en caso de que no se obtengan los beneficios esperados.

Debido a esto para todo proyecto es de suma importancia el análisis del costo, el esfuerzo y los beneficios que reportará y en este capítulo se describirá esto para el sistema propuesto.

5.2.Planificación basada en casos de uso.

La especificación de los requerimientos mediante Casos de Uso ha probado ser uno de los métodos más efectivos para capturar la funcionalidad de un sistema. Este hecho se puede apreciar en algunas metodologías actuales ampliamente difundidas, como el proceso Unificado de Rational (Rational Unified Process), en las cuales se propone especificar la funcionalidad de los sistemas mediante la utilización de Casos de Uso.

Se trata de un método de estimación del tiempo de desarrollo de un proyecto mediante la asignación de "pesos" a un cierto número de factores que lo afectan, para finalmente, contabilizar el tiempo total estimado para el proyecto a partir de esos factores.

Paso 1. Factor de Peso de los actores sin ajustar (UAW)

Tipo de actor	Descripción	Factor de peso	Actores	Total
Simple	Sistema con sistema a través de interfaz de programación.	1	0	0
Medio	Sistema con sistema mediante protocolo de interfaz basada en texto.	2	0	0

Complejo	Persona que interactúa con el sistema mediante interfaz gráfica.	3	1	3

UAW = S(Factor * Actores) UAW 3

Tabla 5.1- Factor de peso de los actores sin ajustar.

Paso 2. Factor de peso de los Casos de Uso sin ajustar (UUCW)

Tipo de CU	**Descripción**	**Peso**	**Cantidad de CU**	**Total**
Simple	El caso de uso tiene de 1 a 3 transacciones.	5	9	45
Medio	El caso de uso tiene de 4 a 7 transacciones.	10	0	0
Complejo	El caso de uso tiene más de 8 transacciones.	15	0	0

UUCW = Sumatoria(Factor * CantCU) UUCW 45

Tabla 5.2- Factor de peso de los casos de usos sin ajustar

Paso 3. Determinar los puntos de caso de uso sin ajustar (UUCP).

UUCP = UAW + UUCW = 3+45 = 48

Paso 4. Determinar los factores de complejidad técnicos (TCF).

El factor de complejidad técnica (TCF) se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada factor se cuantifica en un valor desde 0 (aporte irrelevante) hasta 5 (aporte muy relevante).

Factor	**Descripción**	**Peso**	**Valor asignado**	**Total**
T1	Sistema distribuido	2	0	0
T2	Tiempo de respuesta	1	4	4
T3	Eficiencia del usuario final	1	2	2
T4	Funcionamiento Interno complejo	1	3	3
T5	El código debe ser reutilizable	1	4	4
T6	Facilidad de instalación	0,5	2	1
T7	Facilidad de uso	0,5	4	2
T8	Portabilidad	2	3	6
T9	Facilidad de cambio	1	4	4
T10	Concurrencia	1	2	2

T11	Incluye objetivos especiales de seguridad	1	2	2
T13	Se requieren facilidades especiales de entrenamiento de usuarios	1	1	1
			Sumatoria	31

Tabla 5.3- Factor de Complejidad Técnica.

UCP = UUCP x TCF x EF

TCF = 0.6 + 0.01 x Σ (Pesoi x Valor asignadoi)

= 0.6 + 0.01 (Total Factor)

= 0,6 + 0,01 (31)

= 0.6 + 0.31

TCF = 0.91

Paso 5. Determinar el factor de ambiente (EF).

El factor de ambiente (EF) está relacionado con las habilidades y entrenamiento del grupo de desarrollo que realiza el sistema. Cada factor se cuantifica con un valor desde 0 (aporte irrelevante) hasta 5 (aporte muy relevante).

Factor	**Descripción**	**Peso**	**Valor asignado**	**Total**
E1	Familiaridad con el modelo de proyecto	1,5	3	4.5

	utilizado			
E2	Experiencia en la aplicación	0,5	1	0.5
E3	Experiencia en la orientación a objetivos.	1	3	3
E4	Capacidad del analista líder.	0,5	3	1,5
E5	Motivación.	1	5	5
E6	Estabilidad de requerimientos	2	4	8
E7	Personal Part–Time	-1	0	0
E8	Dificultad del lenguaje de programación	-1	3	-3
			Sumatoria	16,5

Tabla 5.4- Factor de Ambiente.

EF = 1.4 + (- 0.03 x Σ (Pesoi x Valor asignadoi))

= 1.4 + (- 0.03 x 16.5)

= 1.4 - 0.495

EF = 0.905

Paso 6. Determinar los puntos de caso de uso ajustados (UCP).

UCP = UUCP x TCF x EF donde,

UCP: Puntos de Casos de Uso ajustados

UUCP: Puntos de Casos de Uso sin ajustar

TCF: Factor de complejidad técnica

EF: Factor de ambiente

(UCP = UUCP x TCF x EF)

UCP = 48 x 0.91 x 0.905

UCP = 39.530

Paso 7. Determinar el esfuerzo:

Para obtener el factor de conversión (CF) se cuentan cuantos valores de los que afectan el factor ambiente (E1...E6) están por debajo de la media (3), y los que están por arriba de la media para los restantes (E7, E8). Si el total es 2 o menos se utiliza el factor de conversión 20 Horas-Hombre / Punto de Casos de uso. Si el total es 3 o 4 se utiliza el factor de conversión 28 Horas-Hombre / Punto de Casos de uso. Si el total es mayor o igual que 5 se recomienda efectuar cambios en el proyecto ya que se considera que el riesgo de fracaso del mismo es demasiado alto.

En este caso se puede decir que:

CF=20 (Factor de conversión)

E = UCP x CF

E = 39.530 x 20

E = 790.6 Horas /Hombre

E = 4.11 Mes/Hombre

Actividad	**Porcentaje %**	**Horas-Hombres**
Análisis	10	197.65
Diseño	20	395.3
Implementación	40	790.6
Pruebas	15	296.475
Sobrecarga (otras actividades)	15	296.475
Total	100	1976.5

Tabla 5.5- Esfuerzo del Proyecto.

Esfuerzo ⇒Tiempo

E (total): Esfuerzo Total

CH: Cantidad de Hombres

TDES: Tiempo de Desarrollo

Esfuerzo Total:

TDES (total) = E (total) / CH (total)

TDES (total) = 4.11 Mes/Hombre / 2 Hombres = 2.055 meses

Paso 8. Determinar el Costo Total a partir del esfuerzo en HH:

Costo Total (a partir del esfuerzo en HH)

C (total) = E (total en MH) x CHM CHH: Costo por Hombre Mes

C (total) = 4.11 * 100

= $ 411

Valores Finales	
Tiempo de Desarrollo	4.11 meses
Cantidad de hombres	2 hombres
Costo del desarrollo del sistema	$ 411

Tabla 5.6- Resultados sobre el estudio de factibilidad.

5.3.Beneficios tangibles e intangibles.

5.3.1.Beneficios tangibles

El sistema automatizado para la gestión de los procesos en la residencia estudiantil no se ha realizado con fines comerciales puesto que su principal objetivo es resolver todos los problemas existentes durante la gestión de la información a la hora de controlar todas las actividades que se realizan.

5.3.2.Beneficios intangibles.

- Disminución del tiempo y esfuerzo que se invierte en las tareas que se realizan, hasta ahora, de forma manual.
- Mejor calidad de planificación.
- Centralizar todos los datos e informaciones del negocio en un sitio para facilitar la búsqueda de información.
- Fácil detección de errores.
- Fácil procesamiento de la información y obtención dinámica de reportes en cualquier momento que se desee.

5.4.Análisis de costos y beneficios.

La publicación del sistema traerá consigo considerables beneficios a la hora de controlar y distribuir las actividades de la residencia, pues reduce considerablemente el tiempo que se emplea realizando esta tarea manualmente, además de que permitirá generar reportes con una mayor agilidad y veracidad de los datos.

Teniendo en cuenta el análisis realizado de los beneficios que reporta el sistema y que no fue necesario realizar inversiones en equipos técnicos para el desarrollo del producto, ni en la tecnología utilizada debido a que es totalmente libre, se concluye que ha sido factible llevar a cabo la realización del software.

5.5.Conclusiones.

En este capítulo se especificó el estudio de la factibilidad del software en cuanto al tiempo de desarrollo, esfuerzo, el costo al desarrollarlo, el análisis de los costos y beneficios, y el análisis de los beneficios tangibles e intangibles, concluyendo que ha sido factible la realización del sistema.

CONCLUSIONES

En este trabajo de diploma se realizó un amplio estudio de la residencia estudiantil de nuestra facultad (2), se profundizó en sus características principales y su funcionamiento, logrando entender a plenitud las actividades que se desarrollan en la misma y seleccionándose una mejor vía de planificación y distribución. Se realizó una buena selección de las herramientas, lenguajes y tecnologías necesarias para la confección de un sistema que cumpliera con los requerimientos definidos.

RECOMENDACIONES

Se propone:

- Estudiar y analizar otros procesos que se realicen en la residencia para agregar nuevas funcionalidades al sistema.

BIBLIOGRAFÍA

1. Anónimo, Metodología, http://es.wikipedia.org/wiki/Metodolog%C3%ADa

2. Araceli Torres Lecuanda, Metodologías modernas de desarrollo de Sistemas de Información, http://www.monografias.com/trabajos12/docmento/docmento.shtml#intro

3. Anónimo, Proceso unificado de Racional, http://es.wikipedia.org/wiki/Proceso_Unificado_de_Rational

4. Anónimo, Lenguajes de programación, http://www.frt.utn.edu.ar/sistemas/paradigmas/lenguajes.htm

5. Disponible en: http://es.wikipedia.org/wiki/UML#Diagramas

6. Ing. Enrique Canseco, Metodología MSF, http://www.coworker.com.mx/nota.asp?id=28

7. Ing. María A. Mendoza Sánchez, Metodologías De Desarrollo De Software, http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html 10/02/2008

8. Anónimo, Herramientas Case, http://es.wikipedia.org/wiki/CASE

9. Aurora Vizcaíno, Ismael Caballero, http://alarcos.inf-cr.uclm.es/per/fgarcia/isoftware/doc/LabTr1_RationalRose.pdf

10. Anónimo, Sistemas de gestión de bases de datos, http://es.wikipedia.org/wiki/SGBD 04/01/2008

11. Anónimo, http://www.dbrunas.com.ar/postgres/migrapg.pdf 25/01/2008

12. Mota, S.A. Entonación de Bases de Datos

13. Daniel Pecos, PostgreSQL vs. MySQL, http://www.netpecos.org/docs/mysql_postgres/index.html

14. Anónimo, PHP, http://es.wikipedia.org/wiki/PHP

15. Anónimo, Introducción al lenguaje PERL., http://kataix.umag.cl/~mmarin/topinf/perl.html

16. Anónimo, ASP.NET, http://es.wikipedia.org/wiki/ASP.NET

17.Anónimo, ¿Qué es ASP.NET?, http://es.gotdotnet.com/quickstart/aspplus/doc/whatisaspx.aspx

18.Anónimo, Lenguaje de programación Java, http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Java

19.Patrón arquitectónico http://es.wikipedia.org/wiki/Modelo_Vista_Controlador

20.Framework CakePHP (Sin traducir)http://www.cakephp.org/

21.[Disponible en http://httpd.apache.org/docs/2.0/new_features_2_0.html]

22.Anónimo, Conceptos básicos de Dreamweaver8 (I), http://www.aulaclic.es/dreamweaver8/t_1 1htm.

23.Jennifer Taylor Título, Nuevas funciones y ventajas de Dreamweaver8, http://www.adobe.com/es/devnet/dreamweaver/articles/dw8_newfeatures.html

24.Anónimo, http://descargar.mp3.es/lv/group/view/kl61486/NuSphere_PhpED.htm

25.Juan Manuel Lemus, Smarty: razones para utilizar plantillas en mis proyectos, http://www.maestrosdelweb.com/editorial/smarty-plantillas/

26.http://codeigniter.com/

27.Roger S. Pressman,http://bibliodoc.uci.cu/pdf/reg02689.pdf.

28.Anónimo, http://es.wikipedia.org/wiki/Cliente-servidor

ANEXOS

Anexo 1. Descripción de los Casos de Uso del Negocio:

Distribuir Beca

(1) Caso de uso del negocio	**Distribuir Beca**
Actores	Vicedecano
Resumen	El caso de uso se inicia cuando el Vicedecano realiza la entrega de los datos de los estudiantes y los edificios para realizar la distribución de la Beca. El caso de uso finaliza cuando el Vicedecano verifica la distribución de la Beca.
Casos de uso asociados	-
Acción del actor	**Respuesta del proceso de negocio**
1. El Vicedecano entrega los datos para realizar la distribución de la Beca	2. El planificador consulta los datos y distribuye la Beca.
	3. El planificador entrega el listado de estudiantes por edificio al Vicedecano.
4. El Vicedecano verifica la distribución de la Beca	
Cursos Alternativos	
	4.1 Si la distribución de la Beca no está correcta por cualquier error entonces se efectúa nuevamente el caso de uso.
Pre condiciones	El Vicedecano debe entregar los datos con suficiente tiempo para realizar la distribución.
Pos condiciones	El planificador debe entregar la distribución en tiempo y forma para que el Vicedecano verifique si existen errores y poder erradicarlos lo más pronto posible.

Distribuir Cuartelería

(2) Caso de uso del negocio	**Distribuir Cuartelería**
Actores	Vicedecano
Resumen	El caso de uso se inicia cuando el Vicedecano entrega los datos, es decir el listado de estudiantes por edificios para realizar el plan de la Cuartelaría. El caso de uso finaliza cuando el Vicedecano verifica el plan de la Cuartelería.
Casos de uso asociados	-
Acción del actor	**Respuesta del proceso de negocio**
1. El Vicedecano entrega los datos para realizar el plan de la Cuartelería	2. El planificador consulta los datos y realiza el plan de Cuartelería.
	3. El planificador entrega el plan de la Cuartelería al Vicedecano.
4. El Vicedecano verifica el plan de la Cuartelería.	
Cursos Alternativos	
	4.1 Si el plan de la Cuartelería es incorrecto por cualquier error entonces se inicia nuevamente el caso de uso.
Pre condiciones	El Vicedecano debe entregar los datos con suficiente tiempo para realizar el plan de la Cuartelería.
Pos condiciones	El planificador debe entregar el plan confeccionado en tiempo y forma para que el Vicedecano verifique si existen errores y poder erradicarlos rápidamente.

Distribuir Guardia

(3) Caso de uso del negocio	**Distribuir Guardia**
Actores	Vicedecano
Resumen	El caso de uso se inicia cuando el Vicedecano entrega los datos de los estudiantes y de los profesores para realizar el plan de la Guardia. El caso de uso finaliza cuando el Vicedecano verifica el plan de Guardia.
Casos de uso asociados	-
Acción del actor	**Respuesta del proceso de negocio**
1. El Vicedecano entrega los datos para realizar el plan de Guardia	2. El planificador consulta los datos y realiza el plan de Guardia.
	3. El planificador entrega el plan de la Guardia al Vicedecano.
4. El Vicedecano verifica el plan de la Guardia.	
Cursos Alternativos	
	4.1 Si el plan de Guardia es incorrecto por cualquier error entonces se inicia nuevamente el caso de uso.
Pre condiciones	El Vicedecano debe entregar los datos con suficiente tiempo para realizar el plan Guardia.
Pos condiciones	El planificador debe entregar el plan realizado en tiempo y forma para que el Vicedecano verifique si existen errores y poder erradicarlos rápidamente.

ReporteEstudiante

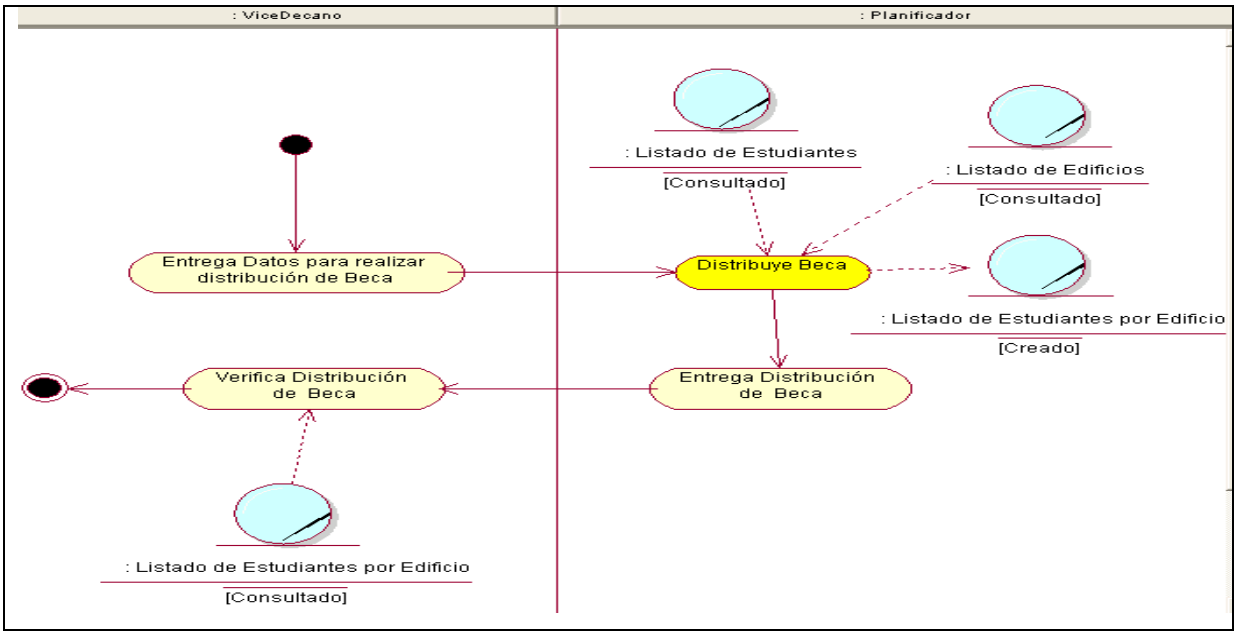
(4) Caso de uso del negocio	**Reporte Estudiante**
Actores	Vicedecano
Resumen	El caso de uso se inicia cuando el Vicedecano entrega los datos, es decir el listado de aptitudes de los estudiantes para realizar el Reporte. El caso de uso finaliza cuando el Vicedecano verifica el Reporte del Estudiante.
Casos de uso asociados	-
Acción del actor	**Respuesta del proceso de negocio**
1. El Vicedecano entrega los datos para realizar el Reporte del Estudiante	2. El planificador consulta los datos y realiza el Reporte del Estudiante.
	3. El planificador entrega el Reporte del Estudiante al Vicedecano.
4. El Vicedecano verifica el Reporte del Estudiante.	
Cursos Alternativos	
	4.1 Si el Reporte del Estudiante no es el que se esperaba entonces se inicia nuevamente el caso de uso.
Pre condiciones	
Pos condiciones	

3.2-Reglas del Negocio

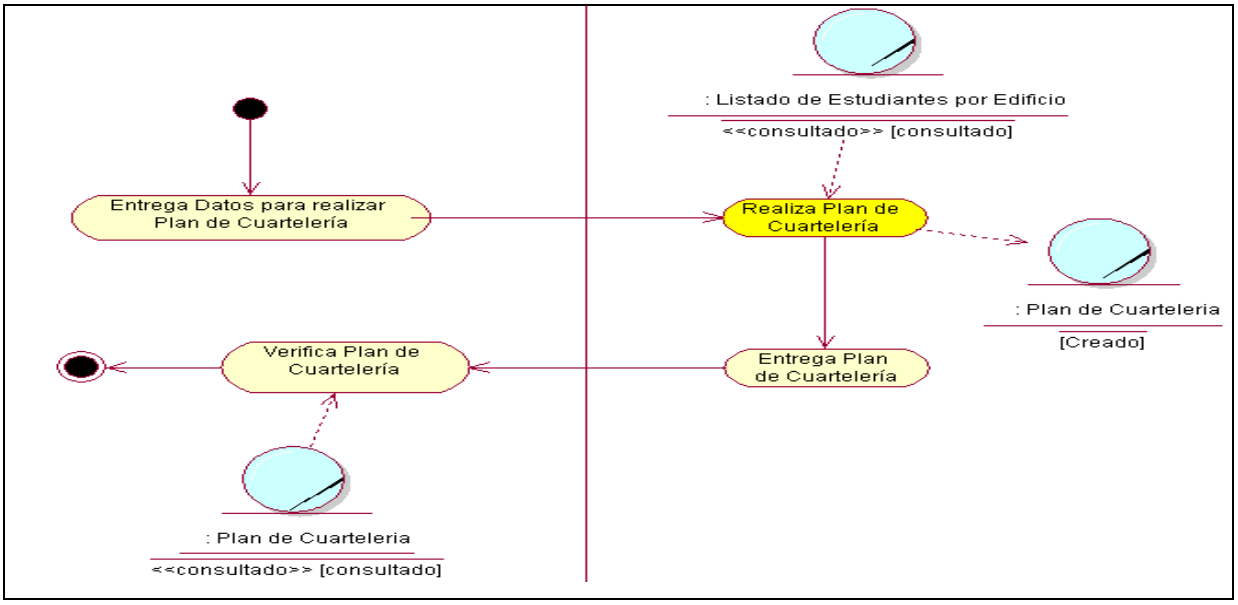
El ViceDecano(Cliente) es el encargado de confeccionar y distribuir la planificación de las actividades que se realizan en la Residencia.

Anexo 2. Diagrama de Actividades

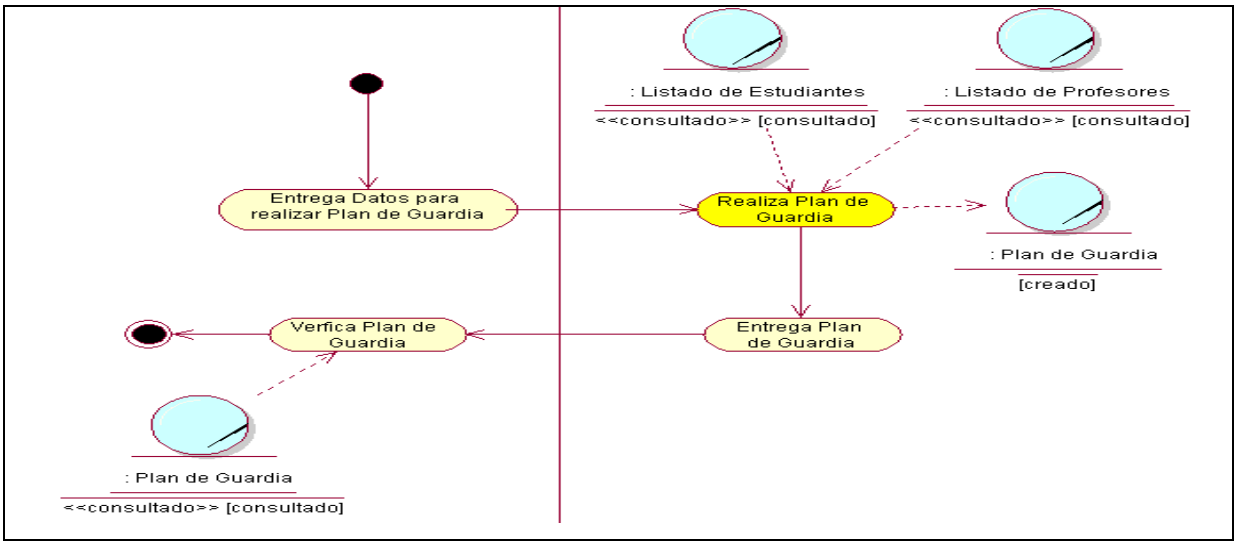
DA-Distribuir Beca



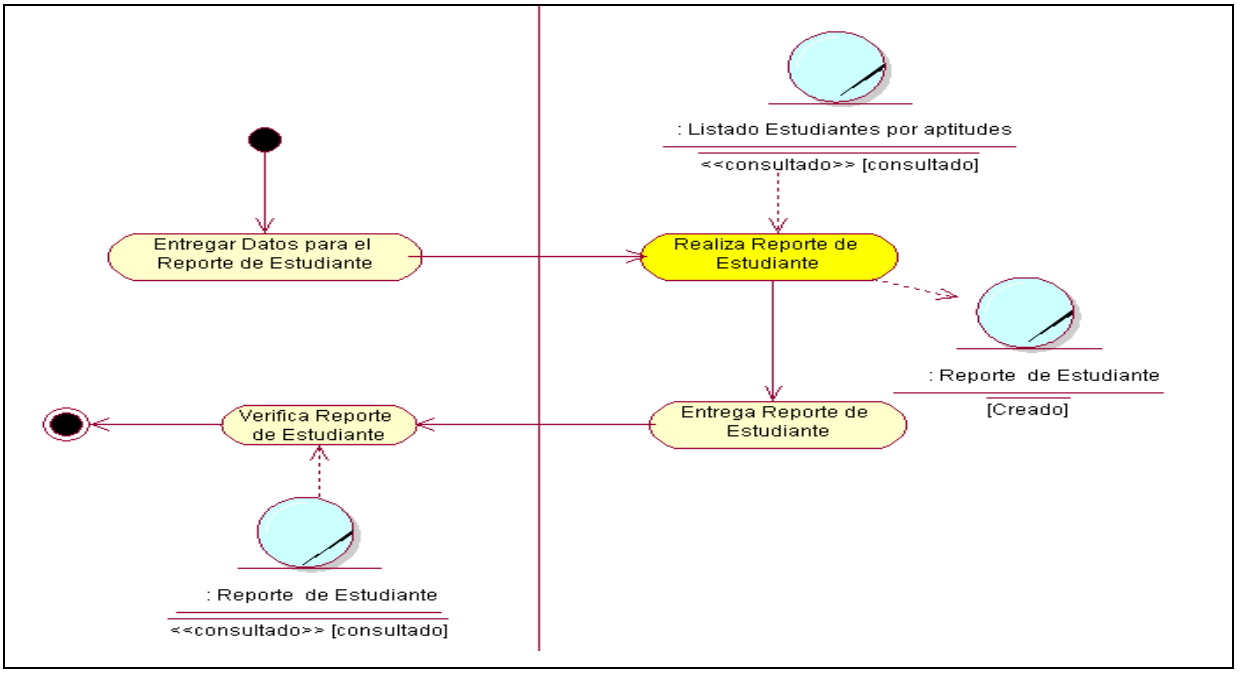
DA-Distribuir Cuartelería



DA-Distribuir Guardia



DA-Reporte Estudiante



Anexo 3. Descripción del los casos de uso del sistema.

Autenticar Usuario

Caso de Uso:	Autenticar Usuario
Actores:	Usuario (ViceDecano)
Resumen:	El CU se inicia cuando se introduce un usuario y contraseña en el sistema para autenticarse, y este, si los datos introducidos son correctos le asigna al usuario sus privilegios y le da acceso a la página principal y en caso contrario muestra un mensaje de error.
Referencias	RF1.
Prioridad	Crítico

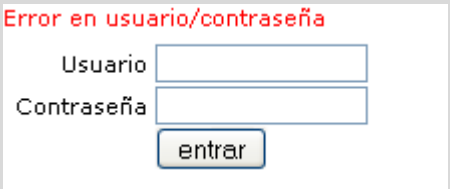
Flujo Normal de Eventos

Acción del Actor	**Respuesta del Sistema**
1. El usuario entra sus datos al sistema.	1.1 El sistema verifica los datos. 1.2 Si los datos están correctos el sistema le da entrada y le asigna los privilegios de administración.

Prototipo de Interfaz

Acción del Actor	**Respuesta del Sistema**
	1.2 Si el sistema no identifica los datos del usuario, se muestra un mensaje de error.

Prototipo de Interfaz



Poscondiciones	La autenticación se realiza de forma correcta y se muestran las funcionalidades a realizar en la Página Principal

Gestionar Beca

Caso de Uso:	Gestionar Beca
Actores:	Usuario (ViceDecano)
Resumen:	El CU se inicia cuando se el ViceDecano Solicita Gestionar_Beca en la página Principal, y es redirecccionado a otra página donde puede realizar diferentes acciones, como son: Asignar Beca, Modificar Asignación de Beca, Mostrar Asignación de Beca y Modificar la Beca.
Referencias	RF3
Prioridad	Crítico

Escenario Asignar_Beca	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1 El ViceDecano solicita distribuir Beca. 2 Selecciona los estudiantes no ubicados por sexo y los edificios con sus capacidades libres. 3 Selecciona la opción de Distribuir.	1.1 Redirecciona a la página “Distribuir”. 3.1 Distribuye la Beca.
Poscondiciones	La Beca queda correctamente distribuida.
Precondiciones	Tiene que existir los edificios y los estudiantes.

Escenario Modificar Asignación de Beca	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1 El ViceDecano solicita un movimiento en la asignación de la Beca. 2. Selecciona los estudiantes de acuerdo al Edificio, el Paso de	1.1 Redirecciona a la página “Movimientos”.

Escalera y el Apartamento. 3. Selecciona la opción de realizar los movimientos que sean necesarios, incluyendo la opción de eliminar estudiante.	3.1 Realiza los movimientos requeridos.
Poscondiciones	La distribución de la Beca sufre una modificación y queda actualizada la misma.
Precondiciones	Tiene que estar distribuido al menos un estudiante.
Escenario Mostrar Asignación de Beca	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El ViceDecano solicita ver estado de asignación de la Beca. 2. Selecciona un edificio.	1.1 Redirecciona a la página “Estado”. 2.1 Muestra la ubicación de los estudiantes en el edificio.
Poscondiciones	La Beca sufre una modificación y queda actualizada la misma.
Precondiciones	Tiene que estar distribuido al menos un estudiante
Escenario Modificar Beca	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El ViceDecano solicita modificar datos de la Beca. 2. Seleciona un edificio 3. Modifica datos de la Beca y da la opción de Guardar.	2.1 Muestra un formulario con los datos del edificio por apartamento. 3.1 Guarda los datos introducidos.
Poscondiciones	La Beca sufre una modificación y queda actualizada la misma.
Precondiciones	Tiene que existir al menos un edificio.

Gestionar Cuartelería

Caso de Uso:	Gestionar Cuartelería
Actores:	Usuario (ViceDecano)
Resumen:	El CU se inicia cuando se el ViceDecano Solicita Gestionar Cuartelería en la página Principal, y es redirecccionado a otra página donde puede realizar las siguientes acciones : Asignar Cuartelería y Mostrar Estado de la Cuartelería.
Referencias	RF4
Prioridad	Crítico

Escenario Asignar Cuartelería	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El ViceDecano solicita distribuir Cuartelería. 2. Selecciona un edificio y las fechas de inicio y fin de la Cuartelería. 3. Selecciona la opción Distribuir	1.1 Redirecciona a la página “Distribuir”. 3.1 Distribuye la Cuartelería.
Poscondiciones	La Cuartelería queda correctamente distribuida.
Precondiciones	Tiene que existir al menos una distribución de Beca.

Escenario Mostrar Estado de la Cuartelería.	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El ViceDecano solicita ver estado de la Cuartelería 2. Selecciona un edificio, un paso de escalera y las fechas de inicio y fin de la Cuartelería. 3. Marca la opción de Mostrar.	1.2 Redirecciona a la página “Estado”. 3.1 Muestra estado de la Cuartelería según los criterios.

Poscondiciones	
Precondiciones	Tiene que existir al menos una distribución de Cuartelería.

Gestionar Guardia

Caso de Uso:	Gestionar Guardia
Actores:	Usuario (ViceDecano)
Resumen:	El CU se inicia cuando se el ViceDecano Solicita Gestionar Guardia en la página Principal, y es redirecccionado a otra página donde puede realizar las siguientes acciones : Asignar Guardia y Mostrar Estado de la Guardia.
Referencias	RF5
Prioridad	Crítico

Escenario Asignar Guardia	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El ViceDecano solicita distribuir Guardia. 2. Selecciona las fechas de inicio y fin de la guardia. 3. Marca la opción de Distribuir	1.1 Redirecciona a la página “Distribuir”. 3.1 Distribuye la Guardia.
Poscondiciones	La Guardia queda correctamente distribuida.
Precondiciones	Tiene que exitir al menos un profesor y un grupo.

Escenario Mostrar Estado de la Guardia.	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El ViceDecano solicita ver estado de la Guardia. 2. Selecciona las fechas de inicio y fin de la Guardia.	1.1 Redirecciona a la página “Estado”. 2.1 Muestra estado de la Guardia.

Poscondiciones	
Precondiciones	Tiene que exitir al menos una distribución de la Guardia

Gestionar Aptitud

Caso de Uso:	Gestionar Aptitud
Actores:	Usuario (ViceDecano)
Resumen:	El CU se inicia cuando se el ViceDecano Solicita Gestionar Aptitud en la página Principal, y es redirecccionado a otra página donde puede realizar la las siguientes acciones: Insertar Aptitud, Eliminar Aptitud y Asignar Aptitud a un Estudiante.
Referencias	RF6
Prioridad	Crítico
Escenario Insertar Aptitud	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El ViceDecano solicita Insertar Nueva Aptitud. 2. Escribe el nombre de la Aptitud que desea añadir. 3. Marca la opción de Insertar	3.1 Inserta la aptitud y muestra la lista de las aptitudes actualizada.
Poscondiciones	Se actualiza la lista de aptitudes.
Precondiciones	
Escenario Eliminar Aptitud	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El ViceDecano solicita elimina la aptitud que desee.	1.1 Elimina la aptitud y muestra la lista de las aptitudes actualizada.

Poscondiciones	Se actualiza la lista de aptitudes.
Precondiciones	Tiene que existir al menos una aptitud.
Escenario Asignar Aptitud a un Estudiante	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El ViceDecano Asignar Aptitud a un Estudiante. 2. El ViceDecano busca un Estudiante 3. Marca la(s) aptitud(es) que sean compatibles con el Estudiante. 4. Selecciona la opción de asignar.	1.1 Redirecciona a la página “Asignar Aptitud’ 4.1 Guarda los cambios realizados, actualizando la información del estudiante.
Poscondiciones	Se actualiza la información del estudiante.
Precondiciones	Tiene que existir al menos una aptitud y un estudiante..

Gestionar Reporte del Estudiante

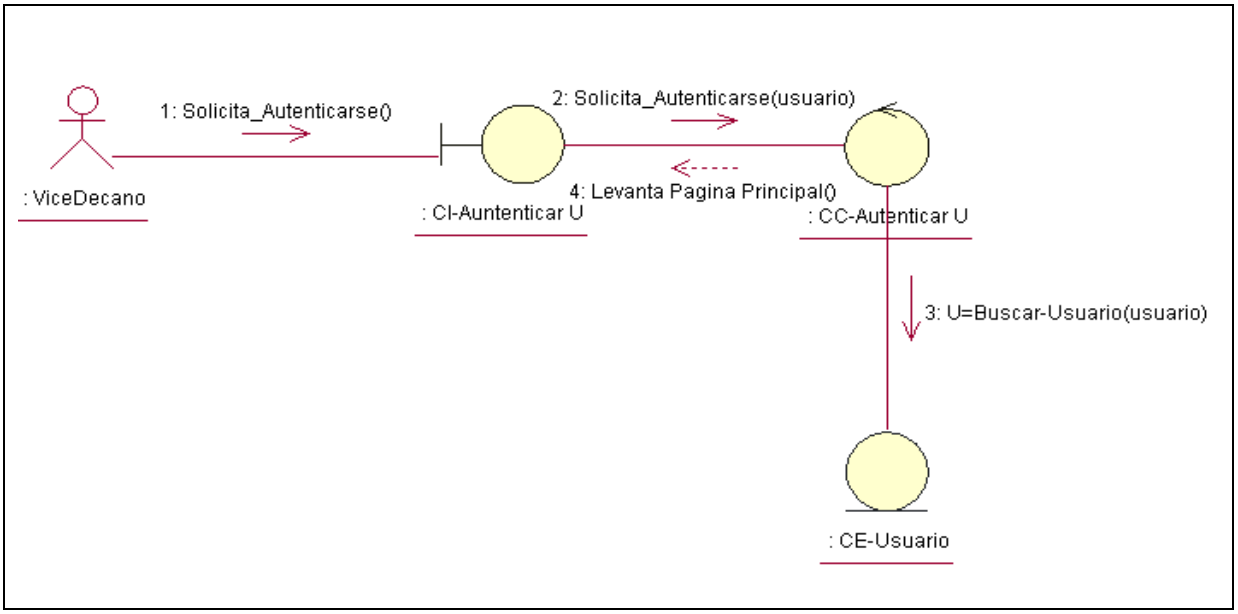
Caso de Uso:	Gestionar Reporte de Estudiante
Actores:	Usuario (ViceDecano)
Resumen:	El CU se inicia cuando el ViceDecano Solicita Gestionar Reporte del Estudiante y este es redireccionado a la página “Reporte”, donde se obtiene el reporte del estudiante.
Referencias	RF7
Prioridad	Crítico
Escenario ReporteEstudiante	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

1. El ViceDecano solicita ver Reporte 2. Busca un Estudiante.	1.1 Redirecciona a la página “Reporte”. 2.1 Muestra los datos del estudiante
Poscondiciones	
Precondiciones	Tiene que exitir al menos un Estudiante.

Anexo 4. Diagramas de Interacción.

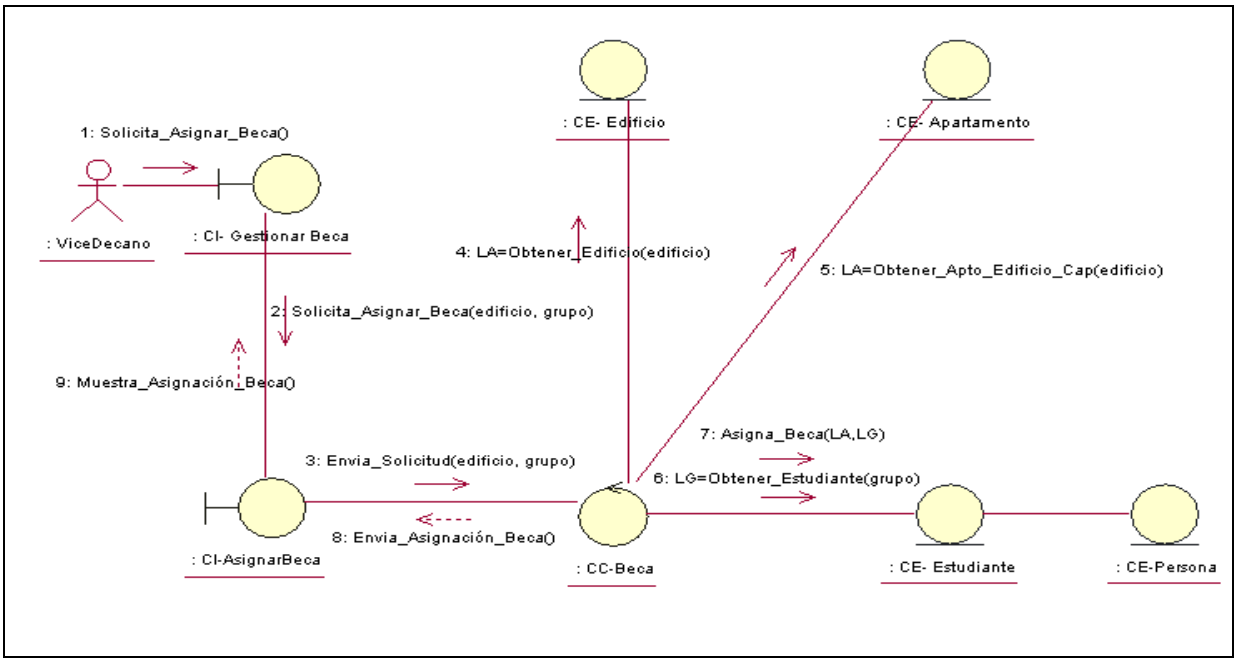
6.1-Diagrama de Colaboración del Análisis

Autenticarse

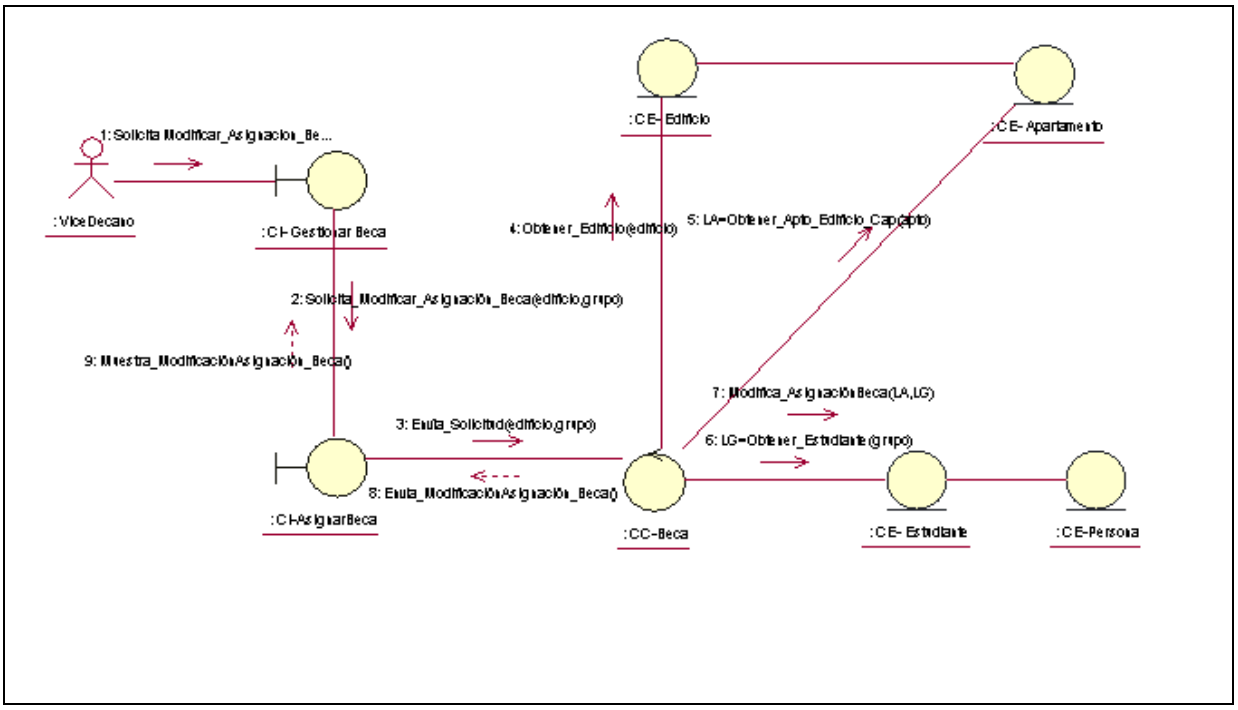


Gestionar Beca

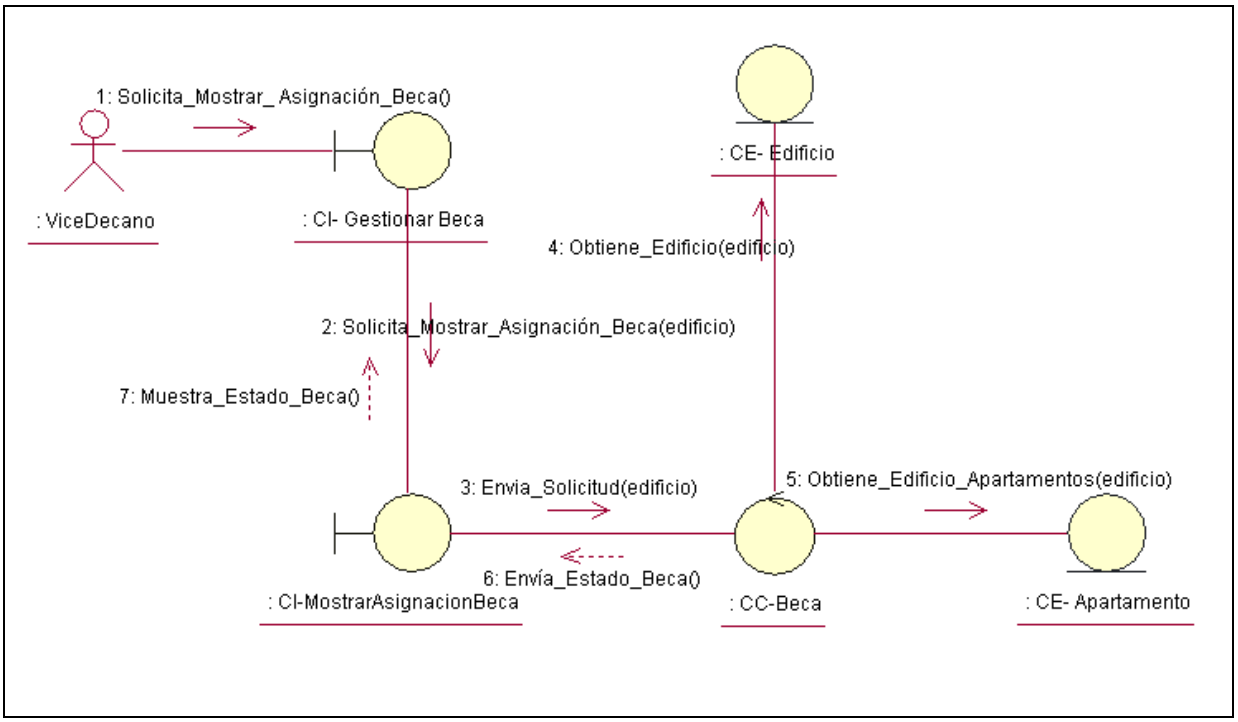
Asignar Beca



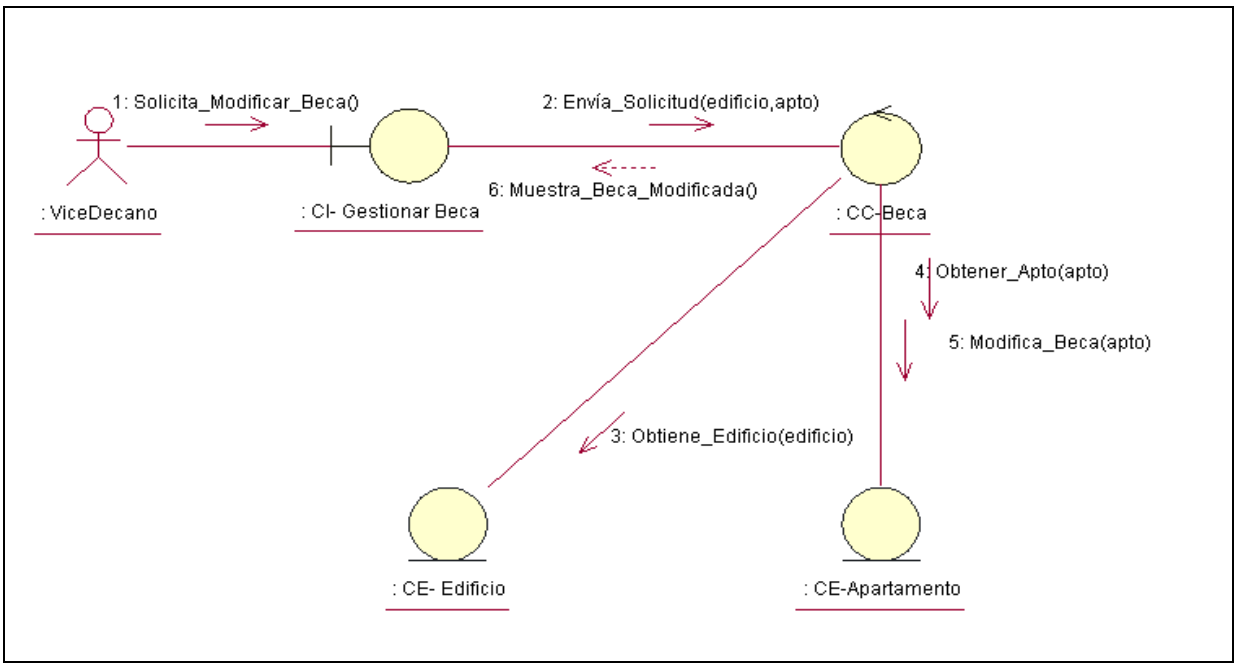
Modificar Asignación de Beca



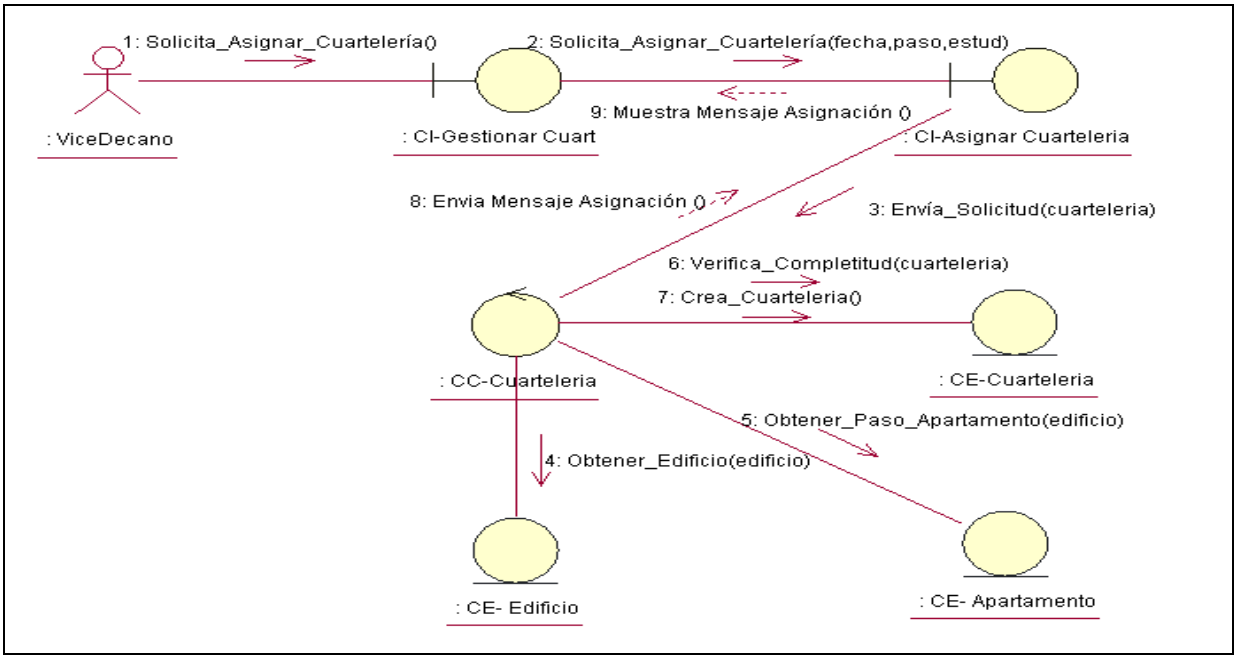
Mostrar Asignación de Beca



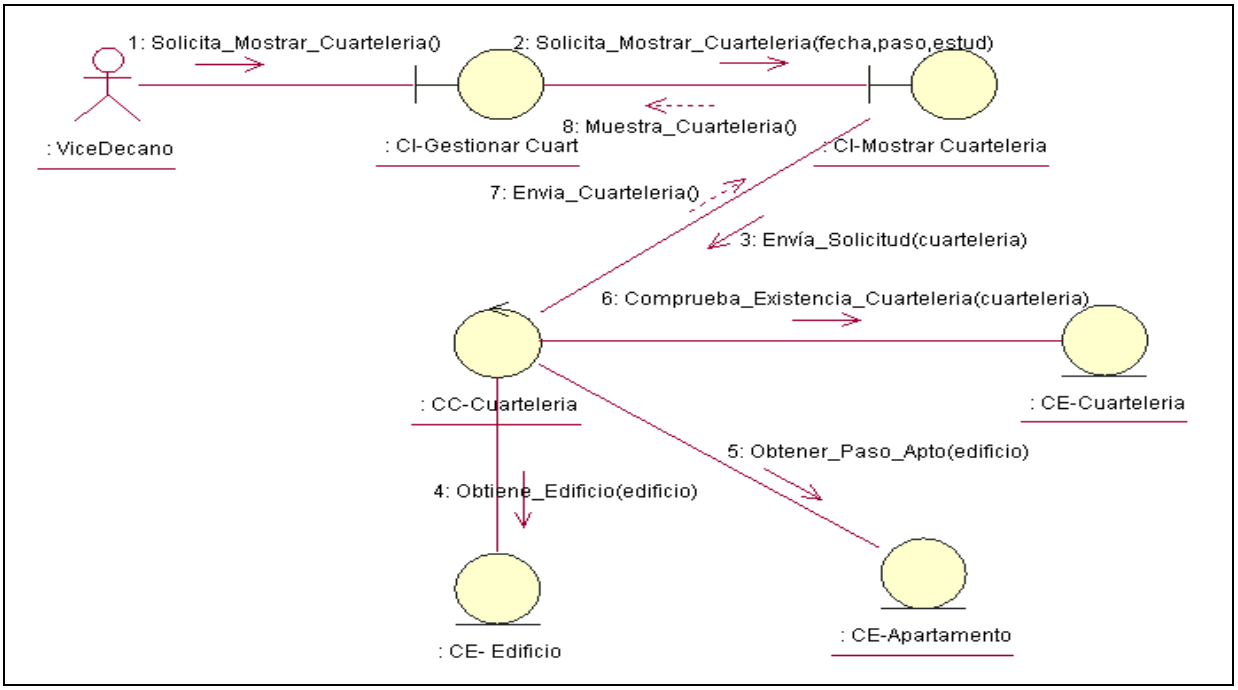
Gestionar Beca (Modificación)



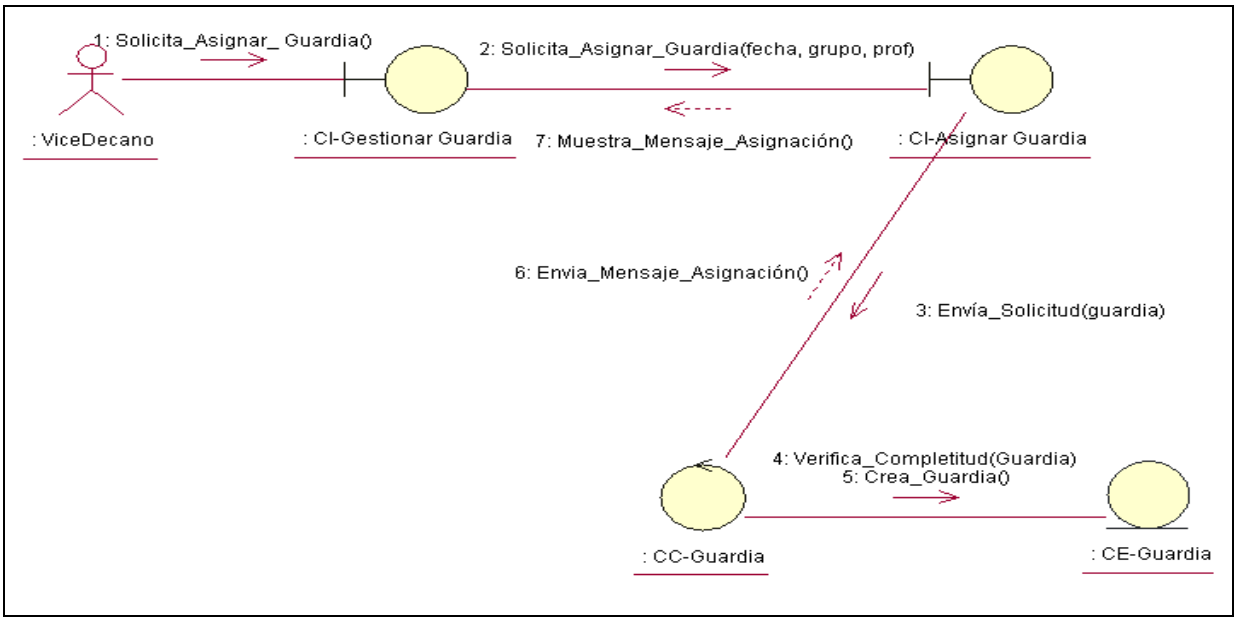
Asignar Cuartelería



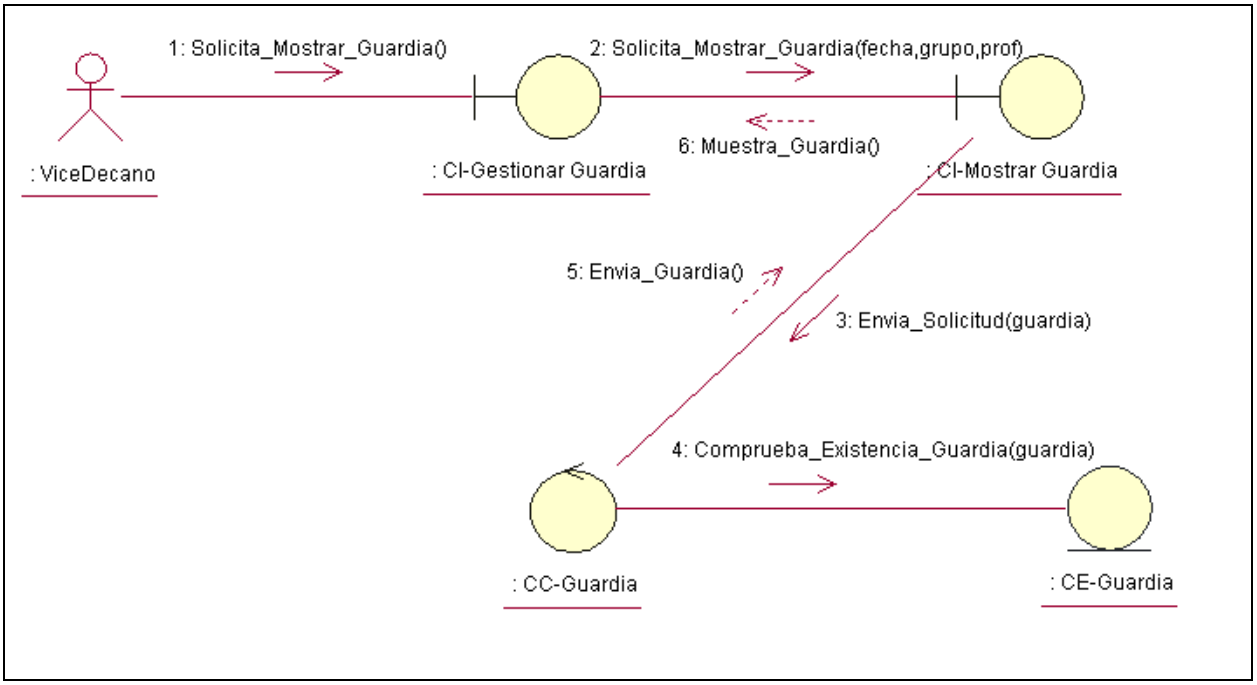
Mostrar Cuartelería



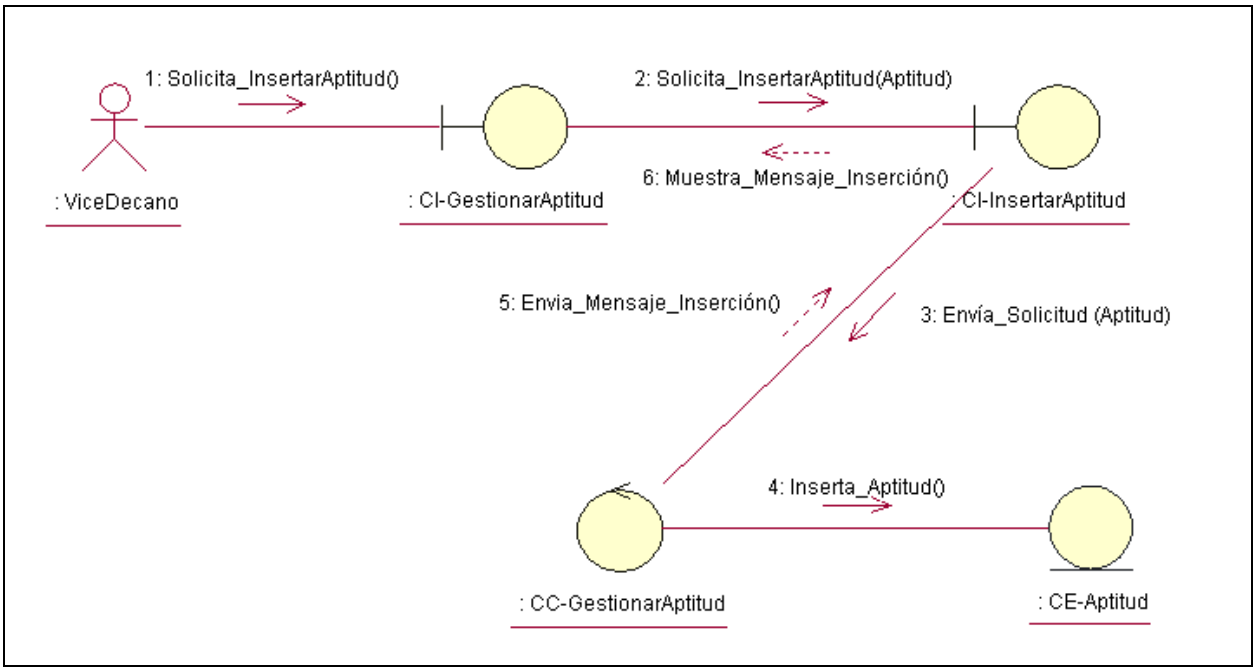
Asignar Guardia



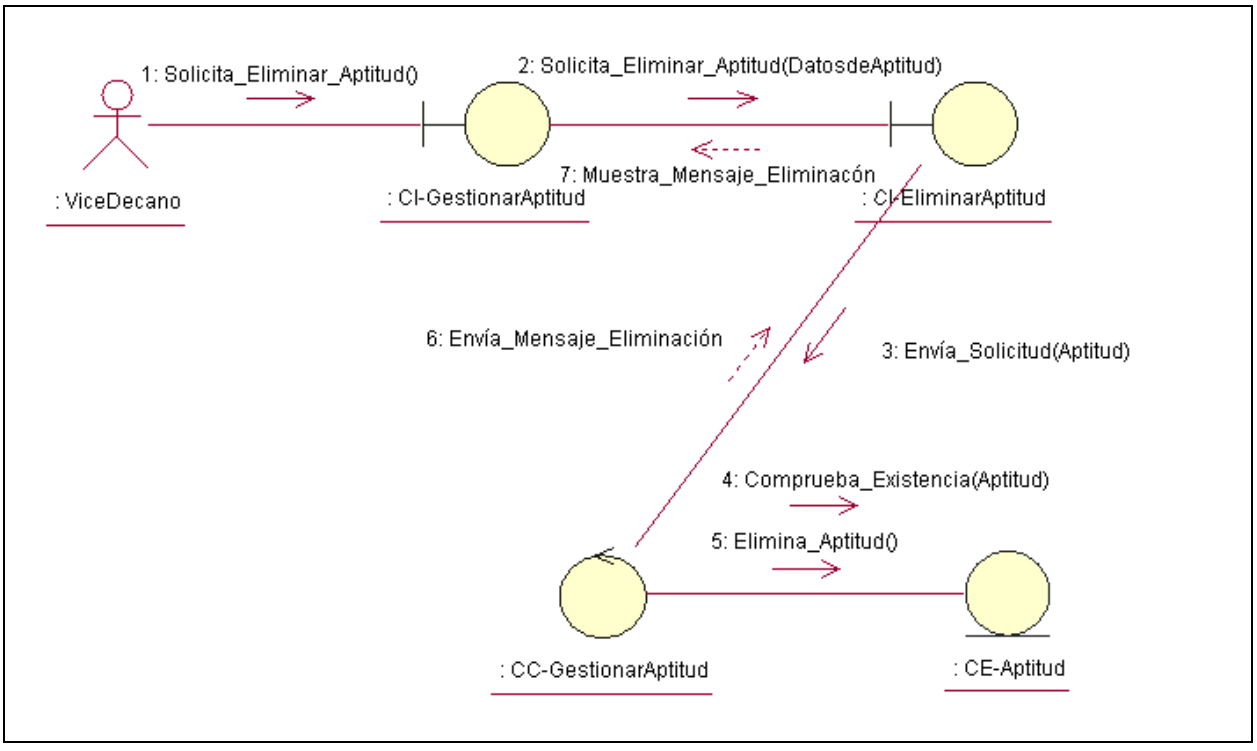
Mostrar Guardia



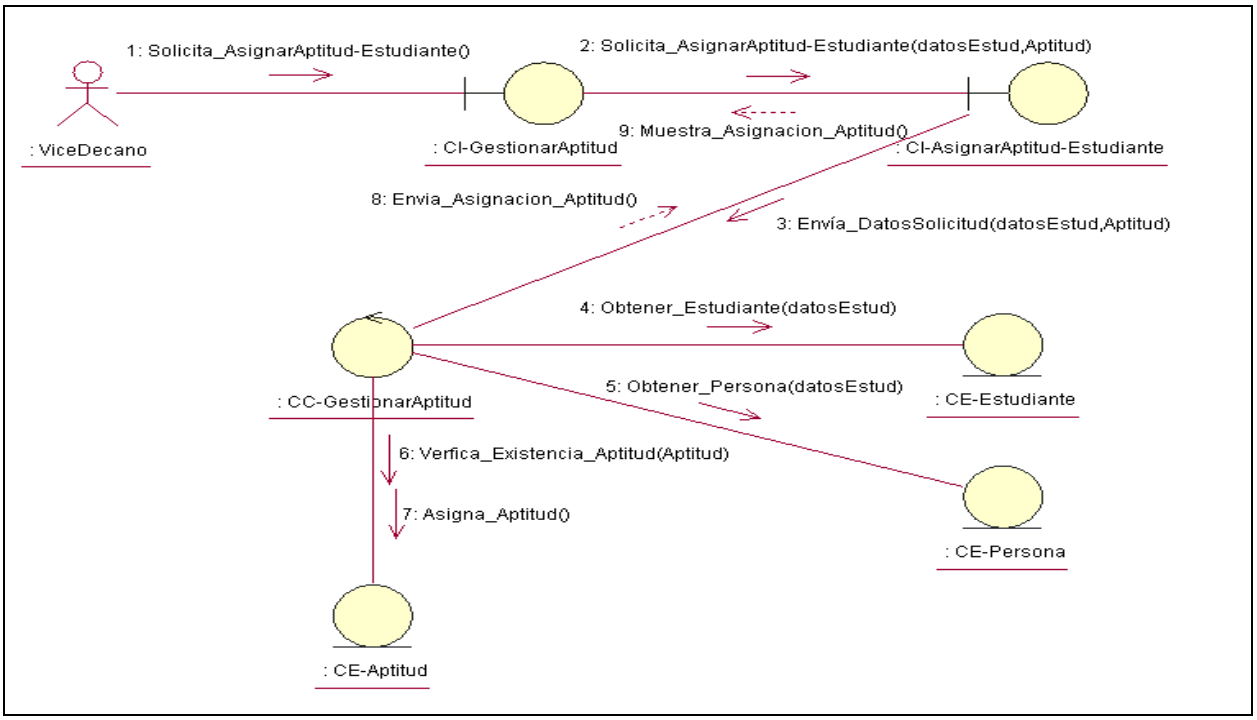
Insertar Aptitud



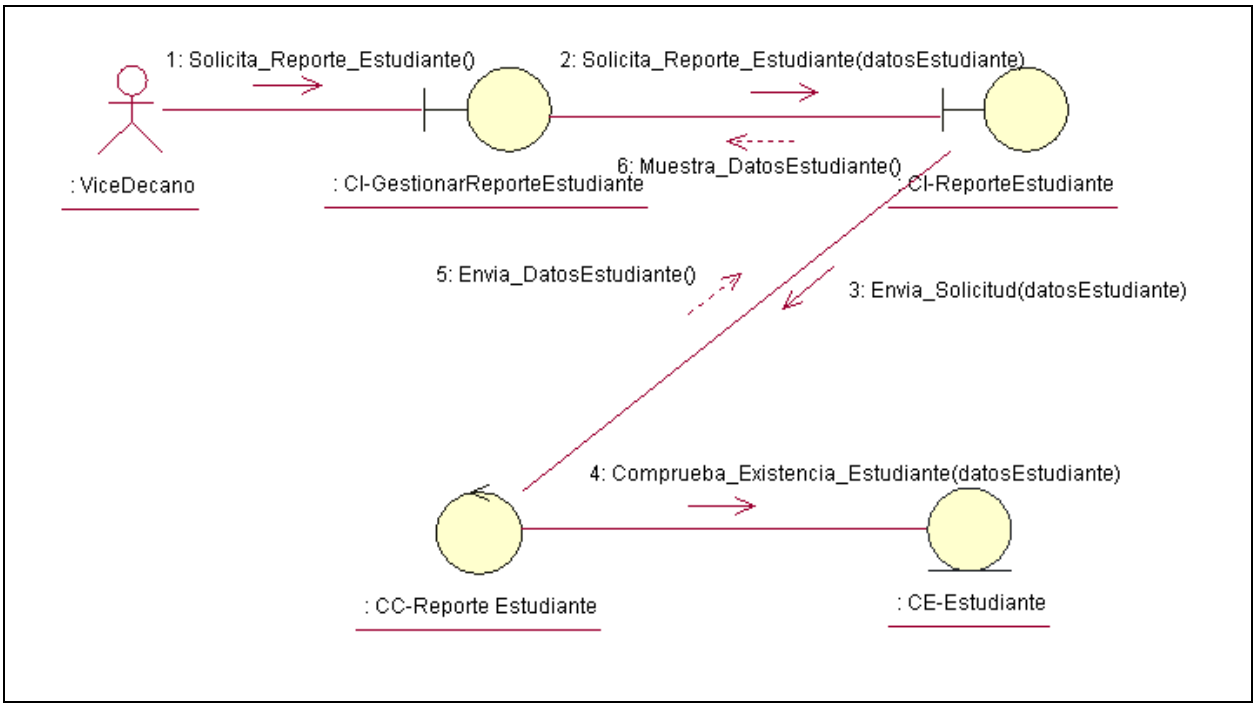
Eliminar Aptitud



Asignar Aptitud a un Estudiante

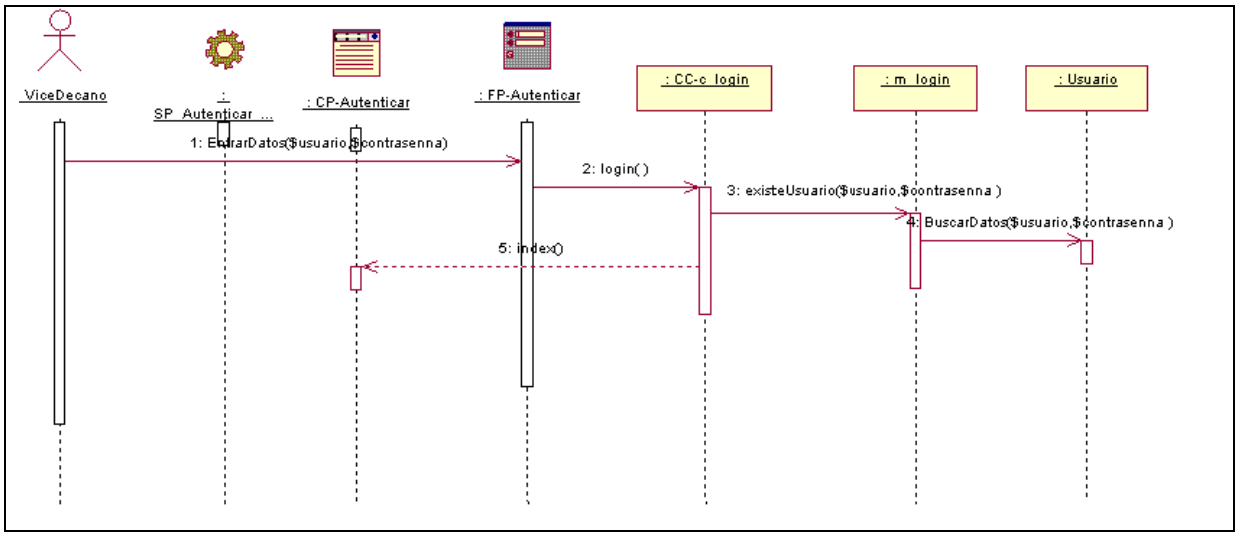


Reporte de Estudiante

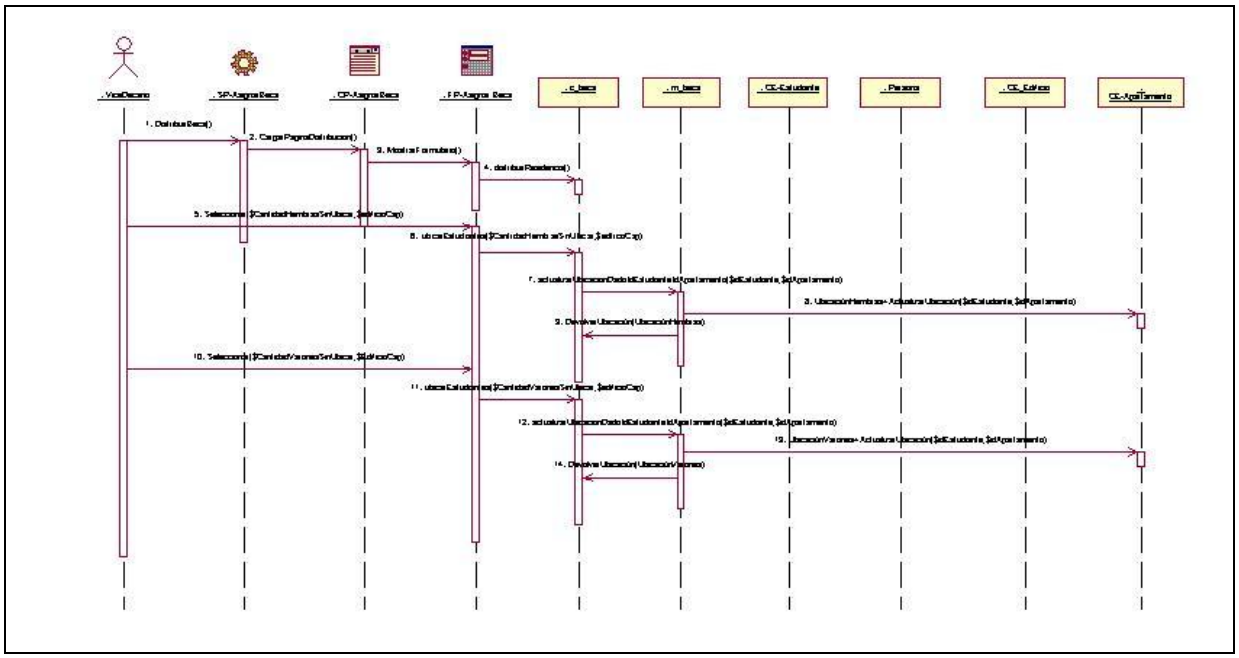


6.2-Diagrama de Secuencia del Diseño

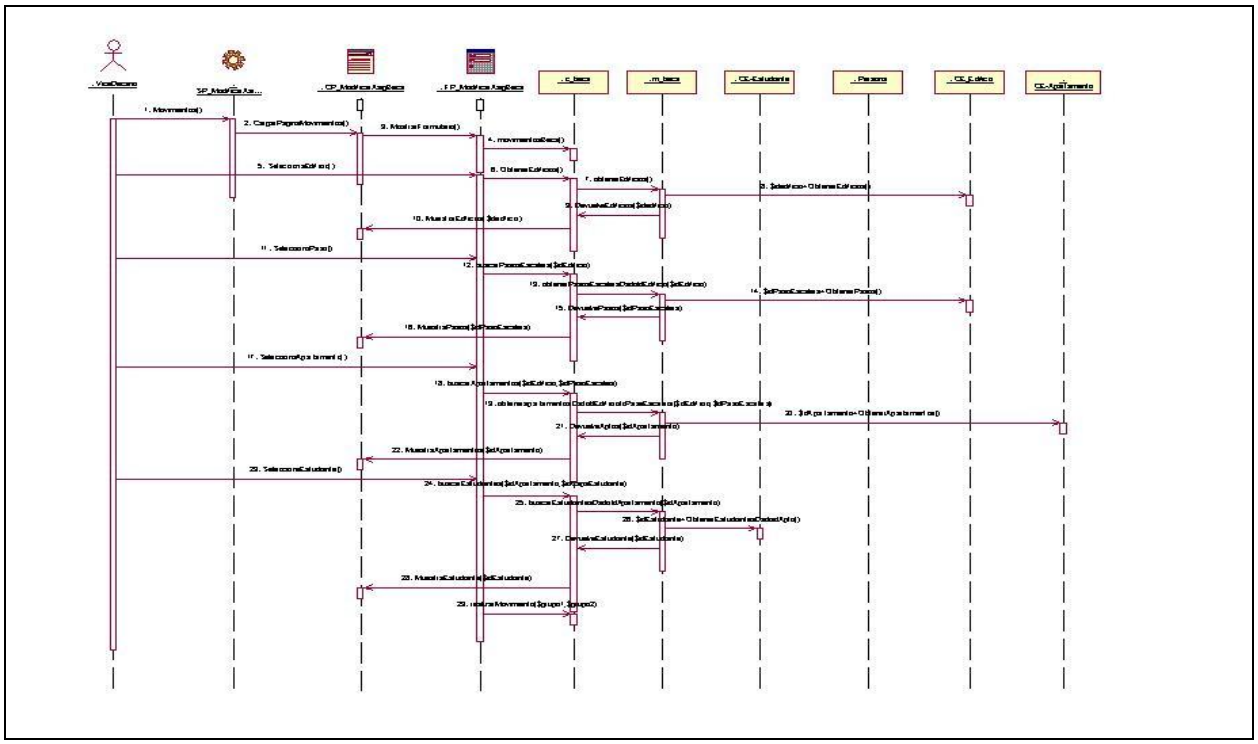
Autenticar Usuario



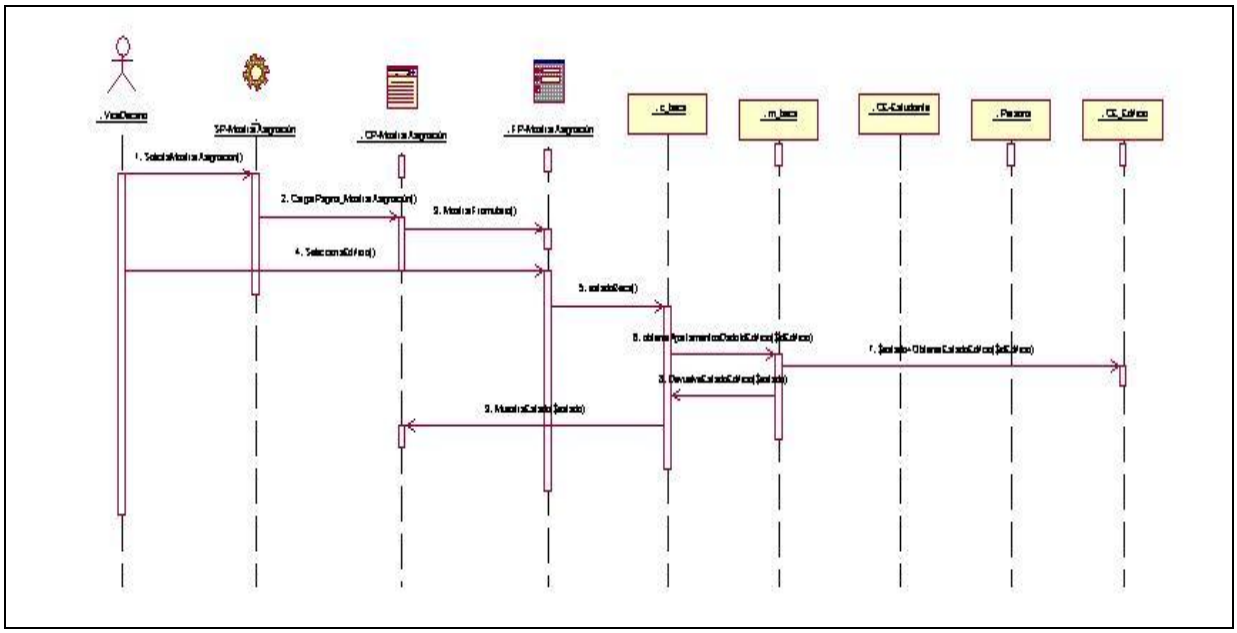
Asignar Beca



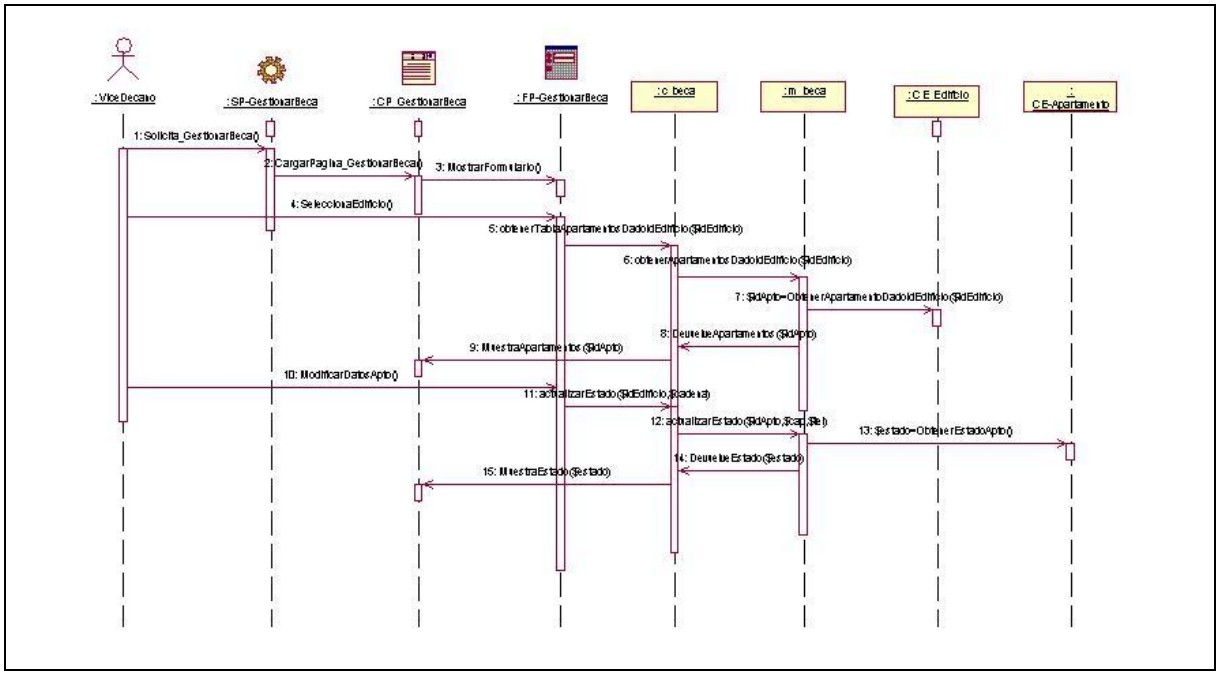
Modificar Asignación de Beca



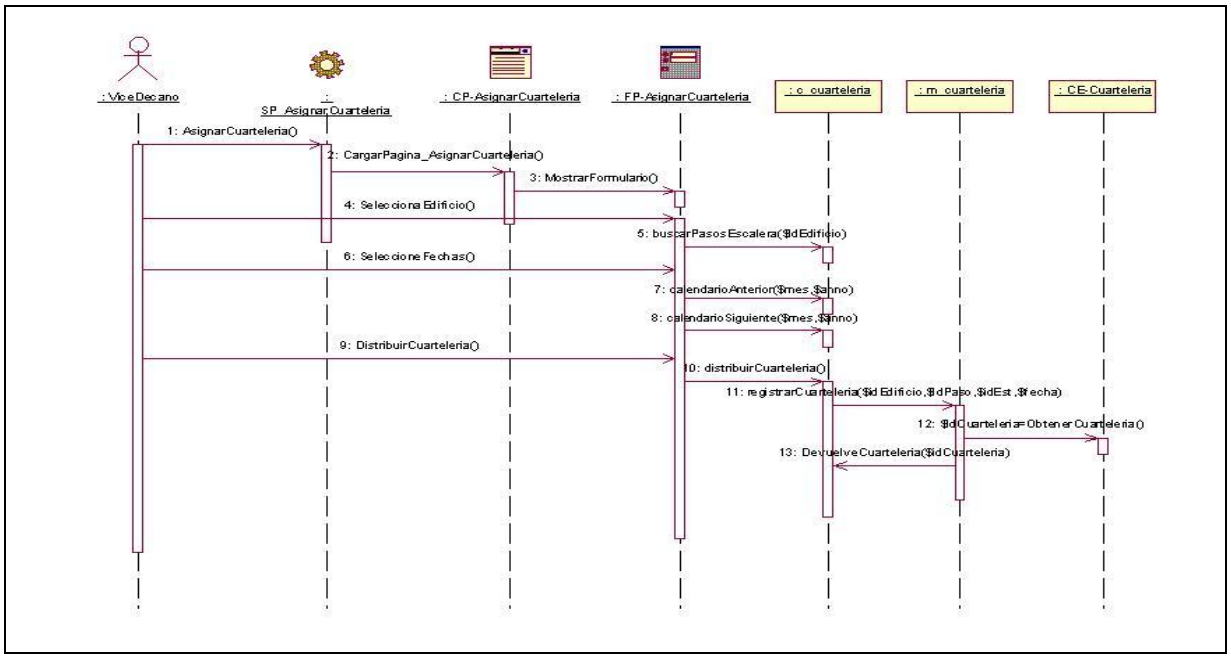
Mostar Asignación de Beca



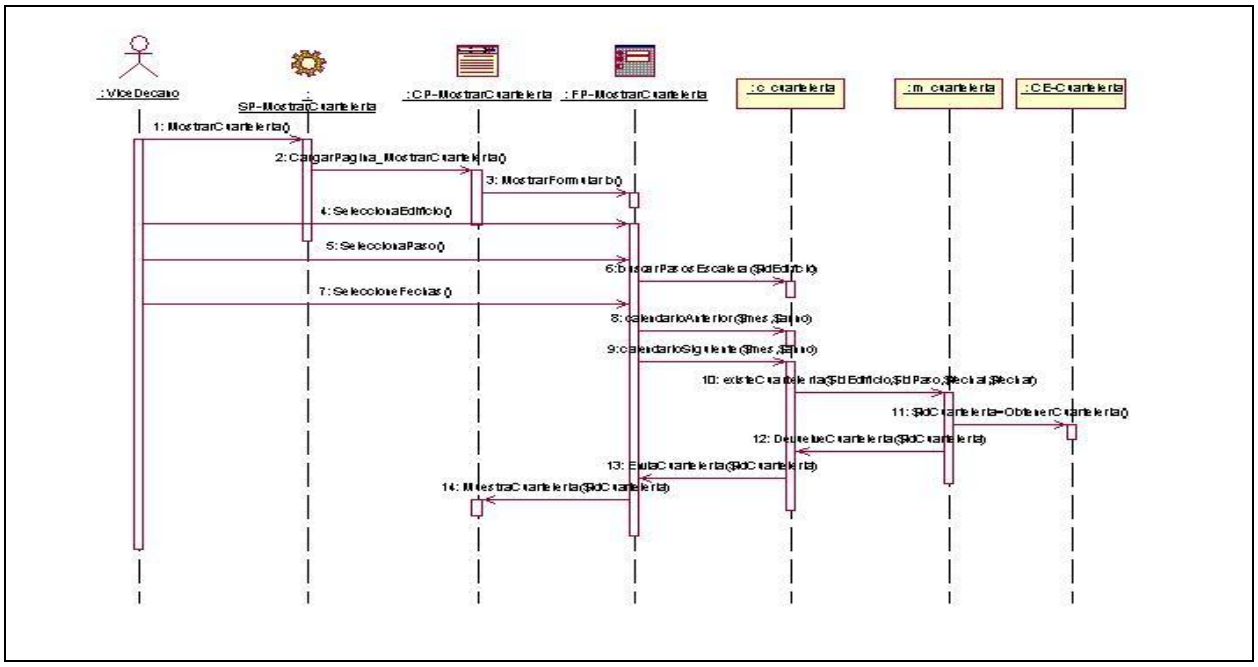
Gestionar Beca (Modificación)



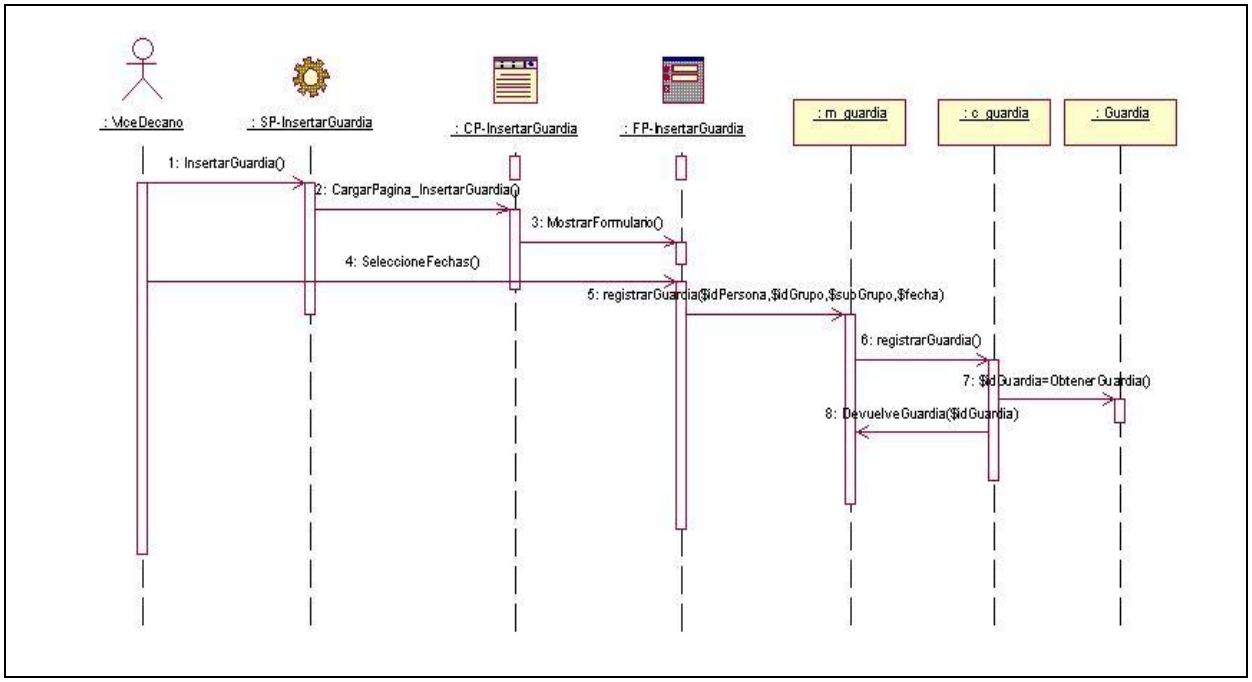
Asignar Cuartelería



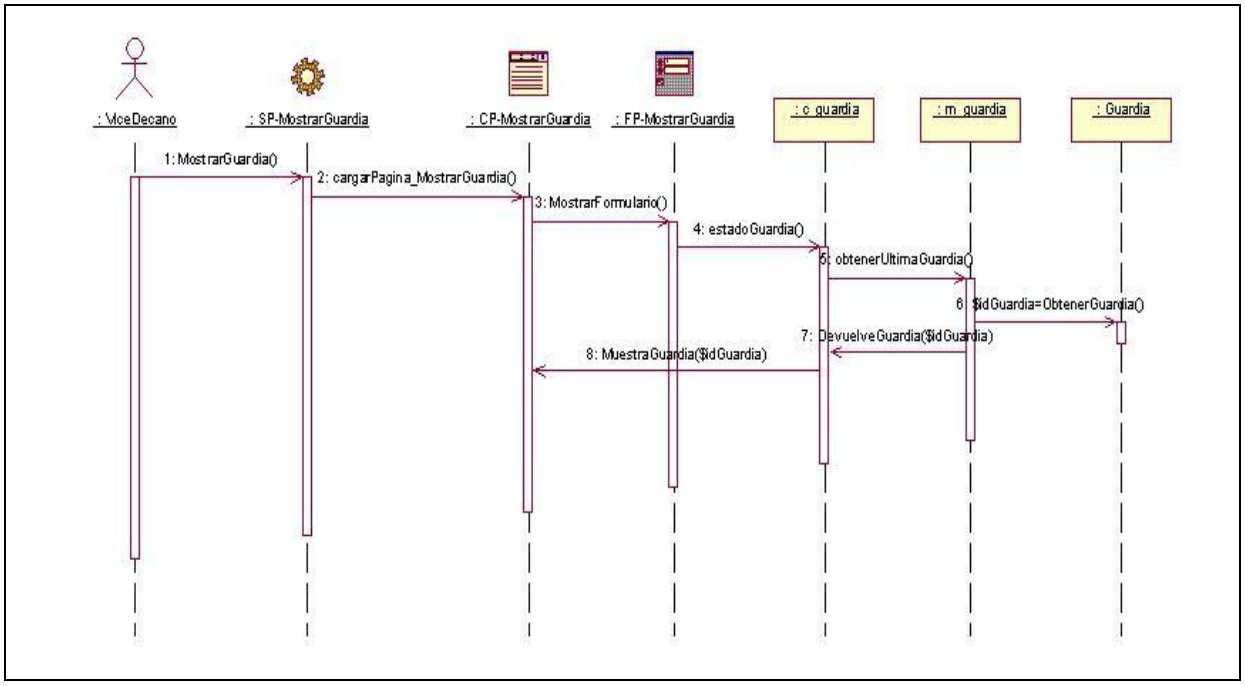
Mostrar Cuartelería



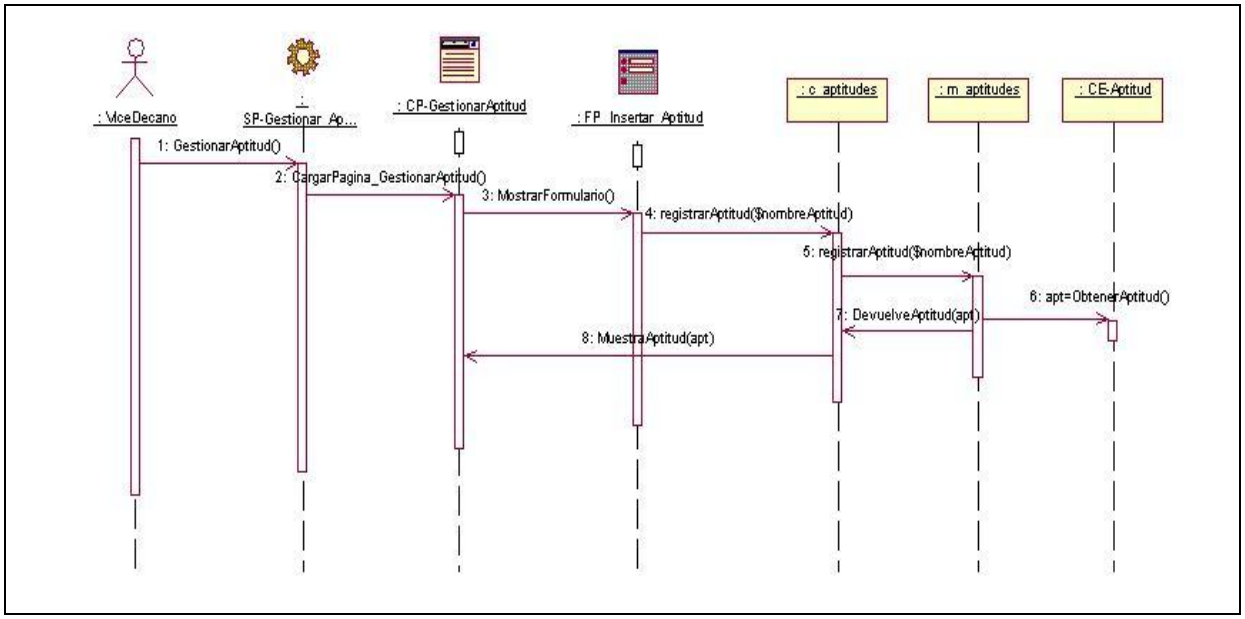
Asignar Guardia



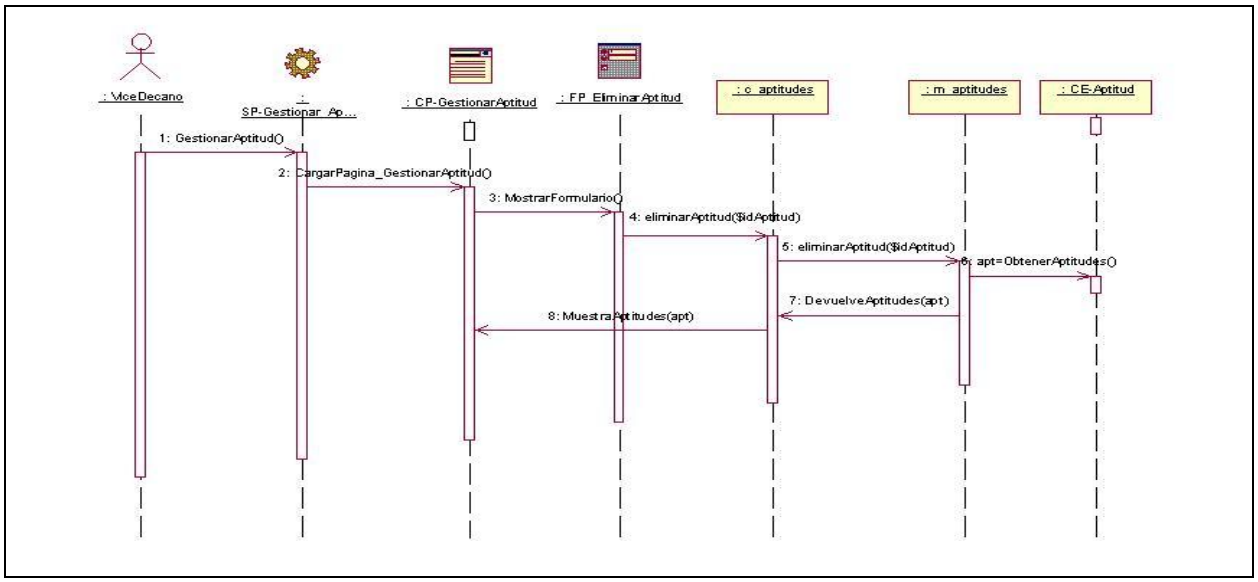
Mostrar Guardia



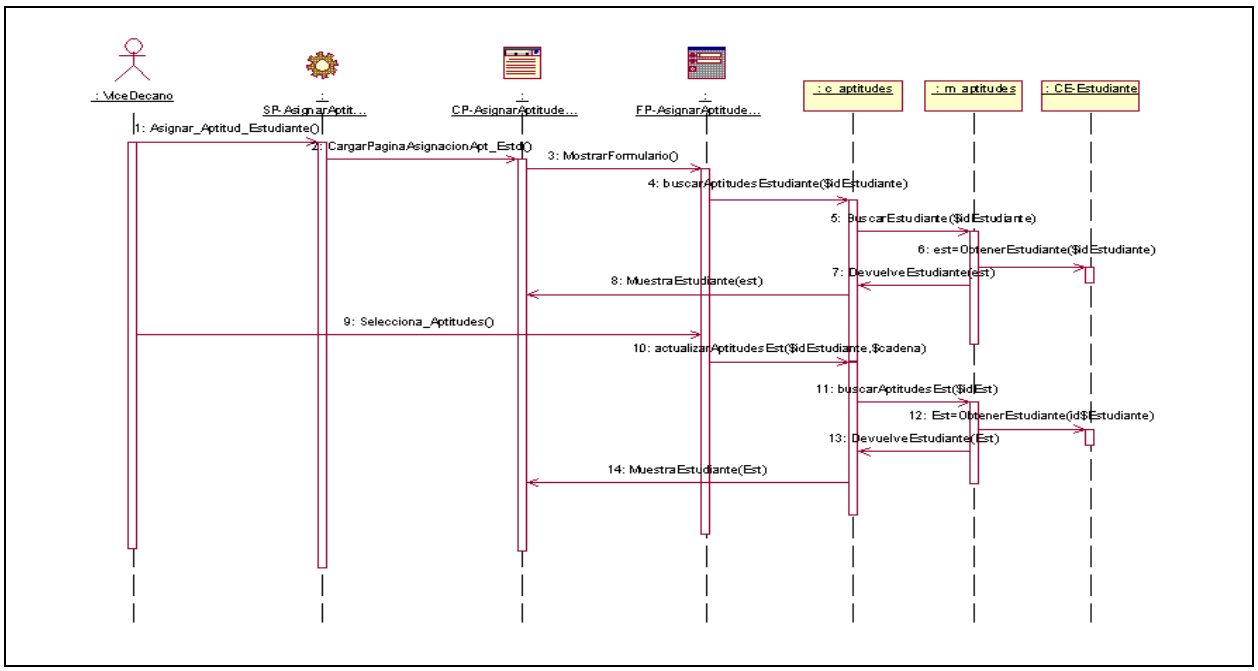
Insertar Aptitud



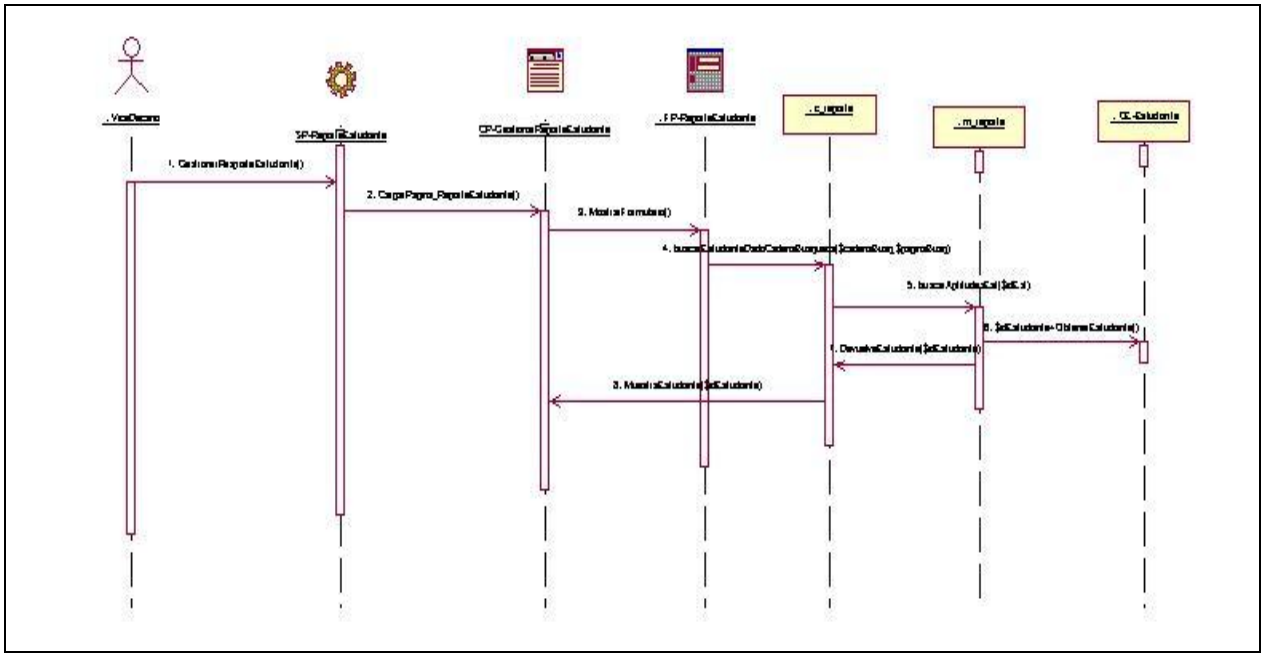
Eliminar Aptitud



Asignar Aptitud a un Estudiante



Reporte Estudiante



GLOSARIO

AppServ: Es una herramienta Open Source para Windows que facilita la instalación de Apache, MySQL y PHP en una sola herramienta.

DOM (Document Object Model): es esencialmente un modelo computacional a través de la cual los programas y scripts pueden acceder y modificar dinámicamente el contenido, estructura y estilo de los documentos HTML y XML.

Framework: En el desarrollo de software, es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

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

HTML: es el acrónimo inglés de HyperText Markup Language, que se traduce al español como Lenguaje de Etiquetas de Hipertexto1 . Es un lenguaje de marcado diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas Web.

jQuery: es un nuevo tipo de bibliotecas de Javascript que permite simplificar la manera de interactuar con los documentos HTML, permitiendo manejar eventos,desarrollar animaciones, y agregar interacción con la tecnología AJAX a nuestras páginas Web.

Scripts: guión o conjunto de instrucciones. Permiten la automatización de tareas creando pequeñas utilidades.

Administrador: es la persona que tiene privilegios para determinadas funcionalidades del sistema.

AJAX: Asynchronous JavaScript And XML.

Requisito Funcional: Requisito que especifica una acción que debe ser capaz de realizar el sistema, sin considerar restricciones físicas. Requisito que especifica comportamiento de entrada/salida de un sistema.

Requisito No Funcional: Requisito que especifica propiedades del sistema, como restricciones del entorno o de implementación, rendimiento, dependencias de la plataforma, mantenibilidad, extensibilidad o fiabilidad. Requisito que especifica restricciones físicas sobre un requisito funcional.

Caso de uso: es una operación/tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso.

Clase: Elemento de software que describe o caracteriza una entidad del mundo real o un entidad del espacio de implementación.