Universidad de las Ciencias Informáticas

Facultad 8

Sistema de Gestión de Información de la Facultad 8. Módulo para la Gestión de la Residencia Estudiantil Versión 2

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Yuniesky Dávila Pérez

Yadrian Martínez Padrón

Tutor: Ing. Yobannys Cabrera González

Co-tutores: Ing. Fidel Alberto Curbelo Rosell

Ing. Dayron Cruz Iñigo

"Ciudad de la Habana. Junio, 2008"

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de la presente tesis y reconozco 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 _____.

_____
Firma del Autor
Yuniesky Dávila Pérez

_____
Firma del Autor
Yadrian Martínez Padrón

_____
Firma del Tutor
Ing. Yobannys Cabrera González

"La verdadera medida de nuestra valía se compone de todos los beneficios que los demás han obtenido de nuestro éxito. "

CULLEN HIGHTOWER

Agradecimientos

Yuniesky:
*A mis **padres** por haberme dado todo el apoyo incondicional, por haber confiado en mí, y por ser las personas que más amo en el mundo.*
*A **Dayron, Osiris** y **Yudita** que lejos de ser mis mejores amigos los considero mis hermanos, gracias por tanto apoyo y comprensión.*
A mis compañeros de aula por soportarme durante tantos años.
*A mi compañero de tesis **Yadrian**, a mis tutores **Yobannys, Dayron** y **Fidel**.*
A mi familia en general que siempre me estuvo apoyando en todo momento.
*A mi abuela **Bautista**, por sus consejos, y guiarme siempre por el buen camino.*
Quiero agradecer a todos los que de una forma u otra aportaron su granito de arena por estar aquí donde estoy.
A todos Muchas Gracias….

Yadrian:
Quiero agradecer a mi madre, mi padre, mi hermana, y el resto de mi familia que en todo momento de mi carrera han estado a mi lado y me han apoyado en todo.
A todos los profesores que han contribuido en mi formación.
A la Revolución y a nuestro Comandante en Jefe por haberme dado la oportunidad de estudiar en esta maravillosa escuela.
A mis amigos Robin, Denis, José A. Iglesias (Candelita), Yander, Alfredo, Abelito, Yulio, Edgar, Arnay, Reinier, Fidelito, Andy y Yoan.
A mi compañero de tesis Yuniesky, y a mis tutores Yobannys, Dayron y Fidel.
A Humberto, David, Rider, Dixon, Everaldo, Lázaro, Robert, Yasmani, Héctor, Glemnys, Yulieski y a todos los que de una forma u otra me han ayudado y han aclarado mis dudas.
De forma general quiero agradecer a todas las personas que hicieron posible la culminación de este trabajo, y de esta forma mi sueño de ser universitario.

A todos Muchas Gracias.

Dedicatoria

Yuniesky:

A mi familia en general y muy especial a mis padres.

Y a ti mi abuelita que se que disfrutas este éxito más que nadie, y que estás muy orgullosa de mi.

Yadrian:

A toda mi familia, muy especialmente a mis padres, mi hermana y también a mis abuelos Eloína, Rubén, Ode y Francisco.

A todos aquellos que siempre confiaron en mí y que me brindaron su apoyo constante en todo momento.

Resumen

En el presente trabajo se pretende realizar el análisis, diseño e implementación de nuevas funcionalidades que le serán agregadas al Módulo para la Gestión de la Residencia Estudiantil de la Facultad 8 de la Universidad de las Ciencias Informáticas (UCI). Esto con el objetivo de proporcionarle a la dirección de la facultad la posibilidad de tener registrados en algún servidor de base de datos todo lo referido a los procesos de cuartelerías, visitas a apartamentos, trabajos socialmente útiles y guardias estudiantiles, lo cual hace más rápida y eficiente la búsqueda de una información determinada sobre alguno de ellos. Dentro de las nuevas funcionalidades que se desean implementar está la de conformar un nuevo módulo para la seguridad, donde el acceso de los usuarios a los contenidos esté definido por roles.

Primero se analiza como es el funcionamiento de las actividades que se desarrollan en el área de residencia de la universidad, además se realiza un estudio exhaustivo de los principales modelos, metodologías y estándares para el desarrollo de software, así como de las tendencias y tecnologías actuales, permitiendo este análisis seleccionar la metodología que apoya la solución del problema y las herramientas de desarrollo a emplear en la producción del software.

Luego se exponen las características del sistema que se propone, además de cómo está construido el mismo. Una vez concluido el sistema, este contribuirá a mejorar el tratamiento de la información y a su vez la disponibilidad de la misma.

Palabras claves: Módulo para la Gestión de la Residencia Estudiantil de la Facultad 8, cuartelerías, visitas a apartamentos, trabajos socialmente útiles, guardias estudiantiles, área de residencia de la universidad.

ÍNDICE

INTRODUCCIÓN

En la actualidad con el avance de la informática y las comunicaciones en Cuba y el auge que ha tomado la informatización de las instituciones cubanas, se desarrollan sistemas para la gestión de información de cada una de las mismas, por lo que se ha hecho necesario que nuestros centros educacionales apliquen esas nuevas técnicas para mejorar su funcionamiento, en cuanto a calidad y rapidez a la hora de procesar la información que manejan.

La Universidad de las Ciencias Informáticas (UCI), es uno de los centros universitarios más grandes de nuestro país, que cuenta con 10 Facultades y estas a su vez con una gran matrícula de estudiantes, profesores y trabajadores, por lo que se necesita en todo momento tener un control sobre cada uno de ellos, así como de las actividades en las que están envueltos a diario.

La facultad 8 de la UCI no está exenta de ello, ya que posee una matrícula de alrededor de 1000 estudiantes y poco más de 100 profesores y trabajadores que conviven a diario en la universidad, de ahí la necesidad de tener un control sobre cada una de las actividades que tienen lugar en áreas como la residencia estudiantil. Esto mantendría una mayor organización sobre todo lo que allí acontece a diario, como es el caso de: guardias estudiantiles, cuartelerías, visitas a los apartamentos, control de la limpieza y organización de los mismos, entre otras. Además de algunas actividades que no necesariamente se realizan en la residencia estudiantil, como es el caso del trabajo socialmente útil.

Todas las actividades antes mencionadas tienen una gran importancia en la formación integral de los estudiantes en la cual están implicados de alguna manera los profesores y trabajadores que conforman la facultad 8 de la UCI, por lo que en la medida que se tenga un mayor control sobre cómo se llevan a cabo cada una de estas, se tendrá también una visión de cuál ha sido en general el comportamiento de cada estudiante fuera del marco docente.

Al realizar una investigación y observación de los procesos que tienen lugar en la residencia estudiantil de la facultad 8 se han identificado una serie de problemas que persisten, aun cuando la facultad cuenta con un sistema para la gestión de algunos procesos que tienen lugar en dicha área, como por ejemplo:

- Mucha información todavía suele ser almacenada en *Microsoft Word* y *Excel*, que no son eficientes a la hora de realizar búsquedas avanzadas. En muchos casos, el control de las actividades se realizan en formato duro, es decir: libro, hojas etc. lo que puede traer consigo la pérdida de información.

- El volumen de información es cada vez mayor, lo cual provoca que la búsqueda de la misma en determinado momento se torne un poco lenta.
- No existe la posibilidad de que varios usuarios puedan acceder a la información almacenada simultáneamente, lo que provoca en muchas ocasiones un atraso a la hora de contar con la información necesaria en tiempo.
- La información no se encuentra totalmente centralizada, es decir, aun existen personas que envían informaciones por correo electrónico o la hacen llegar por medio de algún dispositivo de almacenamiento, lo cual puede provocar que dicha información sea consultada por personas no autorizadas para ello.

Después de realizar un análisis exhaustivo de los procesos que tienen lugar en la residencia estudiantil de la facultad 8 de la UCI se llega a la conclusión de que sería factible automatizar todos aquellos procesos relacionados con esta área que no fueron implementados en el sistema que posee la facultad actualmente.

De todo lo anteriormente expresado se deriva como **objeto de estudio** los procesos que tienen lugar en la residencia estudiantil de la Facultad 8 de la UCI.

De ello se deriva el **campo de acción** que comprende este trabajo, que son los procesos relativos a los partes de la guardia estudiantil, las cuartelerías, las visitas a los apartamentos, los trabajos socialmente útil (TSU) que realizan las brigadas, entre otros.

Por esta razón se plantea para este trabajo como **Idea a defender** que, si se realiza el análisis, diseño e implementación de nuevas funcionalidades al módulo de gestión de la residencia estudiantil, perteneciente al Sistema de Gestión de Información de la Facultad 8(SGIF), con relación a los procesos antes mencionados, se piensa que los mismos, se realizarán con mayor organización y rapidez.

Como **objetivo general** se tiene: Análisis, diseño e implementación de nuevas funcionalidades al módulo de gestión de la residencia estudiantil de la facultad.

Se plantean como **objetivos específicos**:

- Investigar todo lo relacionado a los procesos que tienen lugar en la residencia estudiantil.
- Informatizar los procesos relacionados con las cuartelerías, partes de la guardia, visita a los apartamentos y los TSU.
- Implementar un módulo para la seguridad.
- Conformar la documentación del sistema, con vista al desarrollo de posteriores versiones.

Para dar cumplimiento a estos objetivos se realizarán las siguientes **tareas**:

- Entrevistar al Vicedecano de Residencia y Extensión de la facultad, además de otras personas vinculadas a las actividades que tienen lugar en esta área, para obtener una mayor información sobre los procesos que allí se realizan.
- Realizar un estudio de las herramientas y tecnologías posibles a utilizar para la informatización de las nuevas funcionalidades.
- Determinar los requisitos que debe cumplir el sistema.
- Realizar el análisis, diseño e implementación de nuevas funcionalidades al módulo de gestión de la residencia estudiantil que existen en la facultad actualmente.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción.

En el presente capítulo se realizará un estudio de todo lo referente al estado del arte del tema a desarrollar, así como las tendencias, tecnologías, métodos y herramientas que serán utilizadas en el desarrollo de este proyecto.

1.2 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, además puede ser utilizado en múltiples plataformas.

1.3 Tendencias y tecnologías actuales.

1.3.1 Lenguajes de programación para la Web.

PHP (Personal Home Page).

PHP es un lenguaje de script interpretado en el lado del servidor utilizado para la generación de páginas Web dinámicas, similar al ASP de *Microsoft* o el JSP de Sun, embebido en páginas HTML y ejecutado en el servidor. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas de sí mismo. La meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas. [1]

Ventajas de PHP.

- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL
- 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. [1]

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, la 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 [2].

Ventajas del PERL.

- Construcción de pequeños programas que pueden ser usados como filtros para obtener información de ficheros y 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 intérprete PERL correspondiente a cada sistema operativo.
- Es uno de los lenguajes más 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. [2]

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).

Ventajas de ASP.NET.

- Mejor rendimiento.
- Compatibilidad con herramientas de primer nivel.
- Eficacia y flexibilidad.
- Simplicidad.
- Facilidad de uso.
- Escalabilidad y disponibilidad.
- Posibilidad de personalización y extensibilidad.
- Seguridad. [3]

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. [4].

Ventajas de Java.

- Es multiplataforma, pues para ejecutar un programa basta compilarlo una sola vez: a partir de entonces, se puede hacer correr en cualquier máquina que tenga implementado un intérprete de Java.
- Orientado a objetos.
- Dispone de un mecanismo conocido como "recogida de basura", el cual —usando la capacidad multitarea de Java— hace que, durante la ejecución de un programa, los objetos que ya no se utilizan se eliminen automáticamente de la memoria.
- Posee *applets*, que son pequeños programas en Java que se cargan junto con una página web desde un servidor, y que son ejecutados como una parte de la página web, estos se sirven del propio código del navegador en cuya máquina

virtual se ejecutan, utilizándolo para tareas tales como presentación gráfica o comunicaciones.

- Es un lenguaje de gran facilidad de aprendizaje, pues en su concepción se eliminaron todos aquellos elementos que no se consideraron absolutamente necesarios. Por ejemplo, en comparación con otros lenguajes como C ó C++, es notable la ausencia de punteros.[4]

Fundamentación del lenguaje a escoger.

Tabla 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. Perl presenta como desventaja 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 está disponible bajo la licencia GPL.

1.3.2 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, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. [5]

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. [5]

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.

1.3.3 Metodología.

Metodología, del griego (metà "más allá" 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. [7]

Metodología: Conjunto de métodos empleados para el desarrollo de sistemas automatizados. [8]

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.[8]

Metodología orientada a objetos.

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.[6]

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. [6]

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).

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.
El UML cuenta con varios tipos de modelos, los cuales muestran diferentes aspectos de las entidades representadas [16].

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 [16]

Microsoft Solution Framework (MSF).

MSF en una metodología de *software* flexible, se puede adaptar a proyectos de distintos tamaños y complejidades [17].

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 [17].

MSF tiene las siguientes **características**: [18]

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: [17]

- **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.

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 [18].

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 [18].

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.

1.3.4 Sistemas Gestores 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 Datos: 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, consultas.
- ✓ No hay duplicidad de información, comprobación de información en el momento de introducir la misma.
- ✓ Integridad referencial al terminar los registros.[12]

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 en un 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.[12]

PostgreSQL.

PostgreSQL es el servidor de base de datos relacional orientada a objetos de software libre más potente que existe, liberado bajo la licencia BSD. [10]

Soporta:

- *Querys* complejos, incluyendo *subselects*.
- Integridad referencial *(Foreign Keys*).
- *Triggers*.
- Vistas (*Views*).
- Integridad Transaccional (ACID).
- Control de versionado concurrente (MVCC)[10]

Ventajas de PostgreSQL.

- ✓ 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 [13].

Desventajas de PostgreSQL:

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

MySQL

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. Su diseño multihilo le permite soportar una gran carga de forma muy eficiente. [19]

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 [19].

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.

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 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.

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

CakePHP.

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* [20].

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 [21].

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 [22].

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 [21]

Selección del framework a utilizar.

Después de analizar las características particulares de cada uno de los *frameworks* antes mencionados se llegó a la conclusión que los tres cumplían con los requisitos necesarios para el desarrollo de la aplicación, se selecciona *CodeIgniter* como frameworks para la aplicación pues permite trabajar remoto, y cuenta con una buena documentación.

En Symfony Project la mayoría de las tareas, como la de paginación, son mucho más complicados y la simplicidad definitivamente no es una cualidad de este marco.

1.3.6 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.

1.3.7 Otras herramientas a utilizar.

Macromedia Dreamweaver 8 es un *software* fácil de usar que permite crear páginas web profesionales. [14]

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. [14]

Se puede crear tablas, editar marcos, trabajar con capas, insertar comportamientos JavaScript, etc., de una forma muy sencilla y visual. [14]

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. [14]

Dreamweaver es compatible con todas las principales tecnologías de servidor como, por ejemplo, *ColdFusion*, PHP, ASP, ASP.NET y JSP. [15]

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.[11]

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 [11].

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.

1.4 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 visitas a apartamentos, cuartelerías, todo lo referente a los trabajos socialmente útil, 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 implementados 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 la residencia estudiantil.

Para efectuar el diseño de dicho sistema, se hace uso del Proceso Unificado Racional que hace uso del 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 4.6.2*.
- Sistema gestor de base de datos: PostgreSQL (versión 8.2).
- Herramienta Diseño Web: Macromedia *Dreamweaver* 8.
- *Framework*: CodeIgniter.
- Herramienta *Case: Rational Rose.*
- Servidor Web: Apache 2.2

CAPÍTULO 2 CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

En el presente capítulo se enuncia y describe los procesos de negocio, se exponen las reglas que debe cumplir el negocio en cuestión, se dan a conocer cuáles serán los procesos a automatizar, además de describirse los actores y trabajadores que intervienen en el mismo, así como la realización de los diagramas de casos de uso del negocio, diagramas de actividades y el modelo de objeto. También se identifican cuáles son los requisitos funcionales y no funcionales que debe cumplir la aplicación. Se muestra además una descripción detallada de cada uno de los casos de usos del sistema.

2.2 Descripción de los procesos de negocio.

A continuación se muestra una descripción de los procesos de negocio por separado, que contribuirá a un mejor entendimiento de los mismos.

2.2.1 Cuartelerías.

Las cuartelerías son realizadas por los estudiantes, de lunes a sábado se encuentra en cada paso de escalera de su respectivo edificio un estudiante de cuartelero, estos tienen la responsabilidad de tener un control de todo el personal ajeno que accede al edificio, así como la realización y mantenimiento de la limpieza del paso de escalera y las áreas de los alrededores del edificio.

A diario se emite una evaluación (B, R, M) de cada una de las cuartelerias realizadas, contribuyendo así a la evaluación integral de cada estudiante.

2.2.2 Guardia estudiantil.

En la facultad existe una planificación para la guardia estudiantil, donde se refleja el grupo de estudiantes (brigada) y el profesor responsable, además del día en que le corresponde realizar la guardia.

Una vez realizada la guardia se confecciona un reporte con los datos pertenecientes a la guardia estudiantil, donde quedará reflejado el profesor encargado de controlar la guardia, la brigada y de esta los estudiantes que asistieron.

2.2.3 Trabajo socialmente útil.

El trabajo socialmente útil surge debido al poco personal existente en la universidad que contribuya en el embellecimiento y mantenimiento de las áreas verdes, así como la limpieza de otros lugares como son los docentes, viales etc.

Es por ello que todas las facultades mediante los TSU brinda su apoyo a esta tarea, en la facultad 8 se realizan en dos secciones (mañana y tarde), en cada una se encuentra una brigada para el desarrollo de dicha tarea. Una vez realizado el TSU se emite un reporte con los datos del mismo, donde se da a conocer la cantidad de estudiantes que asistieron, el lugar y una evaluación.

2.2.4 Visita a los apartamentos.

Las visitas a los apartamentos son realizadas por profesores o dirigentes de la facultad, para evaluar una serie de aspectos que se deben cumplir en el área de la residencia estudiantil, como son: limpieza y organización de los apartamentos, cuidado de los medios básicos, ahorro de energía, así como la educación formal de los estudiantes que viven allí, una vez realizada la visita se emite una evaluación de los aspectos antes mencionados, así como una evaluación general del apartamento.

2.3 Reglas de Negocio.

2.3.1 Reglas de restricción.

- En un día, realizan el TSU 2 brigadas una en cada sección.
- La guardia estudiantil es realizada por una brigada solamente en un día.
- Un estudiante realiza la guardia en una sola posta ese día.
- Un solo estudiante esta de cuartelero en un paso de escalera.

2.3.2 Reglas de derivación.

- Cuando un estudiante no se presenta a realizar la guardia, TSU, Cuartelería se dice que este incumplió.
- Cuando un estudiante abandona la guardia, TSU, cuartelería antes de culminar la misma se convierte en ausente.
- Cuando un estudiante no está en ninguno de los casos anteriores, se dice que este cumplió con la guardia, TSU, cuartelería.

2.4 Identificar los actores y trabajadores del negocio y describir CU negocio.

2.4.1 Actores del negocio.

Actores del negocio	Justificación
Vicedecano de Extensión y Residencia (VDER)	Es uno de los directivos de la Facultad que solicita informaciones relativas a los procesos que se realizan en la residencia estudiantil.

2.4.2 Trabajadores del negocio.

Trabajadores del negocio	Justificación
Secretaria del Vicedecano de Extensión y Residencia.	Persona que asiste al Vicedecano de Extensión y Residencia. Es la encargada de conformar todas las solicitudes.

2.4.3 Descripción de CU Negocio.

"Registrar Trabajo Socialmente Útil (TSU)"

Caso de Uso	**Registrar Trabajo Socialmente Útil (TSU)"**
Actores	Vicedecano de Residencia (inicia).
Propósito	Agregar información perteneciente a un TSU de una brigada determinada
Resumen	El CUN se inicia cuando el Vicedecano de Residencia le solicita a su secretaria registrar un TSU donde se reflejen las incidencias de una brigada en el mismo. La secretaria por su parte realiza dicha acción y termina el CUN.

Curso normal de los eventos	
Acción del actor	**Respuesta del Negocio**
1. El Vicedecano de Residencia solicita registrar TSU de las brigadas que lo están efectuando ese día.	1.1 La secretaria del Vicedecano busca los datos necesarios para realizar dicha solicitud.
	1.2 Una vez obtenido los datos registra el TSU.
	1.3 Emite un mensaje de que ha registrado el TSU.
2. El Vicedecano recibe mensaje de registro realizado y termina el CUN.	

"Registrar visitas a los apartamentos"

Caso de Uso	**Registrar visitas a los apartamentos**
Actores	Vicedecano de Residencia (inicia).
Propósito	Registrar la información perteneciente a la visita en los apartamentos de la residencia.
Resumen	El CUN se inicia cuando el Vicedecano de Residencia le solicita a su secretaria que registre las visitas a los apartamentos que fueron efectuadas. La secretaria por su parte pide los datos necesarios para realizar dicha acción y termina el CUN.

Curso normal de los eventos	
Acción del actor	**Respuesta del Negocio**
1. El Vicedecano de Residencia solicita registrar las visitas a los apartamentos.	1.1 La secretaria del Vicedecano le solicita los datos necesarios para realizar dicha solicitud.
2. El Vicedecano le entrega los datos.	2.1 La secretaria recibe los datos.

	2.2 La secretaria registra las visitas a los apartamentos.
	2.3 La secretaria envía mensaje de que ha realizado el registro de las visitas satisfactoriamente.
3. El Vicedecano recibe el mensaje de registro realizado y termina el CUN.	

"Registrar Cuartelería"

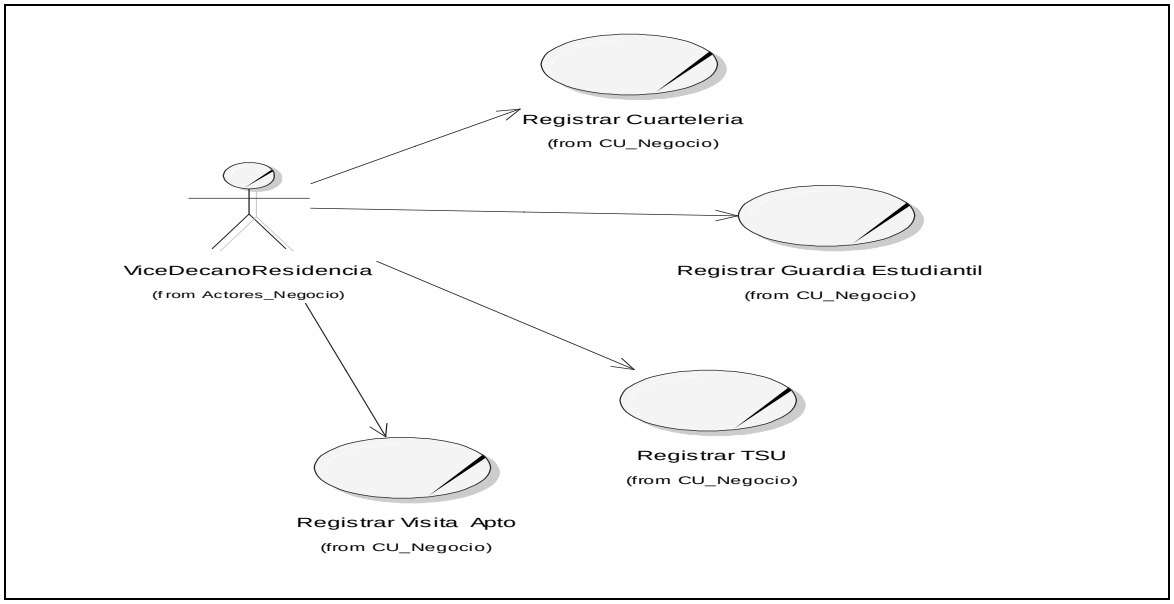
Caso de Uso	**Registrar Cuartelería**
Actores	Vicedecano de Residencia (inicia).
Propósito	Adicionar información perteneciente a las cuartelerías realizadas en la residencia estudiantil.
Resumen	El CUN se inicia cuando el Vicedecano de Residencia le solicita a su secretaria registrar las cuartelerías efectuadas. La secretaria por su parte pide los datos necesarios para realizar dicha solicitud, realiza el proceso y el CU termina.
Curso normal de los eventos	
Acción del actor	**Respuesta del Negocio**
1. El Vicedecano de Residencia solicita registrar cuartelerías.	La secretaria del Vicedecano le solicita la información necesaria para realizar dicha solicitud.
2. El Vicedecano le entrega los datos.	La secretaria recibe los datos y registra las cuartelerías.
	2.3 La secretaria envía un mensaje informando que ha realizado el proceso satisfactoriamente.
3. El Vicedecano recibe el mensaje y termina el CUN.	

"Registrar Guardia Estudiantil"

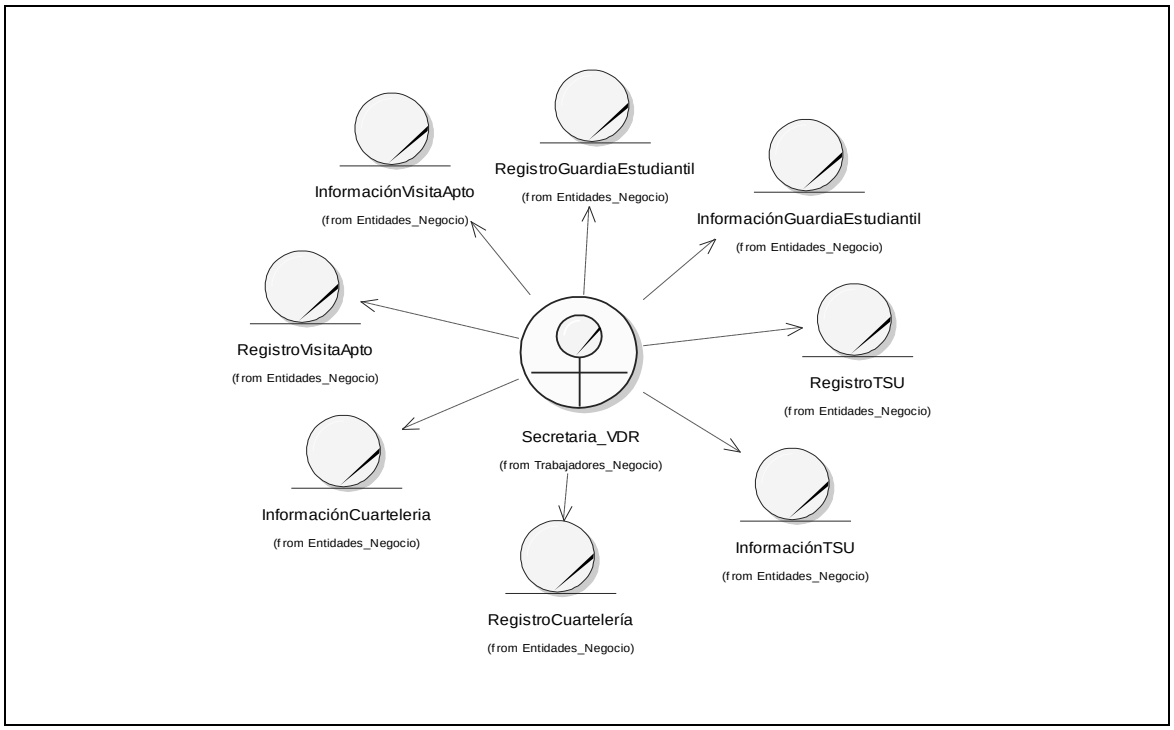
Caso de Uso	**Registrar Guardia Estudiantil**
Actores	Vicedecano de Residencia (inicia).
Propósito	Registrar la información perteneciente a la guardia estudiantil.
Resumen	El CUN se inicia cuando el Vicedecano de Residencia le solicita a su secretaria registrar la guardia estudiantil. La secretaria por su parte realiza solicita los datos necesarios para realizar dicha operación y el CU termina.
Curso normal de los eventos	
Acción del actor	**Respuesta del Negocio**
1. El Vicedecano de Residencia solicita registrar guardia estudiantil.	1.1 La secretaria del Vicedecano le solicita los datos necesarios para realizar dicha solicitud.

2. El Vicedecano le entrega los datos.	2.2 La secretaria recibe los datos y realiza la operación de registrar guardia estudiantil.
	2.3 La secretaria le informa al vicedecano que ha realizado la operación satisfactoriamente.
3. El Vicedecano recibe el mensaje y termina el CUN.	

2.5 Diagrama de CUN.

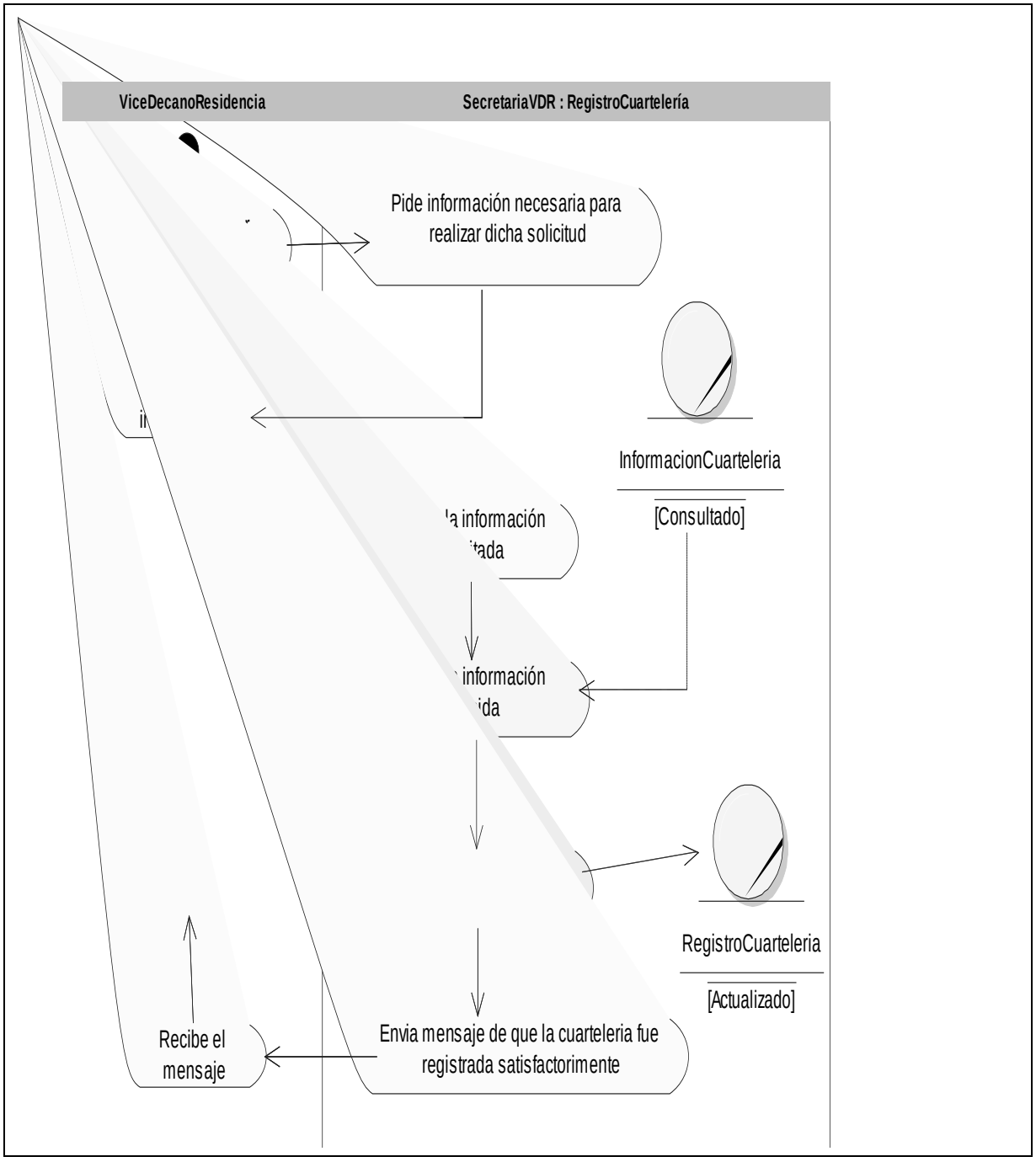


2.6 Diagrama de Objeto del Negocio.

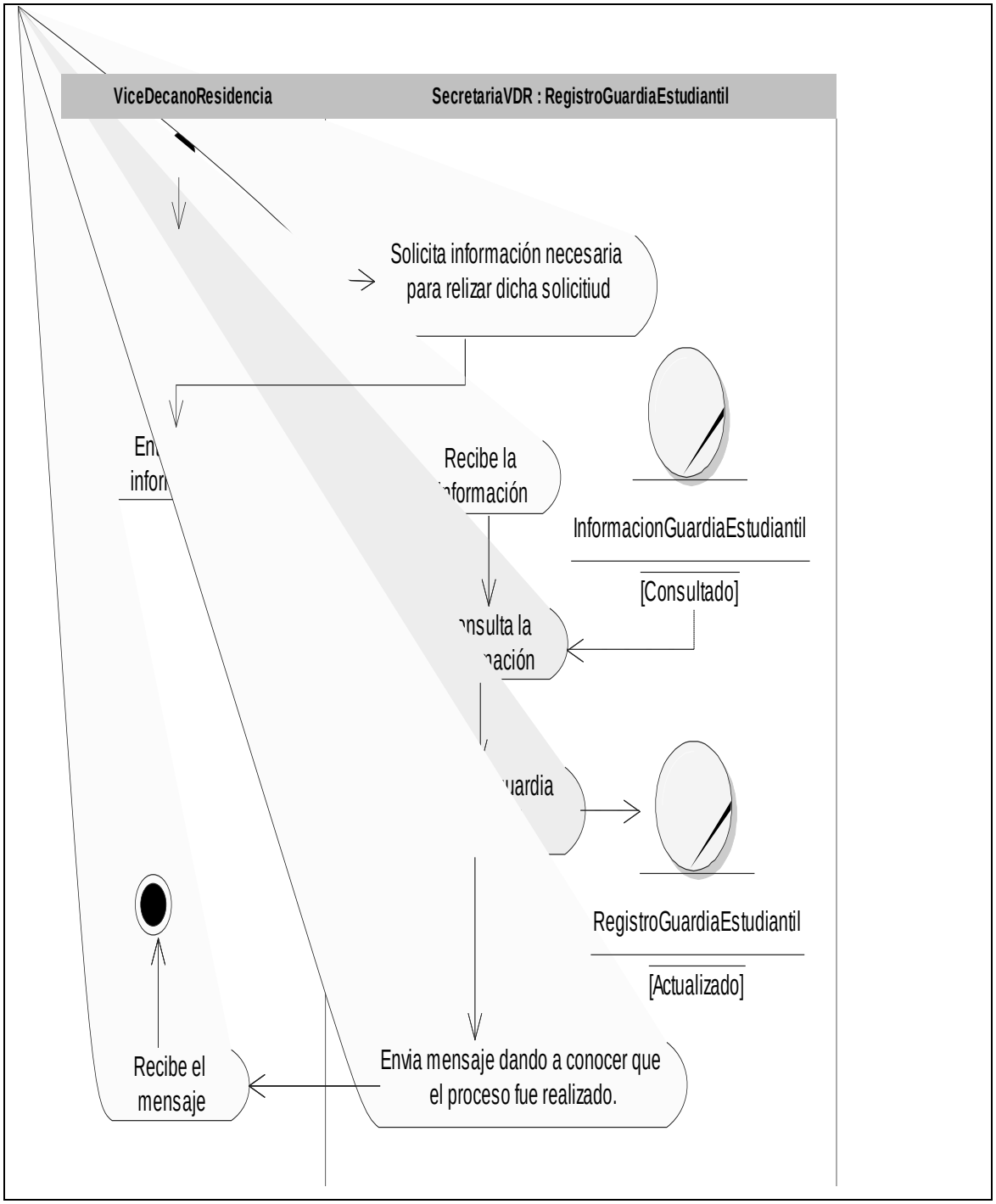


2.7 Diagrama de Actividades.

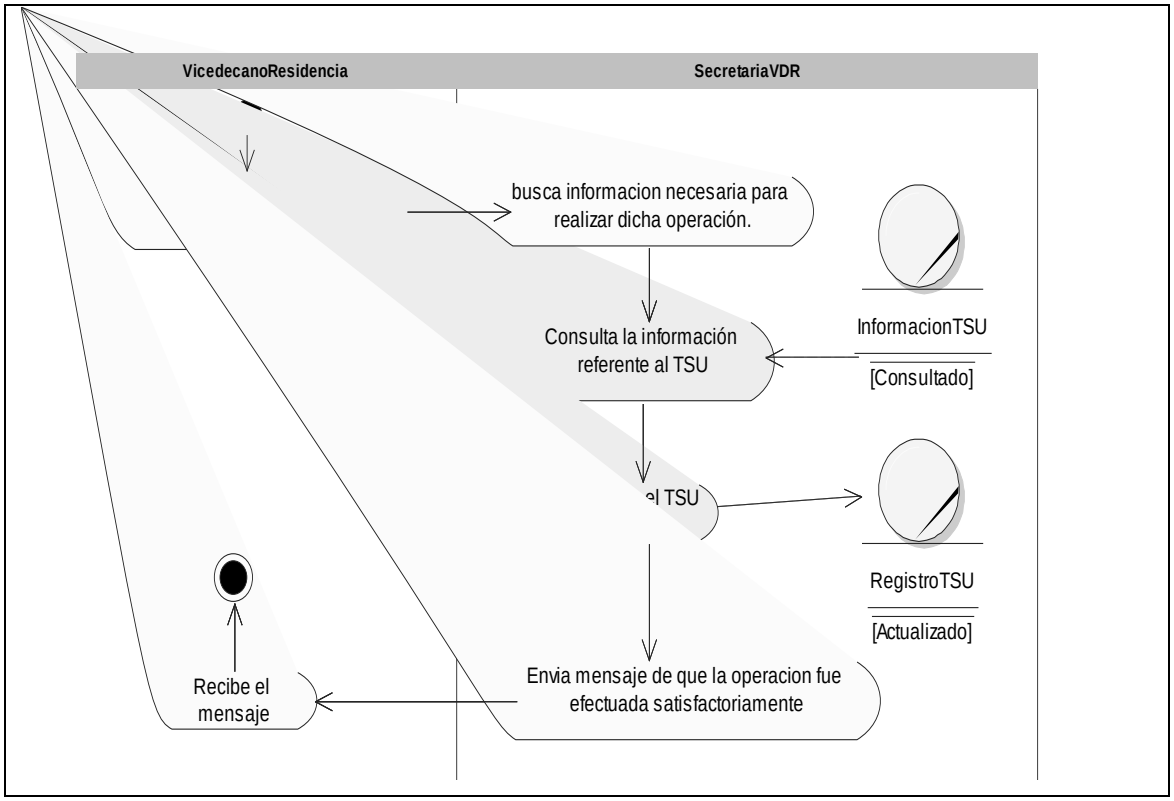
2.7.1 Diagrama de actividades CUN "Registrar de Cuartelería".



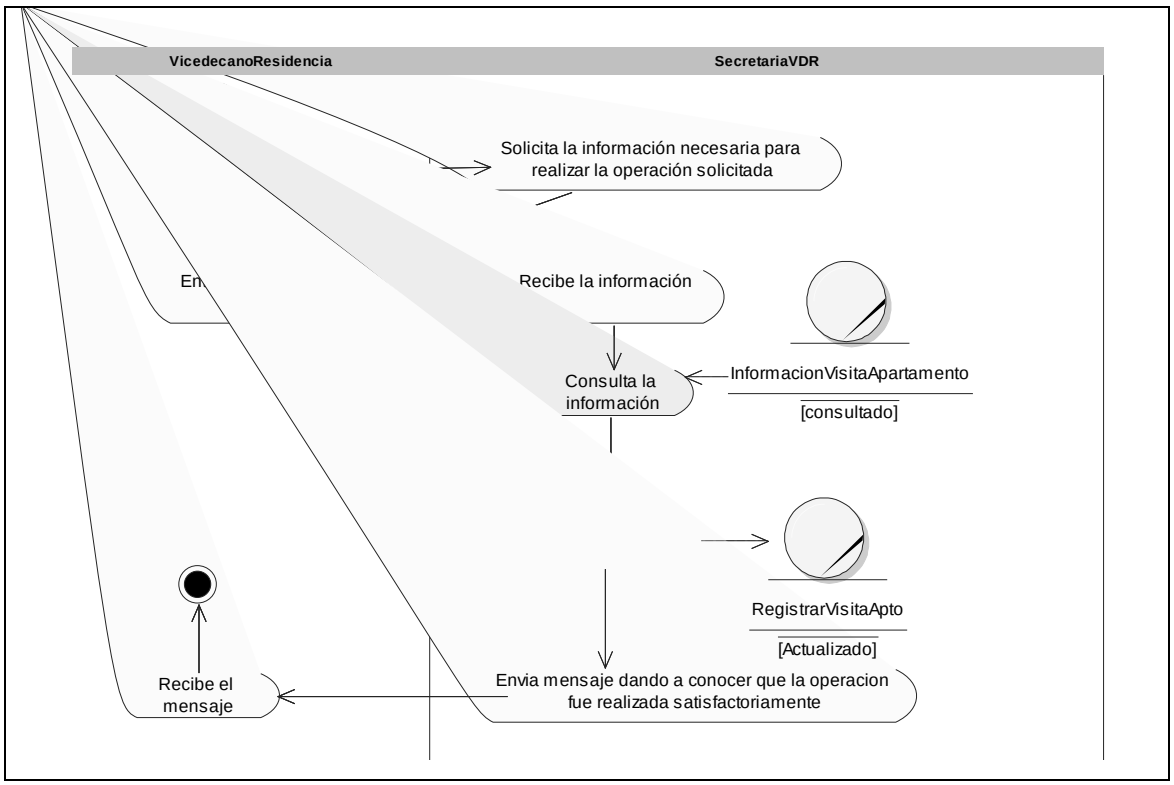
2.7.2 Diagrama de actividades CUN "Registrar Guardia Estudiantil".



2.7.3 Diagrama de actividades CUN "Registrar TSU".



2.7.4 Diagrama de actividades CUN "Registrar Visita a Apartamentos".



2.8 Requerimientos.

2.8.1 Requisitos funcionales

RF-1 Gestionar usuario.

a. Adicionar usuario.

b. Eliminar usuario.

RF-2 Buscar estudiantes

RF-3 Gestionar posta de guardia.

3.1 Adicionar posta.

3.2 Eliminar posta.

RF-4 Gestionar cuartelería.

4.1 Adicionar cuartelería.

4.2 Modificar cuartelería.

4.3 Eliminar cuartelería.

RF-5 Gestionar visita a apartamentos.

5.1 Adicionar visita a apartamento.

5.2 Modificar visita a apartamento.

5.3 Consultar visita a apartamento.

5.4 Eliminar visita a apartamento.

RF-6 Gestionar trabajo socialmente útil (TSU).

6.1 Adicionar TSU.

6.2 Modificar TSU.

6.3 Consultar TSU.

6.4 Eliminar TSU.

RF-7 Gestionar módulo.

7.1 Adicionar módulo.

7.2 Modificar módulo.

7.3 Eliminar módulo.

RF-8 Gestionar lugar trabajo socialmente útil.

8.1 Adicionar lugar TSU.

8.2 Eliminar lugar TSU.

RF-9 Actualizar brigada.

RF-10 Gestionar rol.

10.1 Adicionar rol.

10.2 Modificar rol.

10.3 Eliminar rol.

RF-11 Gestionar guardia estudiantil.

11.1 Adicional guardia estudiantil.

11.2 Modificar guardia estudiantil.

11.3 Consultar guardia estudiantil.

11.4 Eliminar guardia estudiantil.

RF-12 Importar datos.

2.8.2 Requisitos no funcionales.

A continuación se presentan los requisitos no funcionales:

Software

RNF 1. Servidor Web Apache v1.x o superior.

RNF 2. SGBD: PostgreSQL preferiblemente v7.x en adelante.

RNF 3. Navegador Internet Explorer v4.0 o superior.

Hardware

RNF 6. Tarjeta de red.

RNF 7. Para los servidores tanto Web como SGBD: PENTIUM II o superior con 256 MB de RAM o más.

RNF 8. Capacidad de disco duro en Gigabyte, preferiblemente mayor a 10 GB.

Apariencia o interfaz externa

RNF 9. La interfaz no debe contener muchas imágenes que demoren las respuestas al usuario.

RNF 10. El diseño de la interfaz debe ser sencillo y claro. Debe contener elementos visibles que identifiquen cada una de sus acciones.

RNF 11. La navegabilidad debe ser sencilla.

Usabilidad

RNF 12. El sistema podrá ser usado por cualquier persona que acceda a él que tenga algún conocimiento básico de computación y trabajo en la Web.

RNF 13. Rápido acceso de búsqueda de la información, en tiempos cortos.

Rendimiento

RNF 14. El sistema deberá ser capaz de gestionar toda la información y dar respuesta a las solicitudes lo más rápido posible.

RNF 15. Debe ser eficiente a la hora de gestionar las solicitudes logrando que sin mucha navegación por el sitio se obtenga los resultados deseados.

RNF 16. Debe estar disponible las 24 horas del día.

Soporte

RNF 17. El sistema debe ser de fácil instalación y configuración, con vista a poder darle un mantenimiento asequible en caso de fallos.

Portabilidad

RNF 18. Multiplataforma. El sistema se podrá montar sobre Unix, Linux, Windows, etc. Así mismo podrá usar una serie de SGBD como PostgreSQL, MySQL, Oracle, entre otros, aunque preferiblemente se desea la portabilidad sobre software libre.

Seguridad

RNF 19. Realizar todas las validaciones pertinentes.

RNF 20. Chequeo de seguridad sobre las acciones tales como: verificación de borrado.

Políticos-culturales

RNF 21. El sistema debe tener una interfaz que esté acorde con el lugar donde se implantará, es decir, que refleje los ideales de la organización.

Legales

RNF 22. Reconocido y autorizado por instancias superiores tales como la directiva de la UCI.

RNF 23. Documentación legal de uso como Declaración de Autoría.

Ayuda y documentación en línea

RNF 24. Documentación de ayuda para uso del sistema, la cual estará asequible desde cualquier parte del mismo para satisfacer cualquier duda que el usuario presente con el manejo y uso de la aplicación.

Restricciones en el diseño y la implementación

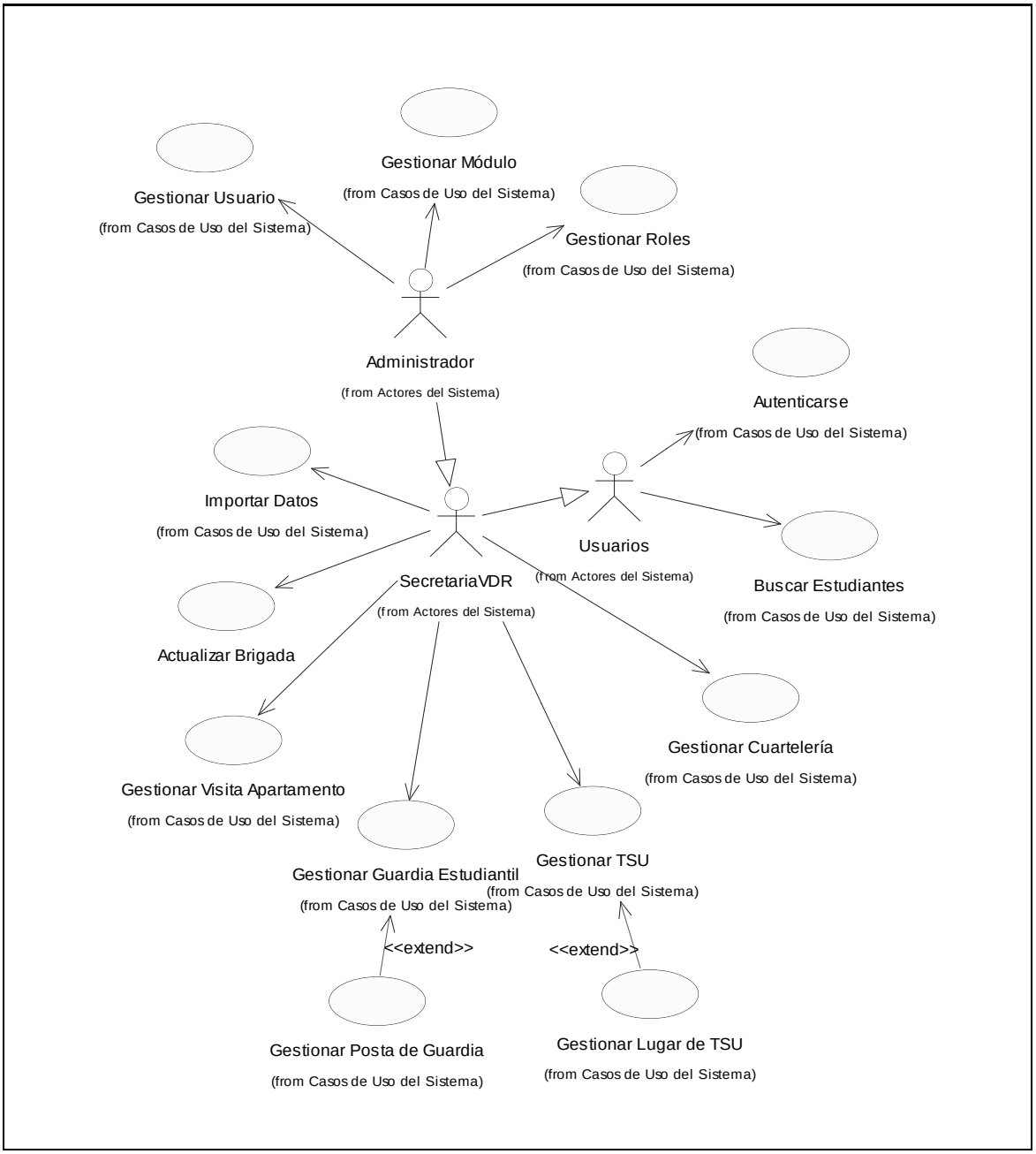
RNF 25. Para la programación en PHP se recomienda el editor Nusphere PHPEd.

RNF 26. Se recomienda el uso de la arquitectura de n capas o modelo vista controlador, con el fin de tener una mayor organización del código, además de hacer el software más extensible.

2.9 Definición de los Actores de Sistema.

Actores del sistema	Justificación
Secretaria del Vicedecano de Extensión y Residencia.	Persona que asiste al Vicedecano de Extensión y Residencia, es la encargada de conformar todas las solicitudes.
Usuarios	Directivos de la facultad, profesores.
Administrador	Persona encargada de administrar el sistema cuenta con acceso a todas las funcionalidades del sistema.

2.10 Diagrama de CUS.



2.11 Descripción de los CUS.

2.11.1 Especificación del Caso de Uso Gestionar Posta de Guardia.

<table>
<tr><td>Caso de Uso:</td><td colspan="2">Gestionar Posta de Guardia</td></tr>
<tr><td>Actores:</td><td colspan="2">Secretaria del Vicedecano Residencia.</td></tr>
<tr><td>Resumen:</td><td colspan="2">El caso de uso inicia cuando la secretaria del Vicedecano de residencia selecciona la opción gestionar posta de guardia, el sistema ejecuta dicha acción y el CUS termina.</td></tr>
<tr><td>Propósito:</td><td colspan="2">Introducir una nueva posta de guardia, eliminar una posta de guardia.</td></tr>
<tr><td>Precondiciones:</td><td colspan="2">La Secretearía del Vicedecano de residencia debe estar autentificada en el sistema.
La secretaria debe tener los permisos necesarios asignados.</td></tr>
<tr><td>Poscondiciones:</td><td colspan="2">Posta de guardia insertada en la Base de Datos.
Posta de guardia eliminada de la Base de Datos.</td></tr>
<tr><td>Referencias</td><td colspan="2">RF-3</td></tr>
<tr><td>Prioridad</td><td colspan="2">Crítico</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td colspan="3">Sección “Gestionar Posta de Guardia”</td></tr>
<tr><td colspan="2">Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td colspan="2">1. Se selecciona la opción Gestionar Posta de Guardia.</td><td>2. Muestra las siguientes acciones:
• Un formulario que permite adicionar una nueva posta de guardia. (Ver Sección: “Adicionar Posta de Guardia”).
• El listado de las postas de guardia existentes hasta ese momento.</td></tr>
<tr><td colspan="2"></td><td>3. Una vez listada las postas de guardia da la posibilidad de:
• Eliminar Posta de Guardia. (Ver Sección: “Eliminar Posta de Guardia”).</td></tr>
<tr><td colspan="3">Pantalla1.</td></tr>
<tr><td colspan="3">Sección : “Adicionar Posta de Guardia“</td></tr>
<tr><td colspan="3">Flujos Normal de Eventos</td></tr>
<tr><td colspan="2">Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td colspan="3">Pantalla 2.</td></tr>
<tr><td colspan="2">4. Introduce los datos de forma correcta y presiona el botón registrar.</td><td>5. Valida los datos.</td></tr>
<tr><td colspan="2"></td><td>6. Adiciona la Posta de Guardia a la Base de Datos y termina el CUS.</td></tr>
<tr><td colspan="3">Prototipo de Interfaz</td></tr>
</table>

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
4. a. Introduce los datos de forma incorrecta.	5.1 En caso de faltar algún campo por llenar o datos incorrectos, se muestra un mensaje de advertencia y regresa al paso 4. 5.2 En caso de coincidir algún dato que deba ser único, se muestra un mensaje de advertencia y regresa al paso 4.
Sección:"Eliminar Posta de Guardia"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
7. Selecciona la opción Eliminar Posta de Guardia.	8. Muestra un mensaje de confirmación para saber si esa la Posta de Guardia que se desea eliminar.
9. Confirma la acción de eliminar la Posta de Guardia seleccionada.	10. Posta de guardia no relacionada con ninguna guardia estudiantil.
	11. Elimina la Posta de Guardia seleccionada de la Base de Datos.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
9. a. No desea eliminar la Posta de Guardia seleccionada y oprime el botón Cancelar.	11.1 No se realiza ninguna operación.
	10.1 Posta relacionada con una guardia estudiantil. Muestra un mensaje dando a conocer que no se puede eliminar dicha posta de guardia.

2.11.2 Especificación del Caso de Uso Gestionar Cuartelería.

Caso de Uso:	**Gestionar Cuartelería.**
Actores:	**Secretaria del Vicedecano Residencia.**
Resumen:	El caso de uso inicia cuando la secretaria del Vicedecano de residencia selecciona la opción gestionar cuartelería, escoge la acción a realizar sobre dicha cuartelería, el sistema ejecuta dicha acción y el CUS termina.
Propósito:	Registrar, Modificar, Eliminar los datos pertenecientes a un una cuartelería.
Precondiciones:	La Secretaria del Vicedecano de residencia debe estar autentificada en el sistema. La secretaria debe tener los permisos necesarios asignados.
Poscondiciones:	Cuartelería insertada en la Base de Datos. Datos de la Cuartelería modificada en la Base de Datos. Cuartelería eliminada de la Base de Datos.
Referencias	**RF-4**

Prioridad	**Crítico**
Flujo Normal de Eventos	
Sección "Gestionar Cuartelería"	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción Gestionar Cuartelería.	2. Muestra una pantalla con una breve descripción del CU gestionar cuartelería, además las siguientes operaciones que el actor puede realizar: • Adicionar Cuartelería (Ver sección **"Adicionar Cuartelería**"). • Listar Cuartelería (Ver sección "**Listar Cuartelería**").
Pantalla1.	
Sección : "Adicionar Cuartelería"	
Flujos Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
3. Selecciona la opción Adicionar Cuartelería.	4. Muestra un criterio de búsqueda de estudiante y un formulario con los datos necesarios para registrar la cuartelería.
Pantalla 2.	
5. Busca el estudiante de cuartelero lo selecciona, Introduce los datos de forma correcta y presiona el botón Registrar.	6. Valida los datos.
	7. Adiciona la cuartelería a la Base de Datos, muestra un mensaje informando que la operación fue realizada con éxito y termina el CUS.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
5. a. Introduce los datos de manera incorrecta.	6.1 En caso de faltar algún campo por llenar o por completar, o datos incorrectos, se muestra un mensaje de advertencia y regresa al paso 5. 6.2 En caso de coincidir algún dato que deba ser único, se muestra un mensaje de advertencia y regresa al paso 5.
Sección : "Listar Cuartelería"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
8. Selecciona la opción Listar Cuartelería.	9. Muestra un formulario con los parámetros correspondientes para realizar el criterio de búsqueda.

10. Selecciona el criterio de búsqueda (Edificio, Paso escalera, Apartamento, fecha, Estudiante, brigada) que existen Cuartelerías en la BD y presiona en el botón Buscar.	11. Verifica si existe alguna cuartelería de acuerdo al criterio de búsqueda seleccionado.
	12. Muestra un listado de las cuartelerías que coinciden con el criterio de búsqueda seleccionado.
	13. Una vez listadas las cuartelerías brinda la posibilidad de : • Modificar Cuartelería (Ver sección "**Modificar Cuartelería**"). • Eliminar Cuartelería (Ver sección "**Eliminar Cuartelería**") El CUS termina.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
10. a. Introduce datos al criterio de búsqueda donde no existen cuartelerías en la BD con ese criterio de búsqueda seleccionado.	12.1 Muestra un mensaje dando a conocer que no existe ninguna cuartelería que corresponda con el criterio de búsqueda seleccionado, regresa al paso 10.
Sección : "Modificar Cuartelería"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
14. Selecciona la opción Modificar Cuartelería.	15. Muestra los datos de la Cuartelería listos para ser modificados.
16. Realiza los cambios necesarios a la Cuartelería correctamente y presiona el botón Actualizar.	17. Verifica los datos.
	18. Actualiza los datos de la Cuartelería en la Base de Datos, regresa a la sección listar cuartelería con el resultado de búsqueda anteriormente seleccionado y el CUS termina.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
16. a. Realiza los cambios introduciendo datos incorrectos.	17.1 En caso de error notifica a la secretaria y brinda la posibilidad de corregir el/los dato/s escritos incorrectamente y se realiza el paso 16. 17.2 En caso de coincidir algún dato que deba ser único, se muestra un mensaje de advertencia y regresa al paso 16.
Sección: "Eliminar Cuartelería"	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
19. Selecciona la opción Eliminar Cuartelería.	20. El sistema muestra un mensaje de confirmación para saber si es la Cuartelería que se desea eliminar.
21. Confirma la acción de eliminar la cuartelería seleccionada.	22. El sistema elimina la cuartelería seleccionada de la Base de Datos y el CUS termina.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
21. a. No desea eliminar la cuartelería seleccionada.	31.1 No se realiza ninguna operación.

2.11.3 Especificación del Caso de Uso Gestionar Visita a Apto.

Caso de Uso:	**Gestionar Visita a Apartamento.**
Actores:	**Secretaria del Vicedecano Residencia.**
Resumen:	El caso de uso inicia cuando la secretaria del Vicedecano de residencia selecciona la opción gestionar visita a apartamento, escoge la acción a realizar sobre dicha visita, el sistema ejecuta dicha acción y el CUS termina.
Propósito:	Registrar, Modificar, Eliminar o Consultar los datos pertenecientes a un una visita a apartamento.
Precondiciones:	La Secretearía del Vicedecano de residencia debe estar autentificada en el sistema. La secretaria debe tener los permisos necesarios asignados.
Poscondiciones:	Visita a apartamento insertada en la base de datos. Datos de la visita a apartamento modificada en la base de datos. Datos de la visita a apartamento mostrado a la secretaria. Visita a apartamento eliminada de la base de datos.
Referencias	**RF-5**
Prioridad	**Crítico**

Flujo Normal de Eventos	
Sección "Gestionar Visita a Apto"	
Acción del Actor	**Respuesta del Sistema**
1. Se selecciona la opción Gestionar Visita a Apartamento.	2. Muestra las siguientes operaciones que el actor puede realizar: • Adicionar Visita a Apartamento (Ver sección "Adicionar Visita a Apartamento"). • Listar Visitas a Apartamentos (Ver sección "Listar Visitas a Apartamentos").
Pantalla1.	
Sección : "Adicionar Visita a Apartamento"	
Flujos Normal de Eventos	

Acción del Actor	Respuesta del Sistema
3. Selecciona la opción adicionar visita a apartamento.	4. Muestra un criterio de búsqueda de personas y un formulario con los datos necesarios para registrar la visita apartamento.
Pantalla 2.	
5. Busca la/s persona/s que realizaron la visita apartamento, Introduce los datos de forma correcta y presiona el botón Adicionar Visita.	6. Valida los datos.
	7. Una vez entrado los datos correctamente debe emitir una evaluación general de la visita al apto.
	8. Adiciona la visita a apartamento a la Base de Datos, muestra un mensaje informando que la operación fue realizada con éxito y termina el CUS.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
5. a. Introduce los datos de manera incorrecta.	6.1 En caso de faltar algún campo por llenar o por completar, o datos incorrectos, se muestra un mensaje de advertencia y regresa al paso 5. 6.2 En caso de coincidir algún dato que deba ser único, se muestra un mensaje de advertencia y regresa al paso 5.
Sección : "Listar Visita a Apartamento"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
9. Selecciona la opción Listar Visita a Apartamento.	10. Muestra un formulario con los parámetros correspondientes para realizar el criterio de búsqueda.
Pantalla 3.	
11. Selecciona el criterio de búsqueda (Edificio, Apartamento, fecha, evaluación) que existen visitas a apartamentos en la base de datos y presiona en el botón Buscar.	12. Verifica si existe alguna visita a apartamento de acuerdo al criterio de búsqueda seleccionado.
	13. Muestra un listado de las visitas a apartamentos que coinciden con el criterio de búsqueda seleccionado.
	14. Una vez listadas las visitas a apartamentos brinda la posibilidad de :

	• Consultar Visita a Apartamento (Ver sección "**Consultar Visita a Apartamento**"). • Modificar Visita a Apartamento (Ver sección "**Modificar Visita a Apartamento**"). • Eliminar Visita a Apartamento (Ver sección "**Eliminar Visita a Apartamento**") El CUS termina.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
11. a. Introduce datos al criterio de búsqueda donde no existen visita a apartamento en la base de datos con ese criterio de búsqueda seleccionado.	12.1 Muestra un mensaje informado que no existen visita de apartamentos para el criterio de búsqueda seleccionado y regresa al paso 11.
Sección : "Modificar Visita a Apartamento"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
15. Selecciona la opción Modificar Visita a Apartamento.	16. Muestra los datos de la Visita listos para ser modificados.
17. Realiza los cambios necesarios a la Visita correctamente y presiona el botón Actualizar.	18. El sistema verifica los datos.
	19. Actualiza los datos de la Visita en la Base de Datos, muestra un mensaje dando a conocer que la operación fue realizada con éxito y el CUS termina.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
17. a. Realiza los cambios introduciendo datos incorrectos.	18.1 En caso de error notifica a la secretaria y brinda la posibilidad de corregir el/los dato/s escritos incorrectamente y se realiza el paso 17.
Sección: "Eliminar Visita a Apartamento"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
20. Selecciona la opción Eliminar Visita a Apartamento.	21. El sistema muestra un mensaje de confirmación para saber si es la visita a apartamento que se desea eliminar.
22. Confirma la acción de eliminar la visita seleccionada.	23. El sistema elimina la visita seleccionada de la base de datos, el CUS termina.
Flujos Alternos	

Acción del Actor	Respuesta del Sistema
22. a. No desea eliminar la visita seleccionada.	23.1 No se realiza ninguna operación.
Sección: "Consultar Visita a Apartamento"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
24. Selecciona la opción Consultar Visita a Apartamento.	25. Muestra los datos de la Visita seleccionada y un botón cerrar.
26. Presiona botón Cerrar	27. Regresa al resultado de búsqueda seleccionado anteriormente.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**

2.11.4 Especificación del Caso de Uso Gestionar Trabajo Socialmente Útil.

Caso de Uso:	**Gestionar Trabajo Socialmente Útil (TSU).**
Actores:	**Secretaria del Vicedecano Residencia.**
Resumen:	El caso de uso inicia cuando la secretaria del Vicedecano de residencia selecciona la opción gestionar TSU, escoge la acción a realizar sobre dicho TSU, el sistema ejecuta dicha acción y el CUS termina.
Propósito:	Registrar, Modificar, Eliminar o Consultar los datos pertenecientes a un TSU.
Precondiciones:	La Secretearía del Vicedecano de residencia debe estar autentificada en el sistema. La secretaria debe tener los permisos necesarios asignados.
Poscondiciones:	TSU insertado en la base de datos. Datos del TSU modificados en la base de datos. Datos del TSU mostrados a la secretaria. Datos del TSU eliminados de la base de datos.
Referencias	**RF-6**
Prioridad	**Crítico**

Flujo Normal de Eventos	
Sección "Gestionar TSU"	
Acción del Actor	**Respuesta del Sistema**
1. Se selecciona la opción Gestionar TSU.	2. Muestra Una pantalla inicio con una breve descripción del CU gestionar TSU y las siguientes operaciones que el actor puede realizar: • Adicionar TSU (Ver sección Adicionar TSU). • Listar TSU (Ver sección Listar TSU).
Pantalla1.	

Sección : Adicionar TSU	
Flujos Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
3. Selecciona la opción Adicionar TSU.	4. Muestra el listado de todos los grupos existentes en la facultad y muestra el formulario a llenar para registrar el nuevo TSU.
Pantalla 2.	
5. Selecciona el grupo que participó en dicho TSU.	6. Muestra los estudiantes pertenecientes a ese grupo.
7. Llena los datos de forma correcta (fecha, tarea, sección, lugar, evaluación, observaciones) y a cada estudiante le pone si asistió o no al TSU y presiona el botón Registrar.	8. Valida los datos.
	9. Adiciona el TSU a la base de datos, muestra un mensaje informando que la operación fue realizada con éxito y el CUS termina.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
7. a. Introduce los datos de manera incorrecta.	8.1 En caso de faltar algún campo por llenar o por completar, o datos incorrectos, se muestra un mensaje de advertencia y regresa al paso 7. 8.2 En caso de coincidir algún dato que deba ser único, se muestra un mensaje de advertencia y regresa al paso 7.
Sección : Listar TSU	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
10. Selecciona la opción Listar TSU.	11. Muestra un formulario con los parámetros correspondientes para realizar el criterio de búsqueda.
Pantalla 3.	
12. Selecciona el criterio de búsqueda (Brigada, fecha, lugar, evaluación) que existen TSU en la BD y da en el botón Buscar.	13. Muestra el listado de TSU
	14. Una vez listados los TSU que corresponden con el criterio de búsqueda brinda la posibilidad de : • Consultar TSU (Ver sección "**Consultar TSU**").

	• Modificar TSU (Ver sección "**Modificar TSU**"). • Eliminar TSU (Ver sección "**Eliminar TSU**") El CUS termina.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
12. a. Introduce datos al criterio de búsqueda con los que no existen TSU en la base de datos.	13.1 Muestra en un mensaje que no existe ningún TSU para el criterio de búsqueda seleccionado y regresa al paso 12.
Sección : "**Modificar TSU**"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
15. Selecciona la opción Modificar TSU.	16. Muestra los datos del TSU listos para ser modificados.
17. Realiza los cambios necesarios al TSU correctamente y presiona el botón Actualizar.	18. El sistema verifica los datos.
	19. Actualiza los datos del TSU en la Base de Datos, muestra un mensaje dando a conocer que la operación fue realizada con éxito y el CUS termina.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
18. a. Realiza los cambios introduciendo datos incorrectos.	18.1 En caso de error notifica a la secretaria y brinda la posibilidad de corregir el/los dato/s escritos incorrectamente y se realiza el paso 17.
Sección "Eliminar TSU"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
20. Selecciona la opción Eliminar TSU.	21. El sistema muestra un mensaje de confirmación para saber si ese es el TSU que se desea eliminar.
22. Confirma la acción de eliminar el TSU seleccionado.	23. El sistema elimina el TSU seleccionado de la Base de Datos.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
23. a. No desea eliminar el TSU seleccionado.	24.1 No se realiza ninguna operación.
Sección "Consultar TSU"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
24. Selecciona la opción Consultar TSU.	25. Muestra los datos del TSU seleccionado, y el botón cerrar.

26. Presiona botón Cerrar	27. Regresa al resultado de la búsqueda seleccionado anteriormente, y el CUS termina.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**

2.11.6 Especificación del Caso de Uso Gestionar Usuario.

Caso de Uso:	**Gestionar Usuario.**
Actores:	**Vicedecano Residencia.**
Resumen:	El caso de uso inicia cuando el vicedecano de residencia selecciona la opción Gestionar Usuario, escoge la acción a realizar el sistema ejecuta dicha acción y el CUS termina.
Propósito:	Introducir o Eliminar los datos pertenecientes a un usuario.
Precondiciones:	El Vicedecano de residencia debe estar autentificado en el sistema. Debe tener los permisos necesarios asignados.
Poscondiciones:	Usuario insertado en la base de datos. Usuario eliminado de la base de datos.
Referencias	**RF-1**
Prioridad	**Crítico**

Flujo Normal de Eventos	
Sección "Gestionar Usuario"	
Acción del Actor	**Respuesta del Sistema**
1. Se selecciona la opción Gestionar Usuario.	2. Muestra un formulario con los datos necesarios para registrar un nuevo usuario. Ver sección: "**Adicionar Usuario**". 3. Listado de los usuarios existentes en la base de datos, permitiendo que los mismos puedan ser eliminados. Ver sección "**Eliminar Usuario**".
Pantalla1.	
Sección : "Adicionar Usuario"	
Flujos Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
4. Introduce los datos de forma correcta y presiona el botón Adicionar.	5. Valida los datos.
	6. Adiciona el Usuario a la Base de Datos, y termina el CUS.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**

4. a. Introduce los datos de manera incorrecta.	5.1 En caso de faltar algún campo por llenar o por completar, o datos incorrectos, se muestra un mensaje de advertencia y regresa al paso 4. 5.2 En caso de coincidir algún dato que deba ser único, se muestra un mensaje de advertencia y regresa al paso 4.
Sección: "Eliminar Usuario"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
7. Selecciona la opción Eliminar Usuario.	8. El sistema muestra un mensaje de confirmación para saber si es el usuario que se desea eliminar.
9. Confirma la acción de eliminar el usuario seleccionado.	10. El sistema elimina el usuario seleccionado de la base de datos y el CUS termina.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
21. a. No desea eliminar el usuario seleccionado.	22.1 No se realiza ninguna operación.

2.11.7 Especificación del Caso de Uso Gestionar Lugar TSU.

Caso de Uso:	**Gestionar Lugar TSU**
Actores:	**Secretaria del Vicedecano Residencia.**
Resumen:	El caso de uso inicia cuando la secretaria del vicedecano de residencia selecciona la opción gestionar Lugar TSU, el sistema ejecuta dicha acción y el CUS termina.
Propósito:	Introducir un nuevo lugar TSU, eliminar un lugar TSU.
Precondiciones:	La Secretearía del Vicedecano de residencia debe estar autentificada en el sistema. La secretaria debe tener los permisos necesarios asignados.
Poscondiciones:	Lugar TSU insertado en la base de datos. Lugar TSU eliminado de la base de datos.
Referencias	**RF-8**
Prioridad	**Crítico**
Flujo Normal de Eventos	
Sección "Gestionar Lugar TSU"	
Acción del Actor	**Respuesta del Sistema**
1. Se selecciona la opción Gestionar Lugar TSU.	2. Muestra un formulario con los datos necesarios para registrar un nuevo lugar de TSU. 3. Listado con los lugares de TSU existentes en la base de datos, dando la posibilidad de eliminarlo. Ver sección

	"**Eliminar Lugar TSU**".
Pantalla1.	
Sección : "Adicionar Lugar TSU"	
Flujos Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
Pantalla 2.	
4. Introduce los datos de forma correcta y presiona el botón Registrar.	5. Valida los datos.
	6. Adiciona el lugar TSU a la base de datos, termina el CUS.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
4. a. Introduce los datos de forma incorrecta.	5.1 En caso de faltar algún campo por llenar o datos incorrectos, se muestra un mensaje de advertencia y regresa al paso 4. 5.2 En caso de coincidir algún dato que deba ser único, se muestra un mensaje de advertencia y regresa al paso 4.
Sección "Eliminar Lugar TSU"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
7. Selecciona la opción Eliminar Lugar TSU.	8. Muestra un mensaje de confirmación para saber si ese es el lugar de TSU que se desea eliminar.
9. Confirma la acción de eliminar el lugar TSU seleccionado.	10. Elimina el lugar TSU seleccionado de la base de datos.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
9. a. No desea eliminar el lugar TSU seleccionado.	10.1 No se realiza ninguna operación.

2.11.8 Especificación del Caso de Uso Gestionar Módulo.

Caso de Uso:	**Gestionar Módulos.**
Actores:	**Vicedecano Residencia.**
Resumen:	El caso de uso inicia cuando el vicedecano de residencia selecciona la opción gestionar módulo, el sistema ejecuta dicha acción y el CUS termina.

Propósito:	Introducir un nuevo módulo, eliminar un módulo.
Precondiciones:	El vicedecano de residencia debe estar autentificado en el sistema. Debe tener los permisos necesarios asignados.
Poscondiciones:	Módulo insertado en la base de datos. Módulo eliminado de la base de datos.
Referencias	**RF-7**
Prioridad	**Crítico**

Flujo Normal de Eventos	
Sección "Gestionar Módulo"	
Acción del Actor	**Respuesta del Sistema**
1. Se selecciona la opción Gestionar Módulo.	2. Muestra un formulario con los datos necesarios para registrar un nuevo módulo. 3. Listado con los Módulos existentes en la base de datos, dando la posibilidad de eliminarlo. Ver sección "**Eliminar Módulo**".
Pantalla1.	
Sección : "Adicionar Módulo "	
Flujos Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
Pantalla 2.	
4. Introduce los datos de forma correcta y presiona el botón Registrar.	5. Valida los datos.
	6. Adiciona el nuevo módulo a la base de datos, termina el CUS.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
4. a. Introduce los datos de forma incorrecta.	5.1 En caso de faltar algún campo por llenar o datos incorrectos, se muestra un mensaje de advertencia y regresa al paso 4. 5.2 En caso de coincidir algún dato que deba ser único, se muestra un mensaje de advertencia y regresa al paso 4.
Sección "Eliminar Módulo"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
7. Selecciona la opción Eliminar Módulo.	8. Muestra un mensaje de confirmación para saber si ese es el módulo que se desea eliminar.

9. Confirma la acción de eliminar el módulo seleccionado.	10. Elimina el módulo seleccionado de la base de datos.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
9. a. No desea eliminar el módulo seleccionado.	10.1 No se realiza ninguna operación.

2.11.9 Especificación del Caso de Uso Gestionar Rol.

Caso de Uso:	**Gestionar Rol.**
Actores:	**Vicedecano Residencia.**
Resumen:	El caso de uso inicia cuando el vicedecano de residencia selecciona la opción Gestionar Rol, el sistema ejecuta dicha acción y el CUS termina.
Propósito:	Introducir un nuevo rol, eliminar un rol.
Precondiciones:	El vicedecano de residencia debe estar autentificado en el sistema. Debe tener los permisos necesarios asignados.
Poscondiciones:	Rol insertado en la base de datos. Rol eliminado de la base de datos.
Referencias	**RF-10**
Prioridad	**Crítico**

Flujo Normal de Eventos	
Sección "Gestionar Rol"	
Acción del Actor	**Respuesta del Sistema**
1. Se selecciona la opción Gestionar Rol.	2. Muestra la posibilidad de: • Adicionar Rol. Ver sección: "**Adicionar Rol**". • Listado con los roles existentes en la base de datos, dando la posibilidad de eliminarlo. Ver sección "**Eliminar Rol**".
Pantalla1.	
Sección : "Adicionar Rol "	
Flujos Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
3. Selecciona la opción adicionar nuevo rol.	4. El sistema muestra un formulario con los datos a llenar necesarios para registrar un nuevo rol.
5. Introduce los datos de forma correcta y presiona el botón Registrar.	6. Valida los datos.
	7. Adiciona el nuevo rol a la base de datos. 8. Regresa a Gestionar Rol, termina el CUS.

Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
5. a. Introduce los datos de forma incorrecta.	6.1 En caso de faltar algún campo por llenar o datos incorrectos, se muestra un mensaje de advertencia y regresa al paso 5. 6.2 En caso de coincidir algún dato que deba ser único, se muestra un mensaje de advertencia y regresa al paso 5.
Sección "Eliminar Rol"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
9. Selecciona la opción Eliminar Rol.	10. Muestra un mensaje de confirmación para saber si ese es el rol que se desea eliminar.
11. Confirma la acción de eliminar el rol seleccionado.	12. Elimina el rol seleccionado de la base de datos.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
11. a. No desea eliminar el rol seleccionado.	12.1 No se realiza ninguna operación.

2.11.10 Especificación del Caso de Uso Buscar Estudiante.

Caso de Uso:	**Buscar Estudiante.**
Actores:	**Usuario**
Resumen:	El caso de uso inicia cuando el usuario selecciona la opción Buscar Estudiante, el sistema ejecuta dicha acción y el CUS termina.
Propósito:	Buscar estudiante.
Precondiciones:	El usuario de residencia debe estar autentificado en el sistema.
Poscondiciones:	Mostrar datos del estudiante buscado.
Referencias	**RF-2**
Prioridad	**Crítico**
Flujo Normal de Eventos	
Sección "Buscar Estudiantes"	
Acción del Actor	**Respuesta del Sistema**
13. Selecciona la opción Buscar Estudiante.	14. El sistema muestra un formulario para que el usuario introduzca el criterio búsqueda mediante una cadena.
15. Introduce cadena de búsqueda.	16. Muestra listado de estudiantes que coincide con la cadena de búsqueda, el CU termina.
Prototipo de Interfaz	
Flujos Alternos	

Acción del Actor	Respuesta del Sistema
5. a. Introduce cadena de búsqueda para la cual no existen estudiantes en la base de datos.	6.1 Se muestra un mensaje dando a conocer que no hay estudiantes para la cadena de búsqueda introducida.

2.11.11 Especificación del Caso de Uso Actualizar Brigada.

Caso de Uso:	**Actualizar Brigada.**
Actores:	**Secretaria del Vicedecano de Residencia.**
Resumen:	El caso de uso inicia cuando la secretaria selecciona la opción Actualizar Brigada, el sistema ejecuta dicha acción y el CUS termina.
Propósito:	Actualizar los datos pertenecientes a una brigada.
Precondiciones:	La secretaria debe estar autentificada en el sistema.
Poscondiciones:	Datos de la brigada actualizados en la base de datos.
Referencias	**RF-9**
Prioridad	**Crítico**

Flujo Normal de Eventos	
Sección "Actualizar Brigada"	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción Actualizar Brigada.	2. El sistema muestra un listado con las brigadas.
3. Selecciona la brigada a la cual se le desean actualizar los datos (sección, profesor guía, jefe grupo).	4. Muestra un formulario con los datos de la brigada listos para ser modificados.
5. Modifica los datos.	6. Valida los datos
	7. Actualiza los datos de la brigada en la base de datos.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
5. a. Introduce los datos de manera incorrecta.	6.1 En caso de faltar algún campo por llenar, se muestra un mensaje de advertencia y regresa al paso 5. 6.2 En caso de coincidir algún dato que deba ser único, se muestra un mensaje de advertencia y regresa al paso 5.

2.11.12 Especificación del Caso de Uso Importar Datos.

Caso de Uso:	**Importar Datos.**
Actores:	**Secretaria del Vicedecano de Residencia.**
Resumen:	El caso de uso inicia cuando la secretaria selecciona la opción Importar datos, el sistema ejecuta dicha acción conectándose a diferentes *webserver* e importando la información relacionada a personas (estudiantes, trabajadores y profesores), ubicación de los estudiantes en la residencia, así como las brigadas con que cuenta la facultad, y el CUS termina.

Propósito:	Importar datos.
Precondiciones:	La secretaria debe estar autentificada en el sistema.
Poscondiciones:	Datos de las personas, brigadas, de la facultad8 importados en la base de datos.
Referencias	**RF-12**
Prioridad	**Crítico**

Flujo Normal de Eventos	
Sección "Importar Datos"	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción Importar Datos.	2. El sistema se conecta a los *webserver* extrae la información y la inserta en la base de datos.
	3. Muestra un mensaje dando a conocer que la operación fue efectuada correctamente.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3.1 Muestra un mensaje dando a conocer que ha ocurrido algún fallo que vuelva a realizar la operación nuevamente.

2.11.13 Especificación del Caso de Uso Gestionar Guardia Estudiantil.

Caso de Uso:	**Gestionar Guardia Estudiantil.**
Actores:	**Secretaria del Vicedecano Residencia.**
Resumen:	El caso de uso inicia cuando la secretaria del Vicedecano de residencia selecciona la opción gestionar guardia estudiantil, escoge la acción a realizar sobre dicha guardia, el sistema ejecuta dicha acción y el CUS termina.
Propósito:	Registrar, Modificar, Eliminar o Consultar los datos pertenecientes a una Guardia Estudiantil.
Precondiciones:	La Secretearía del Vicedecano de residencia debe estar autentificada en el sistema. La secretaria debe tener los permisos necesarios asignados.
Poscondiciones:	Guardia estudiantil insertada en la base de datos. Datos de la Guardia estudiantil modificados en la base de datos. Datos de la Guardia estudiantil mostrados a la secretaria. Datos de la Guardia estudiantil eliminados de la base de datos.
Referencias	**RF-11**
Prioridad	**Crítico**

Flujo Normal de Eventos	
Sección "Gestionar Guardia Estudiantil"	
Acción del Actor	**Respuesta del Sistema**
1. Se selecciona la opción Gestionar Guardia Estudiantil.	2. Muestra Una pantalla inicio con una breve descripción del CU

	gestionar guardia estudiantil y las siguientes operaciones que el actor puede realizar: • Adicionar Guardia (Ver sección Adicionar Guardia). • Listar Guardia (Ver sección Listar Guardia).
Pantalla1.	
Sección : Adicionar Guardia	
Flujos Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
3. Selecciona la opción Adicionar Guardia.	4. Muestra el listado de todos los grupos existentes en la facultad y Muestra un formulario con los datos a llenar para registrar la guardia.
Pantalla 2.	
5. Selecciona el grupo que participó en dicha guardia.	6. Muestra listado de estudiantes pertenecientes a ese grupo.
7. Llena los datos de forma correcta y a cada estudiante le pone la posta, turno y cumplimiento y presiona el botón Registrar.	8. Valida los datos.
	9. Adiciona la guardia a la base de datos, muestra un mensaje informando que la operación fue realizada con éxito y el CUS termina.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
7. a. Introduce los datos de manera incorrecta.	8.1 En caso de faltar algún campo por llenar o por completar, o datos incorrectos, se muestra un mensaje de advertencia y regresa al paso 7. 8.2 En caso de coincidir algún dato que deba ser único, se muestra un mensaje de advertencia y regresa al paso 7.
Sección : Listar Guardia	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
10. Selecciona la opción Listar Guardia.	11. Muestra un formulario con los parámetros correspondientes para realizar el criterio de búsqueda.
12. Selecciona el criterio de búsqueda que existen guardias en la BD y da en el botón Buscar.	13. Muestra el listado de las guardias.
	14. Una vez listadas las guardias que corresponden con el criterio de búsqueda brinda la posibilidad de : • Consultar guardia (Ver sección "**Consultar Guardia**"). • Modificar guardia (Ver sección "**Modificar Guardia**"). • Eliminar guardia (Ver sección "**Eliminar Guardia**") El CUS termina.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
12. a. Introduce datos al criterio de búsqueda donde no existen guardias en la base de datos con ese criterio de búsqueda seleccionado.	13.1 Muestra un mensaje informando que no existe ninguna guardia para el criterio de búsqueda seleccionado y regresa al paso 12.
Sección : "**Modificar Guardia**"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
15. Selecciona la opción Modificar Guardia.	16. Muestra los datos de la guardia listos para ser modificados.
17. Realiza los cambios necesarios a la guardia correctamente y presiona el botón Actualizar.	18. El sistema verifica los datos.
	19. Actualiza los datos de la guardia en la Base de Datos, muestra un mensaje dando a conocer que la operación fue realizada con éxito y el CUS termina.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
18. a. Realiza los cambios introduciendo datos incorrectos.	18.1 En caso de error notifica a la secretaria y brinda la posibilidad de corregir el/los dato/s escritos incorrectamente y se realiza el paso 17.
Sección "Eliminar Guardia"	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
20. Selecciona la opción Eliminar Guardia.	21. El sistema muestra un mensaje de confirmación para saber si es la Guardia que se desea eliminar.
22. Confirma la acción de eliminar la guardia seleccionada.	23. El sistema elimina la guardia seleccionada de la Base de Datos.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
23. a. No desea eliminar la guardia seleccionada.	24.1 No se realiza ninguna operación.
Sección "Consultar Guardia"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
24. Selecciona la opción Consultar Guardia.	25. Muestra los datos de la Guardia seleccionada, y el botón cerrar.
26. Presiona botón Cerrar	27. Regresa al resultado de la búsqueda seleccionada anteriormente, y el CUS termina.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**

2.12 Conclusiones.

En este capítulo fueron expuestos los requisitos funcionales y no funcionales que debe cumplir la aplicación, se describieron los procesos de negocio para un mejor entendimiento. Se desarrollaron además los respectivos diagramas y casos de usos, así como la descripción textual de cada uno, para tener un mayor conocimiento y claridad a la hora de realizar la aplicación web.

CAPÍTULO 3 ANÁLISIS Y DISEÑO DEL SISTEMA.

3.1 Introducción

En el presente capítulo se diseñan los diagramas de clases del análisis, con el objetivo de brindar una visión preliminar y de fácil entendimiento de los procesos involucrados en el Sistema. Se realiza un modelo de diseño, donde se explica detalladamente el funcionamiento específico de los artefactos que intervienen en la construcción de la Aplicación Web. Se exponen los patrones de diseño a utilizar y como se lleva a cabo la seguridad, el tratamiento de errores y el diseño de la interfaz.

3.2 Análisis del sistema.

Durante el análisis se examinan los requisitos mediante su refinamiento y estructuración, con el objetivo de conseguir una comprensión más precisa de los requisitos y una descripción de los mismos.

3.2.1 Diagramas de clases de análisis.

Los diagramas de clases de análisis, expresan la definición y relación entre las clases, están compuestos por clases que pueden ser de tres tipos fundamentales: interfaz, controladora y entidad.

Caso de uso: Autenticarse.

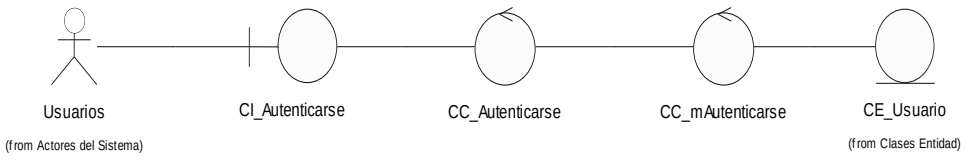


Figura 3.1 Diagrama de clases del Análisis del caso de uso Autenticarse

Caso de uso: Buscar estudiante.

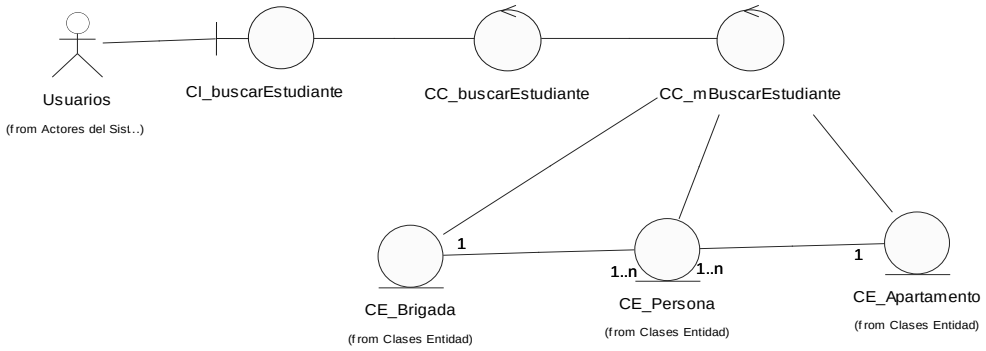


Figura 3.2 Diagrama de clases del Análisis del caso de uso "Buscar Estudiante".

Caso de uso: Gestionar cuartelería.

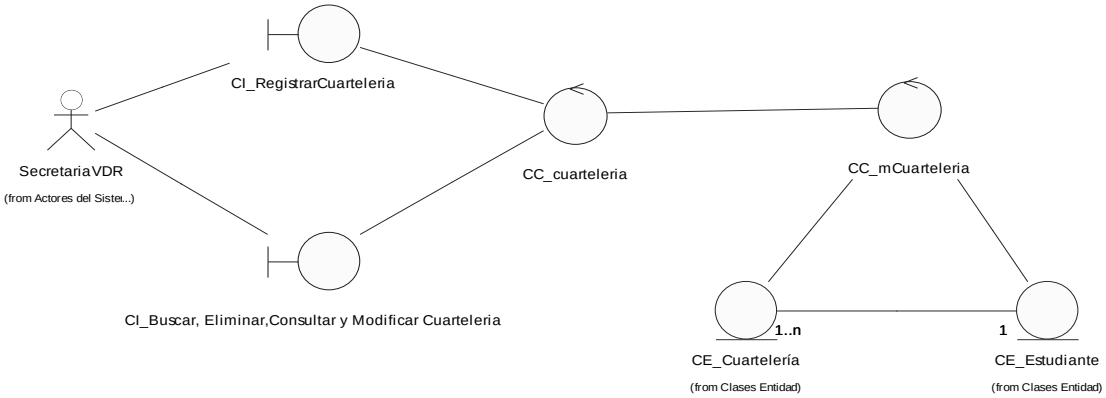


Figura 3.3 Diagrama de clases del Análisis del caso de uso "Gestionar Cuartelería".

Caso de uso: Gestionar lugar de TSU.

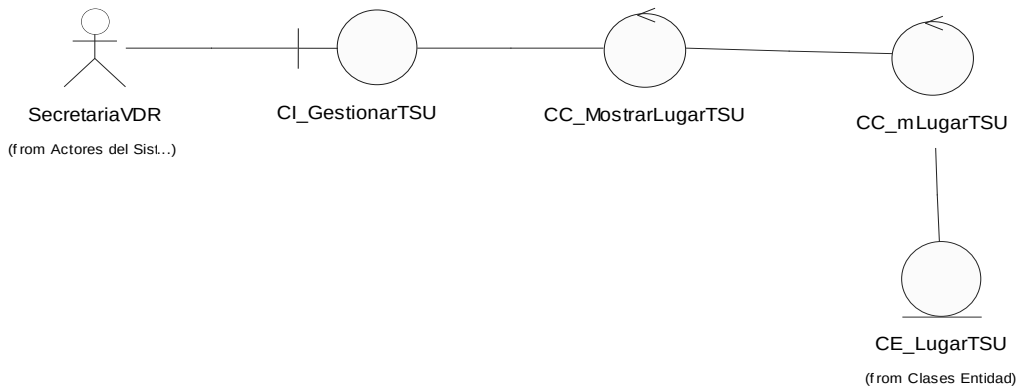


Figura 3.4 Diagrama de clases del Análisis del caso de uso "Gestionar Lugar de TSU".

Caso de uso: Gestionar guardia estudiantil.

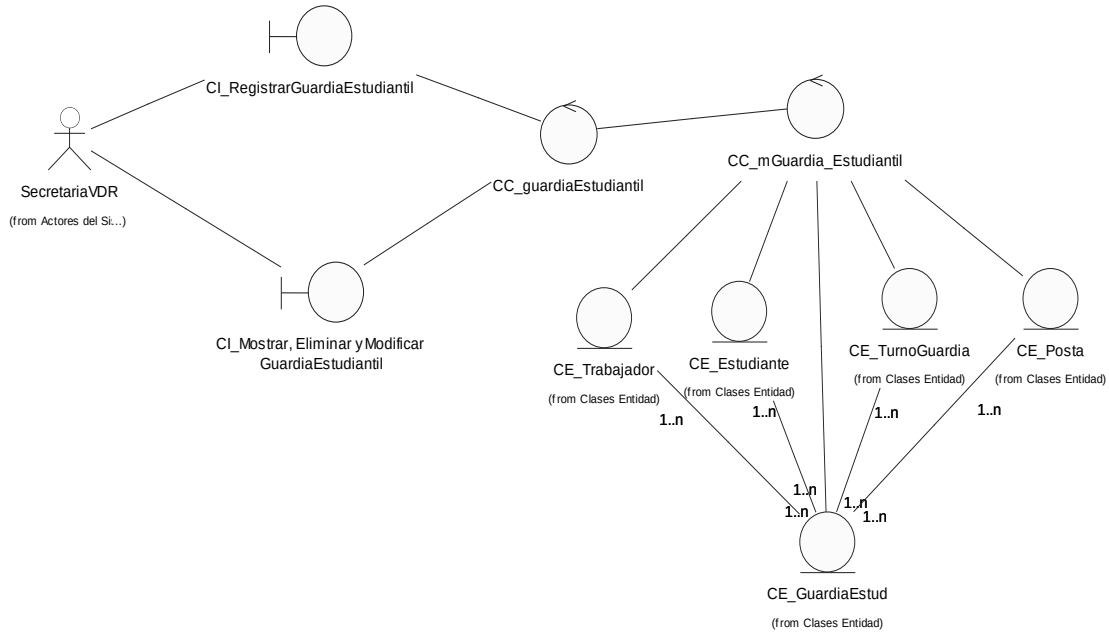


Figura 3.5 Diagrama de clases del Análisis del caso de uso "Gestionar Guardia Estudiantil".

Caso de uso: Gestionar módulo.

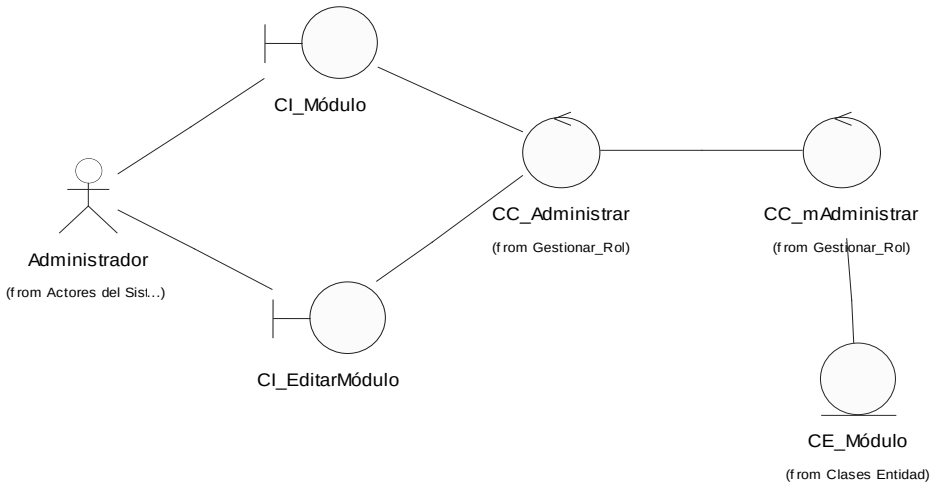


Figura 3.6 Diagrama de clases del Análisis del caso de uso "Gestionar Módulo".

Caso de uso: Gestionar posta de guardia.

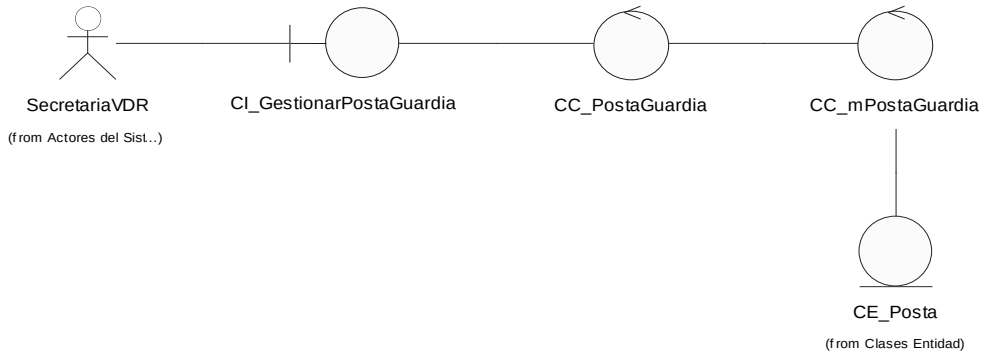


Figura 3.7 Diagrama de clases del Análisis del caso de uso "Gestionar Posta de Guardia".

Caso de uso: Gestionar rol

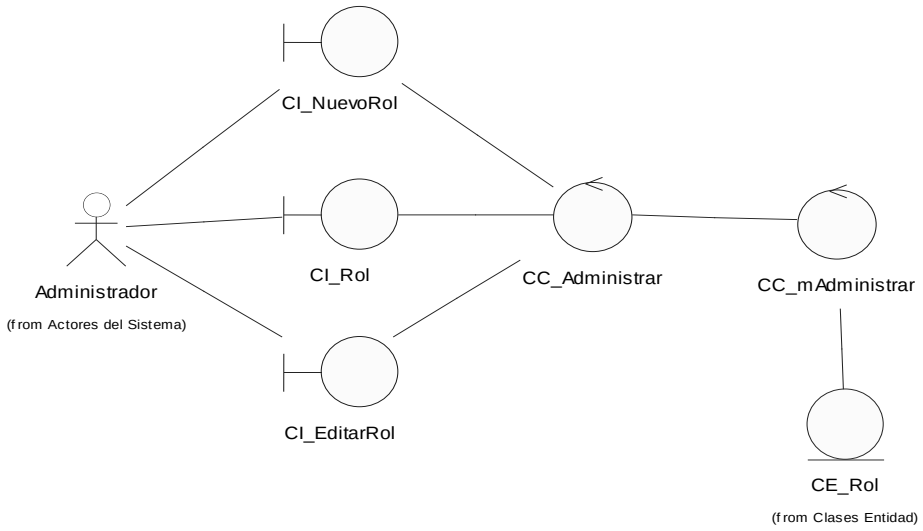


Figura 3.8 Diagrama de clases del Análisis del caso de uso "Gestionar Rol".

Caso de uso: Gestionar TSU.

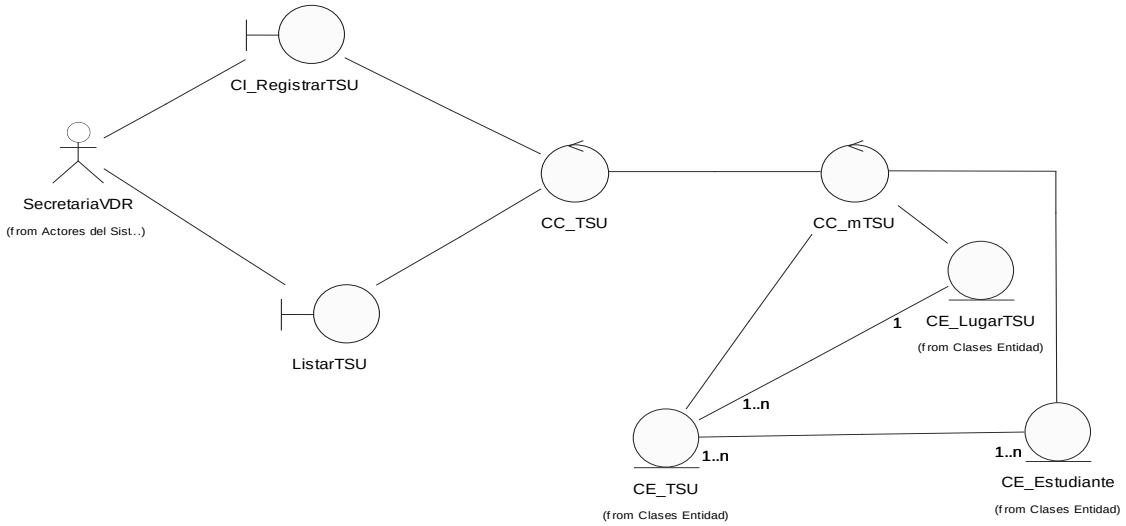


Figura 3.9 Diagrama de clases del Análisis del caso de uso "Gestionar TSU".

Caso de uso: Gestionar usuario.

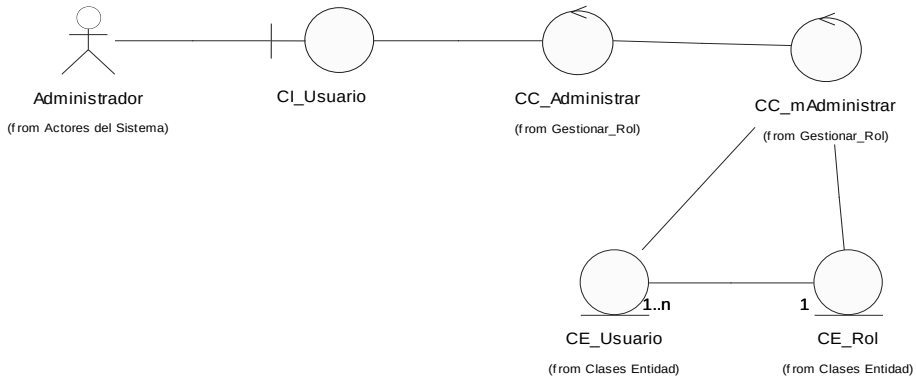


Figura 3.10 Diagrama de clases del Análisis del caso de uso "Gestionar usuario".

Caso de uso: Gestionar visita a apartamento.

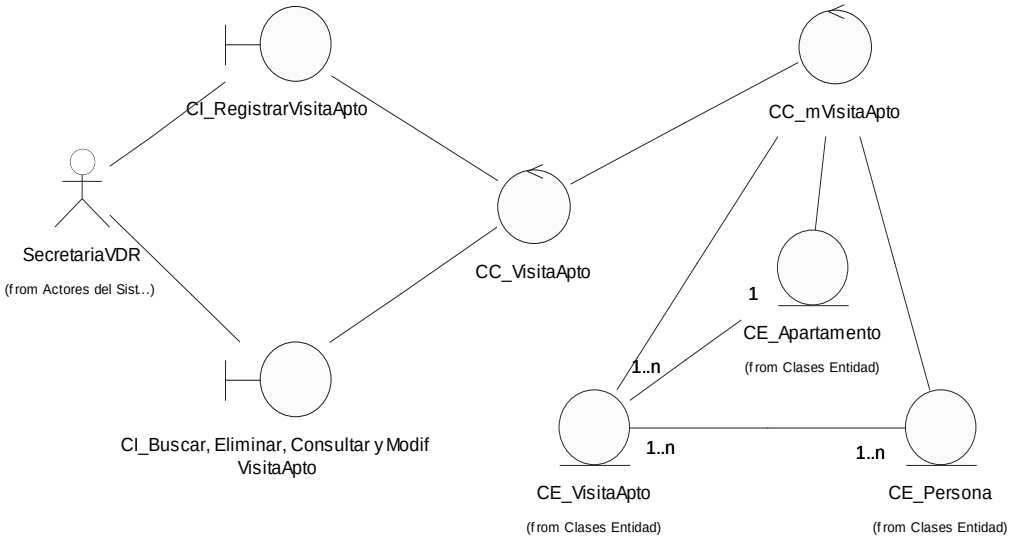


Figura 3.11 Diagrama de clases del Análisis del caso de uso "Gestionar Visita a Apartamento".

3.3 Diseño del sistema

3.3.1 Diagramas de clases del diseño.

Se construyeron varios diagramas de clases Web, uno por cada caso de uso del sistema, con el objetivo de brindar un mayor entendimiento acerca de cómo interaccionan los componentes inmersos en cada uno de ellos y su funcionamiento dentro del proceso. A continuación se muestran los diagramas de algunos de los casos de uso más significativos, y que a su vez son similares a otros en cuanto a su comportamiento; el resto se encuentran reflejados en el Anexo 1.

Caso de uso: Gestionar visita a apartamento.

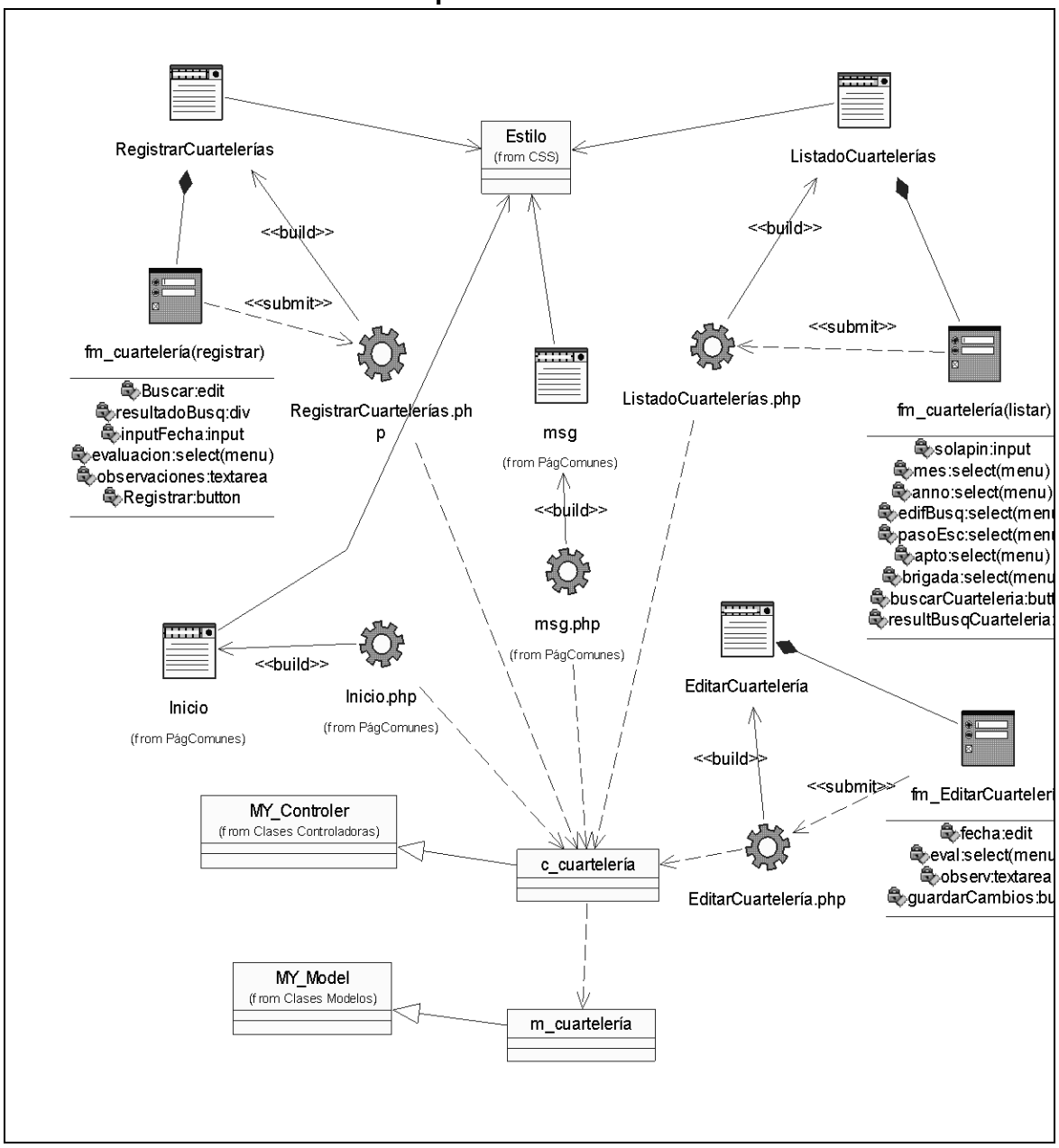


Figura 3.12 Diagrama de clases del Diseño del caso de uso "Gestionar Visita a Apartamento".

Caso de uso: Gestionar trabajo socialmente útil.

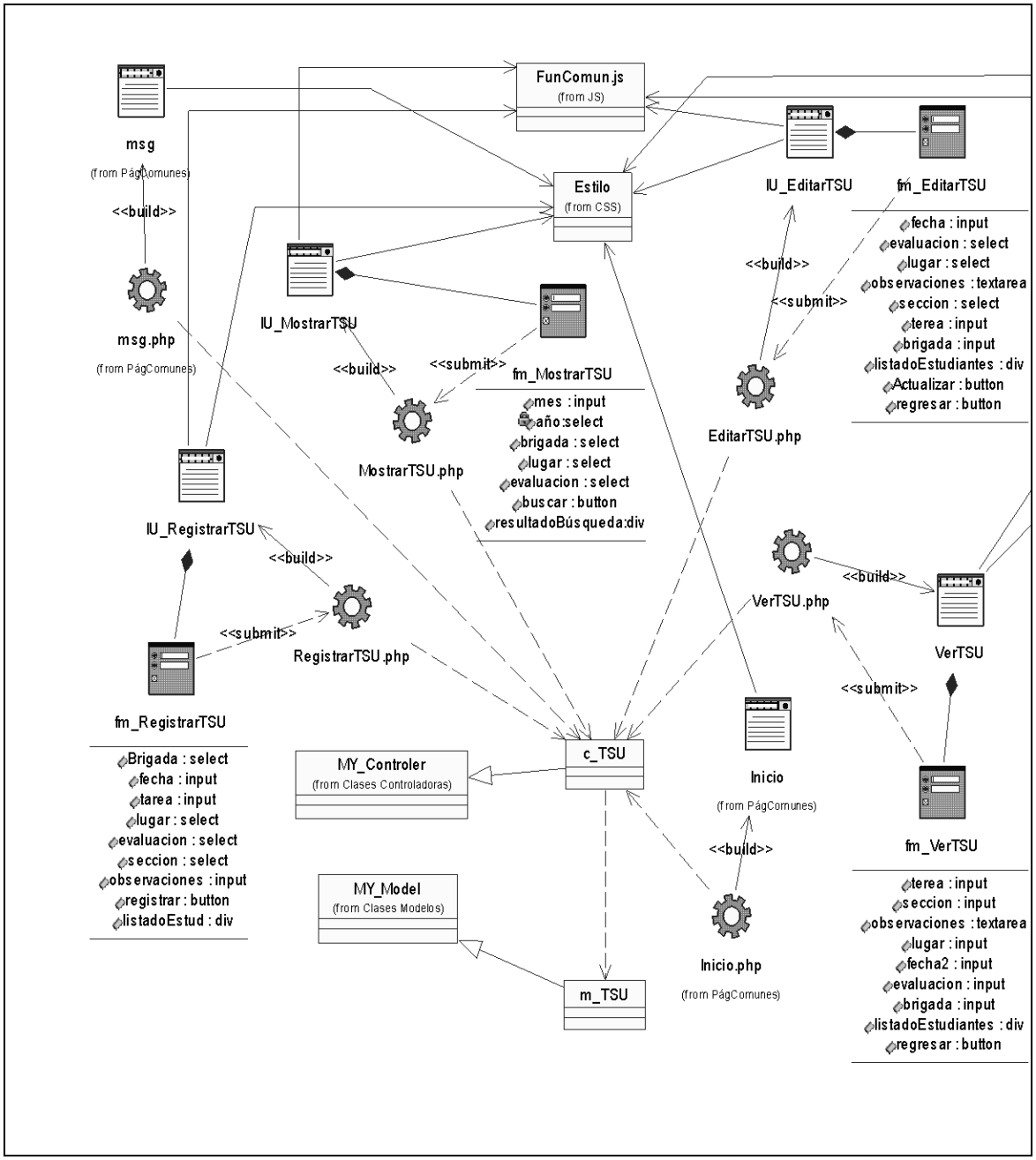


Figura 3.13 Diagrama de clases del Diseño del caso de uso "Gestionar TSU".

Caso de uso: Gestionar rol.

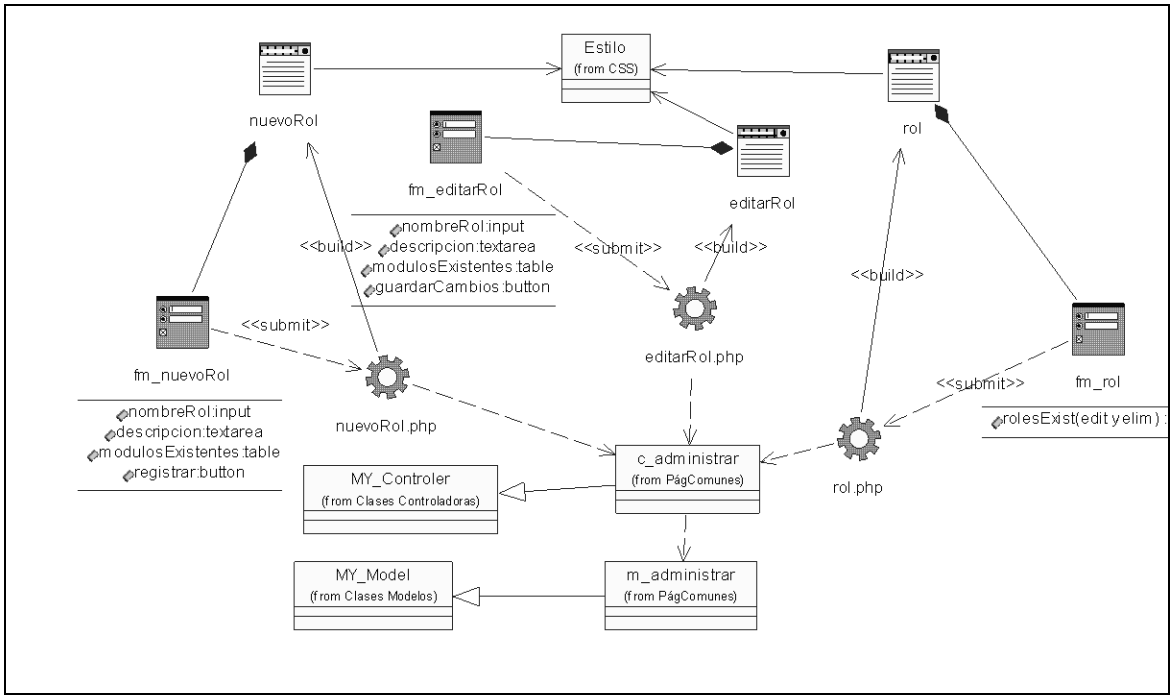


Figura 3.14 Diagrama de clases del Diseño del caso de uso "Gestionar Rol".

Caso de uso: Gestionar Lugar de TSU.

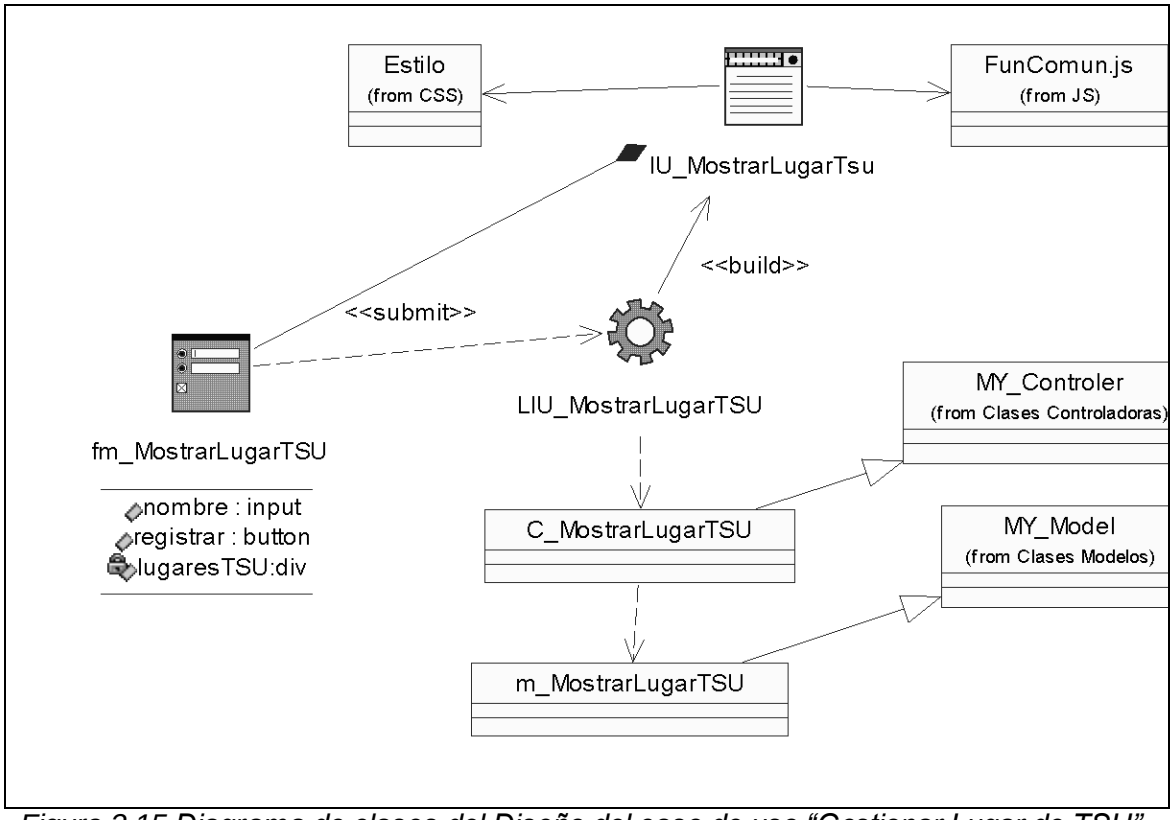


Figura 3.15 Diagrama de clases del Diseño del caso de uso "Gestionar Lugar de TSU".

3.3.2 Diagramas de interacción

Los casos de uso que presentan un comportamiento similar fueron agrupados y se les realizó un diagrama de secuencia para cada uno de sus escenarios (adicionar, modificar, eliminar y consultar), para describir el flujo principal entre las diferentes clases del diseño que intervienen en él. Debido a la gran extensión de los mismos se encuentran reflejados en los anexos (Ver anexo 3).

3.3.3Descripción de Clases del Diseño.

Descripción de Clases de Acceso a Datos

Nombre: m_RegistrarVisitaApto.	
Tipo de Clase : Acceso a Datos	
Nombre	**Descripción**
m_RegistrarVisitaApto()	Constructor de la clase, en este caso invocando al constructor de la clase padre MY_Model().
buscarPersonasDadoCadenaBusqueda($cadena)	Retorna en una lista los atributos de las personas que contengan el segmento de texto “$cadena” en uno de sus campos en la base de datos.
registrarVisitaApto($arrPersonas,$apto,$fecha,$limpieza,$ordenInterior,$cuidadoMedios,$educacionFormal,$ahorroEnergia,$evalGeneral)	Inserta los datos de la visita en la tabla tb_dvisita_apartamento, y el identificador de las personas que participaron junto al identificador de la visita realizada en la tabla tb_dpersona_tb_dvisita_apartamento
getPersona($id)	Devuelve el nombre completo y el solapín de la persona con ese identificador.
getEdificios()	Devuelve todos los edificios que estén registrados en la base de datos.
getAptosDelEdif($edif)	Devuelve todos los apartamentos que contenga un edificio.
busquedaAvanzadaVisitaApto($dia,$mes,$anno,$apto,$evaluacion, $cantidad_mostrar_paginado,$paginaBusq)	Devuelve las visitas realizadas teniendo en cuenta una fecha, un apartamento, una evaluación específica, o la combinación entre algunos de éstos parámetros.

BuscarDatosVisita($idVisita)	Devuelve todos los datos de una visita dado su identificador.
BuscarPersonasDadoIdVisitaApto($idVisita)	Devuelve el listado de las personas que participaron en una visita dado el identificador de esta.
eliminarVisitaApto($idVisita)	Elimina una visita de la tabla tb_dvisita_apartamento y de la tabla tb_persona_tb_dvisita_apartamento.
actualizarVisitaApto($idVisita,$arrPersonas,$fecha,$limpieza,$ordenInterior,$cuidadoMedios,$educacionFormal,$ahorroEnergia,$evalGeneral,$observ)	Modifica los datos de una visita a un apartamento.

Nombre : m_TSU	
Tipo de Clase : Acceso a Datos	
Nombre	**Descripción**
m_TSU ()	Constructor de la clase, en este caso invocando al constructor de la clase padre MY_Model ().
buscarEstudianteDadoIdBrigada($idBrigada)	Retorna una lista con los estudiantes pertenecientes a una brigada específica.
buscarBrigadas()	Retorna el nombre y el identificador de todas las brigadas existentes en la base de datos.
cantEstudiantesDadoBrigada($idBrigada)	Retorna el número de estudiantes que posee una brigada determinada.
cantidadEstudiantesAsistieronTsu($idTsu)	Retorna el número de estudiantes que asistieron a un TSU determinado.
buscarLugar()	Retorna el nombre y el identificador de todos los lugares existentes en la base de datos.
cantEstudiantesDadoBrigada($idBrigada)	Devuelve el número de estudiantes con que cuenta una brigada específica.
cantidadEstudiantesAsistieronTsu($idTsu)	Devuelve el número de estudiantes que asistieron a un TSU determinado.

BuscarDatosDadoIdTsu($idTSU)	Devuelve los datos de un TSU dado su identificador.
registrarTrabajoSocialmenteUtil($arrIdEst,$fecha,$evaluac,$idLugar,$tarea,$seccion,$observ,$idBrigada,$cumplimiento)	Inserta un nuevo TSU en la base de datos.
busquedaAvanzadaTsu($mes,$anno,$brigada,$lugar,$evaluacion,$cantidadMostrar,$indice)	Devuelve los TSU realizados por una brigada en específico teniendo en cuenta un mes, un año, un lugar, una evaluación, o la combinación entre algunos de estos parámetros.
obtenerCantidadTSU($brigada,$mes,$anno,$lugar,$evaluacion)	Devuelve la cantidad de TSU realizados por una brigada en específico teniendo en cuenta un mes, un año, un lugar, una evaluación, o la combinación entre algunos de estos parámetros.
buscarEstudiantesDadoIdTsu($idTsu)	Devuelve el listado de los estudiantes que participaron en un TSU dado el identificador de este.
eliminarTsu($idTsu)	Elimina un TSU de la base de datos.
actualizarDatosTsu($fecha,$tarea,$seccion,$eval,$observ,$idlugarTsu,$idTsu,$arrEst,$arrCump)	Modifica los datos de un TSU.

Nombre : m_cuarteleria	
Tipo de Clase : Acceso a Datos	
Nombre	**Descripción**
m_cuarteleria ()	Constructor de la clase, en este caso invocando al constructor de la clase padre MY_Model ().
buscarEstudiantesDadoCadenaBusqueda ($cadena,$cantidadMostrar,$indice)	Retorna una lista con los estudiantes que contengan la cadena de búsqueda en uno de sus atributos en la base de datos.
obtenerCantidadEstudiantesDadoCadenaBusqueda ($cadena)	Devuelve la cantidad de estudiantes que están en el resultado de la búsqueda de estudiantes dado una cadena.
registrarCuarteleria ($idEstudiante,$fechaCuarteleria,$evaluacionCu	Registra una nueva cuartelería en la

arteleria,$observaciones)	base de datos.
actualizarCuarteleria($idCuarteleria,$fechaCuarteleria,$evaluacionCuarteleria,$observaciones)	Actualiza los datos de una cuartelería ya existente.
obtenerPasosEscalera()	Permite buscar todos los pasos de escalera que poseen los edificios de la base de datos.
obtenerPasoEscaleraDadoNroEdficio ($nroEdificio)	Permite conocer los pasos de escalera que posee un edificio determinado.
obtenerEdificios()	Permite conocer todos los edificios existentes en la base de datos.
obtenerEdificioDadoNroApartamento ($nroApartamento)	Permite conocer a que edificio pertenece un apartamento determinado.
obtenerApartamentos()	Devuelve todos los apartamentos existentes en la base de datos.
obtenerBrigadas()	Devuelve todas las brigadas existentes en la base de datos.
buscarCuartelerias ($solapin,$mes,$anno,$edificio,$pasoEscalera,$apto,$brigada,$cantidadMostrar,$indice)	Devuelve las cuartelerías realizadas teniendo en cuenta un mes, un año, un número de solapín, edificio, paso de escalera, apartamento, brigada o la combinación entre algunos de estos parámetros.
obtenerCantidadCuartelerias ($solapin,$mes,$anno,$edificio,$pasoEscalera,$apto,$brigada)	Devuelve la cantidad de cuartelerías que devolvería el método “buscarCuartelerias ()” con el mismo criterio de búsqueda.
eliminarCuarteleria ($idCuarteleria)	Elimina una cuartelería de la base de datos.
obtenerCuarteleriaDadoIdCuarteleria ($idCuarteleria)	Devuelve los datos de una cuartelería dado su identificador.

Nombre : m_guardiaEstudiantil	
Tipo de Clase : Acceso a Datos	
Nombre	**Descripción**
m_guardiaEstudiantil ()	Constructor de la clase, en este caso invocando al constructor de la clase

	padre MY_Model ().
registrarGuardiaEstudiantil($inputfecha,$arrIdTrabGuardia,$inputObservaciones,$arregloIdEstudiantes,$arrIdPostaGuardia,$arrIdTurnoGuardia,$arrIdCumplimiento,$inputbrigada)	Registra una nueva guardia estudiantil en la base de datos.
obtenerBrigadas()	Devuelve todas las brigadas que existen en la base de datos.
obtenerPostas()	Devuelve todas las postas de guardia que existen en la base de datos.
obtenerTurnos()	Devuelve todos los turnos de guardia que existen en la base de datos.
buscarTrabajadorGuardia($cadena)	Retorna una lista con los trabajadores que contengan la cadena de búsqueda en uno de sus atributos en la base de datos.
buscarEstudianteDadoIdBrigada($idBrigada)	Dado el identificador de una brigada devuelve todos los estudiantes que pertenecen a la misma.
mostrarDatosJefeGrupo($idBrigada)	Dado el identificador de una brigada muestra los datos de su jefe de grupo.
mostrarDatosProfesorGuia($idBrigada)	Dado el identificador de una brigada muestra los datos del profesor guía.
mostrarDatosBrigada($idBrigada)	Devuelve todos los datos de una brigada en específico.
mostrarDatosPersona($idPersona)	Devuelve todos los datos de una persona en específico.
obtenerCantidadGuardia($idBrigada,$mes,$anno)	Devuelve la cantidad de guardias realizadas por una brigada, en un mes o año específico; o la combinación entre algunos de estos parámetros.
busquedaAvanzadaGuardia($idBrigada,$mes,$anno,$cantidadMostrar,$indice)	Devuelve las guardias realizadas por una brigada, en un mes o año específico; o la combinación entre algunos de estos parámetros.
buscarDatosGuardia($idGuardia)	Devuelve todos los datos de una guardia en específico.
BuscarTrabajadoresGuardia($idGuardia)	Devuelve los trabajadores que participaron en una guardia específica.

buscarEstudiantesGuardia($idGuardia)	Devuelve los estudiantes que participaron en una guardia específica.
eliminarGuardia($idGuardia)	Elimina una guardia de la base de datos.
actualizarGuardia($idGuardia,$inputfecha,$arrIdTrabGuardia,$inputObservaciones,$arregloIdEstudiantes,$arrIdPostaGuardia,$arrIdTurnoGuardia,$arrIdCumplimiento)	Actualiza los datos de una guardia estudiantil en la base de datos.
existeGuardiaDadoFecha($dia,$mes,$anno)	Devuelve el número de guardias que se realizaron en una fecha específica.

Nombre : m_lugarTsu	
Tipo de Clase : Acceso a Datos	
Nombre	**Descripción**
m_lugarTsu ()	Constructor de la clase, en este caso invocando al constructor de la clase padre MY_Model ().
registrarLugarTsu ($lugar)	Registra un Nuevo lugar para realizar TSU en la base de datos.
existeLugar ($lugar)	Comprueba si existe un lugar de TSU con el mismo nombre que el que se desea insertar.
mostrarLugarTsu()	Devuelve todos los lugares de TSU que existen en la base de datos.
eliminarLugarTsu ($idLugarTsu)	Elimina un lugar de TSU de la base de datos.

Nombre : m_postaGuardia	
Tipo de Clase : Acceso a Datos	
Nombre	**Descripción**
m_postaGuardia ()	Constructor de la clase, en este caso invocando al constructor de la clase padre MY_Model ().

registrarPostaGuardia ($posta)	Registra una nueva posta de guardia.
existePosta ($posta)	Comprueba si existe una posta de guardia en la base de datos con el mismo nombre que la que se desea insertar.
mostrarPostaGuardia()	Devuelve todas las postas de guardia que existen en la base de datos.
eliminarPostaGuardia ($idposta)	Elimina una posta de guardia de la base de datos.

Nombre : m_actualizarBrigada	
Tipo de Clase : Acceso a Datos	
Nombre	**Descripción**
m_actualizarBrigada ()	Constructor de la clase, en este caso invocando al constructor de la clase padre MY_Model ().
actualizarBrigada ($idBrigada,$seccion,$idJefeBrigada,$idProfesorGuia)	Actualiza los datos de una brigada en la base de datos.
buscarBrigadas()	Devuelve todas las brigadas existentes en la base de datos.
esProfesorGuia ($idProfesorGuia)	Devuelve la brigada de la cual un profesor determinado es el guía.
mostrarDatosJefeGrupo ($idBrigada)	Devuelve los datos del jefe de grupo de una brigada determinada.
mostrarDatosProfesorGuia($idBrigada)	Devuelve los datos del profesor guía de una brigada determinada.
mostrarDatosBrigada($idBrigada)	Devuelve los datos de una brigada determinada.
buscarPersona($cadena,$idBrigada,$cantidadMostrar,$indice)	Devuelve una lista con las personas de una brigada determinada que contengan la cadena de búsqueda en uno de sus atributos en la base de datos.
buscarPersonaProf($cadena,$cantidadMostrar,$indice)	Devuelve una lista con los profesores que contengan la cadena de búsqueda en uno de sus atributos en la base de datos.

obtenerCantidadPersonas($cadena,$idBrigada, $valor)	Devuelve la cantidad de personas que aparecerán al realizar una búsqueda de personas con una cadena específica.
mostrarDatosPersona($idPersona)	Devuelve los datos de una persona específica.

Nombre : m_administrar	
Tipo de Clase : Acceso a Datos	
Nombre	**Descripción**
m_administrar ()	Constructor de la clase, en este caso invocando al constructor de la clase padre MY_Model ().
obtenerModulos()	Devuelve los datos de todos los módulos existentes en la base de datos.
registrarModulo ($nombre,$link,$descripcion)	Registra un Nuevo módulo en la base de datos.
eliminarModulo ($idModulo)	Elimina un módulo de la base de datos.
obtenerModuloDadoIdModulo ($idModulo)	Devuelve los datos de un módulo específico.
actualizarModulo ($inputIdModulo,$inputNombreModulo,$inputEnlace,$inputDescrip)	Actualiza los datos de un módulo en la base de datos.
obtenerRoles()	Devuelve los datos de todos los roles que existen en la base de datos.
eliminarRol ($idRol)	Elimina un rol de la base de datos.
registrarRol ($nombre,$descrip,$arrModulos)	Registra un Nuevo rol en la base de datos.
obtenerModulosDadoIdRol ($idRol)	Devuelve los módulos a los que va a tener acceso un rol determinado.
obtenerRolDadoIdRol ($idRol)	Devuelve los datos de un rol específico.
actualizarRol($idRol,$nombre,$descrip,$arrModulos)	Actualiza los datos de un rol en la base de datos.
obtenerUsuarios()	Devuelve todos los usuarios de la base de datos y los roles que tienen

	asignados cada uno.
registrarUsuario($usuario,$rol)	Registra un nuevo usuario en la base de datos.
eliminarUsuario($idUsuario)	Elimina un usuario de la base de datos.

Nombre : m_buscarEstudiante	
Tipo de Clase : Acceso a Datos	
Nombre	**Descripción**
m_buscarEstudiante ()	Constructor de la clase, en este caso invocando al constructor de la clase padre MY_Model ().
buscarEstudiante ($cadena,$cantidadMostrar,$indice)	Retorna una lista con los estudiantes que contengan la cadena de búsqueda en uno de sus atributos en la base de datos.
obtenerCantidadEstudiantes ($cadena)	Devuelve la cantidad de estudiantes que están en el resultado de la búsqueda de estudiantes dado una cadena.

Descripción de las Clases Controladoras.

Nombre: c_RegistrarVisitaApto	
Tipo de Clase: Controladora	
Nombre	**Descripción**
c_RegistrarVisitaApto()	Constructor de la clase, en este caso invocando al constructor de la clase padre MY_Controller (), y se crea también un objeto de la clase modelo m_RegistrarVisitaApto, llamado objRegistVisita.
buscarPersonasDadoCadenaBusqueda()	Hace una llamada al método mostrarBuscarPersonasDadoCadenaBusqueda ($cadenaBusq) con la cadena de búsqueda deseada.
mostrarBuscarPersonasDadoCadenaBusqueda($cadenaBusq)	Retorna en una tabla el resultado de la búsqueda de personas dada una cadena de caracteres.

registrarVisitaApartamento()	Se registra una nueva visita a un apartamento.
getPersona($id)	Devuelve nombre completo y el solapín de la persona con ese identificador.
getAptosDelEdif()	Hace una llamada al método mostrar_getAptosDelEdif ($edificio) con el edificio deseado.
mostrar_getAptosDelEdif($edificio)	Devuelve en un select los apartamentos pertenecientes a un edificio.
busquedaAvanzadaVisitaApto()	Hace una llamada al método mostrarBusquedaAvanzadaVisitaApto($dia,$mes,$anno,$apto,$evaluacion) con los parámetros de búsqueda deseados.
mostrarBusquedaAvanzadaVisitaApto($dia,$mes,$anno,$apto,$evaluacion)	Devuelve en una tabla las visitas realizadas teniendo en cuenta una fecha, un apartamento, una evaluación específica o la combinación entre alguno de estos parámetros; con las opciones de mostrar, modificar y eliminar a cada una.
mostrarDatosVisitaApto()	Hace una llamada al método mostrarDatosVisitaAptoId($idVisita) con el identificador de la visita deseada.
mostrarDatosVisitaAptoId($idVisita)	Devuelve los datos de una visita determinada.
PersonasParticipantesEnLaVisita($idVisita)	Hace una llamada al método mostrarPersonasParticipantesEnLaVisita($idVisita) con el identificador de la visita deseada.
mostrarPersonasParticipantesEnLaVisita($idVisita)	Muestra en un select las personas participantes en una visita dado el identificador de esta.
eliminarVisitaApto($idVisita)	Elimina la visita a un apartamento con posea este identificador.
actualizarVisitaApto($arrPersonas,$fecha,$limpieza,$ordenInterior,$cuidadoMedios,$educacionFormal,$ahorroEnergia,$evalGeneral)	Modifica los datos de una visita a un apartamento.

Nombre: c_TSU	
Tipo de Clase: Controladora	
Nombre	**Descripción**
c_TSU()	Constructor de la clase, en este caso invocando al constructor de la clase padre MY_Controller (), y se crea también un objeto de la clase modelo m_TSU, llamado “objTSU”.
index()	Aquí se levanta como página de inicio “inicioTsu”.
nuevoTSU()	Aquí se van a obtener todas las brigadas y lugares de TSU existentes en la base de datos y se van a mostrar al levantar la página “RegistrarTSU”.
editarTSU()	Aquí se manda a levantar la página “editarTsu” y se van a mostrar en ella los datos del TSU seleccionado con la posibilidad de modificarlos.
listadoTSU()	Aquí se manda a levantar la página “MostrarTSU” en donde el usuario tiene la posibilidad de buscar un TSU determinado.
buscarEstudiantesDadoIdBrigada()	Hace una llamada al método “mostrarBuscarEstudiantesDadoIdBrigada ()” con la brigada deseada.
mostrarBuscarEstudiantesDadoIdBrigada($idBrigada)	Retorna en una tabla el nombre y el identificador de cada uno de los estudiantes de la brigada con la opción de modificar su participación en el TSU.
registrarTSU()	Se registra un Nuevo TSU.
buscarEstudiantesDadoIdTsu($idTsu)	Hace una llamada al método mostrarbuscarEstudiantesDadoIdTsu ($idTsu) con el identificador del TSU deseado.
mostrarbuscarEstudiantesDadoIdTsu()	Muestra en una tabla los estudiantes pertenecientes a la brigada que realizó el TSU dado el identificador de este último, y estarán marcados en un checkbox

	los que cumplieron correctamente la tarea.
eliminarTsu()	Elimina el TSU que posea un identificador determinado en la base de datos y se realiza la búsqueda avanzada de TSU nuevamente con los mismos parámetros que se encontró el recientemente eliminado.
actualizarTSU()	Actualiza los datos de un TSU.
buscarTsu()	Hace una llamada al método "mostrarBusquedaAvanzadaTsu()" con los parámetros deseados.
mostrarBusquedaAvanzadaTsu($idBrigada,$mes,$anno,$idLugar,$evaluacion,$paginaBusq)	Devuelve los TSU realizados teniendo en cuenta un mes, un año, una brigada, una evaluación específica, un lugar o la combinación entre alguno de estos parámetros; con las opciones de mostrar, modificar y eliminar a cada uno.
regresar()	Permite regresar a la página "MostrarTSU.php" en la cual va a estar reflejada la misma búsqueda que habíamos realizado anteriormente.
verTSU()	Muestra en la página "verTSU" los datos del TSU seleccionado.
mostrarbuscarEstudiantesDadoIdTsuVerTSU($idTsu)	Muestra en una tabla los estudiantes pertenecientes a una brigada que realizó un TSU determinado y los que cumplieron con el mismo.

Nombre: c_cuartelería	
Tipo de Clase: Controladora	
Nombre	**Descripción**
c_cuartelería ()	Constructor de la clase, en este caso invocando al constructor de la clase padre MY_Controller (), y se crea también un objeto de la clase modelo

	m_cuartelería, llamado "adCuarteleria".
index()	Aquí se levanta como página de inicio "inicioCuarteleria".
nuevaCuarteleria ()	Aquí se ordena levantar la página "RegistrarCuarteleria", en la cual se va a poder insertar una cuartelería.
mostrarCalendario()	Muestra el calendario para poder elegir una fecha.
calendarioSiguiente()	Muestra en el calendario el mes siguiente.
calendarioAnterior()	Muestra en el calendario el mes anterior.
buscarEstudiantesDadoCadenaBusqueda()	Captura la cadena de búsqueda y llama al método "mostrarBuscarEstudiantesDado CadenaBusqueda()".
mostrarBuscarEstudiantesDadoCadenaBusqueda($cadenaBusq,$paginaBusq)	Muestra en una tabla el nombre completo y el apartamento al que pertenecen los estudiantes que fueron el resultado de la búsqueda, dando la posibilidad de seleccionar solo uno.
registrarCuarteleria()	Registrar una nueva cuartelería.
listarCuarteleria()	Se levanta la página "listadoCuartelerias" para que el usuario pueda buscar una cuartelería determinada.
buscarCuartelerias()	Captura los parámetros de búsqueda y llama al método "mostrarBuscarCuartelerias".
mostrarBuscarCuartelerias ($solapin,$mes,$anno,$edificio,$pasoEscalera,$apto,$brigada,$paginaBusq)	Muestra en una tabla los datos de las cuartelerías devueltas por la búsqueda con la posibilidad de modificar y eliminar cada una de ellas.
eliminarCuarteleria()	Permite eliminar una cuartelería.
editarCuarteleria()	Muestra la página "editarCuarteleria" y en ella los datos de la cuartelería que va a ser modificada.
actualizarCuarteleria()	Actualiza los datos de una cuartelería.

Nombre: c_guardiaEstudiantil	
Tipo de Clase: Controladora	
Nombre	**Descripción**
c_guardiaEstudiantil ()	Constructor de la clase, en este caso invocando al constructor de la clase padre MY_Controller (), y se crea también un objeto de la clase modelo m_guardiaEstudiantil llamado "objGuardiaEstudiantil".
index()	Aquí se levanta como página de inicio "inicioGuardia".
nuevaGuardia()	Aquí se van a obtener todas las brigadas existentes en la base de datos y se van a mostrar al levantar la página "guardiaEstudiantil", para poder insertar una nueva guardia.
listarGuardia()	Aquí se van a obtener todas las brigadas existentes en la base de datos y se van a mostrar al levantar la página "listadoGuardia".
editarGuardia()	Aquí se van a mostrar todos los datos de la guardia que se seleccionó para modificar en la página "editarGuardiaEstudiantil", dando la posibilidad de cambiarlos.
mostrarEstudiantesGuardiaEditarGuardia($idGuardia)	Muestra los estudiantes de una brigada que participaron en una guardia específica, y el comportamiento de cada uno durante la misma (turno, posta y cumplimiento), dándole la posibilidad al usuario de modificar estos datos.
registrarGuardia()	Inserta una nueva guardia.
buscarEstudianteDadoIdBrigada()	Hace una llamada al método "mostrarBuscarEstudiantesDadoIdBrigada ()" con la brigada deseada.
mostrarBuscarEstudiantesDadoIdBrigada($idBrigada)	Muestra todos los estudiantes de una brigada con la posibilidad de registrar su labor en la guardia (turno, posta y cumplimiento).
buscarTrabajadoresGuardia()	Hace una llamada al método "mostrarBuscarTrabajadoresGuardia ()" con la cadena deseada.

mostrarBuscarTrabajadoresGuardia($cadena)	Muestra todos los trabajadores que contengan en sus atributos la cadena de búsqueda.
cargarDatosProfGuiaJefeGrupo()	Retorna los datos del jefe de grupo y del profesor guía de una brigada específica.
busquedaAvanzadaGuardia()	Hace una llamada al método "mostrarBusquedaAvanzadaGuardia()" con los parámetros deseados.
mostrarBusquedaAvanzadaGuardia($idBrigada,$mes,$anno,$paginaBusq)	Devuelve las guardias realizadas en un mes, un año, por una brigada específica, o la combinación entre alguno de estos parámetros; con las opciones de mostrar, modificar y eliminar a cada una.
mostrarTrabajadoresGuardiaVerGuardia($idGuardia)	Retorna los trabajadores que participaron en una guardia específica.
mostrarListadoEstudiantesGuardiaVerGuardia($idGuardia)	Muestra los estudiantes de una brigada que participaron en una guardia específica, y el comportamiento de cada uno durante la misma (turno, posta y cumplimiento), los datos no pueden ser modificados.
verDatosGuardia()	Permite ir a la página "verGuardia" y que se muestren en esta los datos de la guardia seleccionada.
regresar()	Permite regresar a la página "listadoGuardia" en la cual va a estar reflejada la misma búsqueda que habíamos realizado anteriormente.
eliminarGuardia()	Elimina una guardia estudiantil.
actualizarGuardia()	Actualiza los datos de una guardia.
existeGuardiaDadoFecha()	Retorna la cantidad de guardias que se realizaron en una fecha específica.

Nombre: c_lugarTsu	
Tipo de Clase: Controladora	
Nombre	**Descripción**
c_lugarTsu ()	Constructor de la clase, en este caso invocando al constructor de la clase padre MY_Controller (), y se crea también un objeto de la clase modelo m_ lugarTsu llamado “objLugarTsu”.
index()	Aquí se levanta como página de inicio “MostrarLugarTSU”, y en ella se van a mostrar todos los lugares de TSU que ya existen.
registrarLugarTsu()	Aquí se va a registrar un nuevo lugar de TSU.
eliminarLugarTsu ($idLugarTsu)	Aquí se va a eliminar un lugar de TSU.

Nombre: c_postaGuardia	
Tipo de Clase: Controladora	
Nombre	**Descripción**
c_postaGuardia ()	Constructor de la clase, en este caso invocando al constructor de la clase padre MY_Controller (), y se crea también un objeto de la clase modelo m_postaGuardia llamado “objPostaGuardia”.
index()	Aquí se levanta como página de inicio “mostrarPostasGuardia”, y en ella se van a mostrar todas las postas de guardia que ya existen.
registrarPostaGuardia()	Aquí se va a registrar una nueva posta de guardia.
eliminarPostaGuardia($idPosta)	Aquí se va a eliminar una posta de guardia.

Nombre: c_actualizarBrigada	
Tipo de Clase: Controladora	
Nombre	**Descripción**
c_actualizarBrigada ()	Constructor de la clase, en este caso invocando al constructor de la clase padre MY_Controller (), y se crea también un objeto de la clase modelo m_actualizarBrigada llamado “objActualizarBrigada”.
index()	Aquí se levanta como página de inicio “actualizarBrigada”, y en ella se van a mostrar todas las brigadas que existen.
buscarPersona()	Hace una llamada al método mostrarBuscarPersona() en dependencia de si se va a buscar un profesor, o una persona de una brigada determinada.
mostrarBuscarPersona($cadena,$valor,$idBrigada,$paginaBusq)	Muestra el resultado de la búsqueda de un profesor, o de una persona de una brigada, con la posibilidad de marcar a uno solo como profesor guía o jefe de grupo respectivamente.
cargarDatosBrigada()	Muestra los datos de una brigada con la posibilidad de que sean modificados.

Nombre: c_administrar	
Tipo de Clase: Controladora	
Nombre	**Descripción**
c_administrar ()	Constructor de la clase, en este caso invocando al constructor de la clase padre MY_Controller (), y se crea también un objeto de la clase modelo m_administrar llamado “adAdministrar”.

index()	Aquí se levanta como página de inicio “modulo”, y en ella se van a mostrar todos los módulos que existen en la base de datos.
registrarModulo()	Inserta un Nuevo módulo.
eliminarModulo()	Elimina un módulo.
editarModulo()	Aquí se van a mostrar todos los datos del módulo que se seleccionó para modificar en la página “editarModulo”, dando la posibilidad de cambiarlos.
actualizarModulo()	Aquí se actualiza un módulo y se muestra la página “modulo”.
inicioRol()	Aquí se van a mostrar todos los roles que existen con sus datos en la página “rol”, dando la posibilidad de editar o eliminar cada uno de ellos, y de insertar un nuevo rol.
eliminarRol()	Elimina un rol.
nuevoRol()	Aquí se levanta como página de inicio “nuevoRol”, y en ella se van a mostrar todos los módulos que existen en la base de datos, dando la posibilidad de marcar a los que se desea tenga acceso el nuevo rol.
editarRol()	Aquí se van a mostrar todos los datos del rol que se seleccionó para modificar en la página “editarRol”, dando la posibilidad de cambiarlos.
registrarRol()	Inserta un nuevo rol.
actualizarRol()	Actualiza los datos de un rol.
inicioUsuario()	Aquí se van a mostrar todos los usuarios que existen con sus datos en la página “usuario”, dando la posibilidad de eliminar cada uno de ellos, y de insertar uno nuevo.
registrarUsuario()	Inserta un nuevo usuario.
eliminarUsuario()	Elimina un usuario.

Nombre: c_buscarEstudiante	
Tipo de Clase: Controladora	
Nombre	**Descripción**
c_buscarEstudiante	Constructor de la clase, en este caso invocando al constructor de la clase padre MY_Controller (), y se crea también un objeto de la clase modelo m_buscarEstudiante llamado "objbuscarEstudiante".
index()	Aquí se levanta como página de inicio "buscarEstudiante".
buscarEstudiante()	Captura la cadena de búsqueda y llama al método "mostrarbuscarEstudiente ()".
mostrarbuscarEstudiente ($cadena,$paginaBusq)	Muestra en una tabla el nombre completo, el usuario del dominio, el solapín, la brigada, el edificio, el apartamento y el número de teléfono de los estudiantes que fueron el resultado de la búsqueda.

Descripción de las Clases de Interfaz.

Caso de Uso : Gestionar Visitas a Apartamentos	
Nombre	**Descripción**
Index.php	Página de inicio de la aplicación. Permite la autenticación de los usuarios.
MostrarVisitaApto.php	Página en la que se va a poder buscar una visita determinada. Además se podrán ver sus datos, modificarlos y eliminar la visita si se desea.
RegistrarVisitaApto.php	Página en la que se va a poder registrar una nueva visita.
msg.php	Página que se va a mostrar luego de insertar una visita a un apartamento, y en ella se le informa al usuario si la operación tuvo éxito o no.
Caso de Uso : Gestionar Postas de Guardia	
Nombre	**Descripción**
mostrarPostasGuardia.php	Página que muestra todas las postas de guardia, y posibilita insertar una nueva o eliminar una ya existente.
Caso de Uso : Gestionar Guardia Estudiantil	

Nombre	Descripción
inicioGuardia.php	Muestra una breve descripción del CU.
editarGuardiaEstudiantil.php	Página en la que se van a poder modificar los datos de una guardia estudiantil.
guardiaEstudiantil.php	Página en la que se va a poder insertar una guardia estudiantil.
listadoGuardia.php	Página en la que se va a poder localizar una guardia específica a través de una búsqueda.
verGuardia.php	Página en la que se van a mostrar los datos de la guardia que se ha consultado.
msg.php	Página que se va a mostrar luego de insertar una guardia estudiantil, y en ella se le informa al usuario si la operación tuvo éxito o no.
Caso de Uso : Gestionar Cuartelería	
Nombre	**Descripción**
inicioCuarteleria.php	Muestra una breve descripción del CU.
RegistrarCuarteleria.php	Página que permite insertar una nueva cuartelería.
editarCuarteleria.php	Página que permite modificar los datos de una cuartelería.
listadoCuartelerias.php	Página en la que se va a poder localizar una cuartelería específica a través de una búsqueda.
msg.php	Página que se va a mostrar luego de insertar una cuartelería, y en ella se le informa al usuario si la operación tuvo éxito o no.
Caso de Uso : Gestionar TSU	
Nombre	**Descripción**
inicioTSU.php	Muestra una breve descripción del CU.
verTSU.php	Página en la que se van a mostrar los datos del TSU que se ha consultado.
RegistrarTSU.php	Página que permite insertar un nuevo TSU.
MostrarTSU.php	Página en la que se va a poder localizar un TSU específico a través de una búsqueda.
EditarTsu.php	Página que permite modificar los datos de un TSU.
msg.php	Página que se va a mostrar luego de insertar o modificar un TSU, y en ella se le informa al usuario si la operación tuvo éxito o no.

Caso de Uso : Gestionar Lugar de TSU	
Nombre	**Descripción**
MostrarLugarTSU.php	Página que muestra todos los lugares en donde se realizan los TSU, y posibilita insertar uno nuevo o eliminar uno ya existente.
Caso de Uso : Buscar Estudiantes	
Nombre	**Descripción**
buscarEstudiante.php	Página que permite conocer los datos de un estudiante determinado a través de una búsqueda.
Caso de Uso : Gestionar Brigadas	
Nombre	**Descripción**
actualizarBrigada.php	Página que permite asignarle a una brigada una sección de clases, un nuevo jefe de grupo y un nuevo profesor guía.
Caso de Uso : Gestionar Rol	
Nombre	**Descripción**
rol.php	Página que muestra los datos de todos los roles existentes; con la posibilidad de modificarlos o eliminarlos.
nuevoRol.php	Página en la que se puede insertar un nuevo rol.
editarRol.php	Página en la que se pueden modificar los datos de un rol.
Caso de Uso : Gestionar Módulos	
Nombre	**Descripción**
modulo.php	Página que muestra los datos de todos los módulos existentes; con la posibilidad de modificarlos, eliminarlos o insertar uno nuevo.
editarModulo.php	Página en la que se pueden modificar los datos de módulo.
Caso de Uso : Gestionar Usuarios	
Nombre	**Descripción**
usuario.php	Página que muestra los datos de todos los usuarios existentes; con la posibilidad de eliminarlos o insertar uno nuevo.

3.3.3 Diagrama de Clases Persistentes.

Las clases persistentes son aquellas que la información que encapsulan persiste en el tiempo, lo cual está dado por el almacenamiento físico de la información de la clase. Dichas clases poseen algún atributo propio, y estarán disponibles ya sea para la copia de seguridad en caso de que ocurra algún fallo en el sistema, o para intercambiar información.

Diagrama de Clases Persistentes.

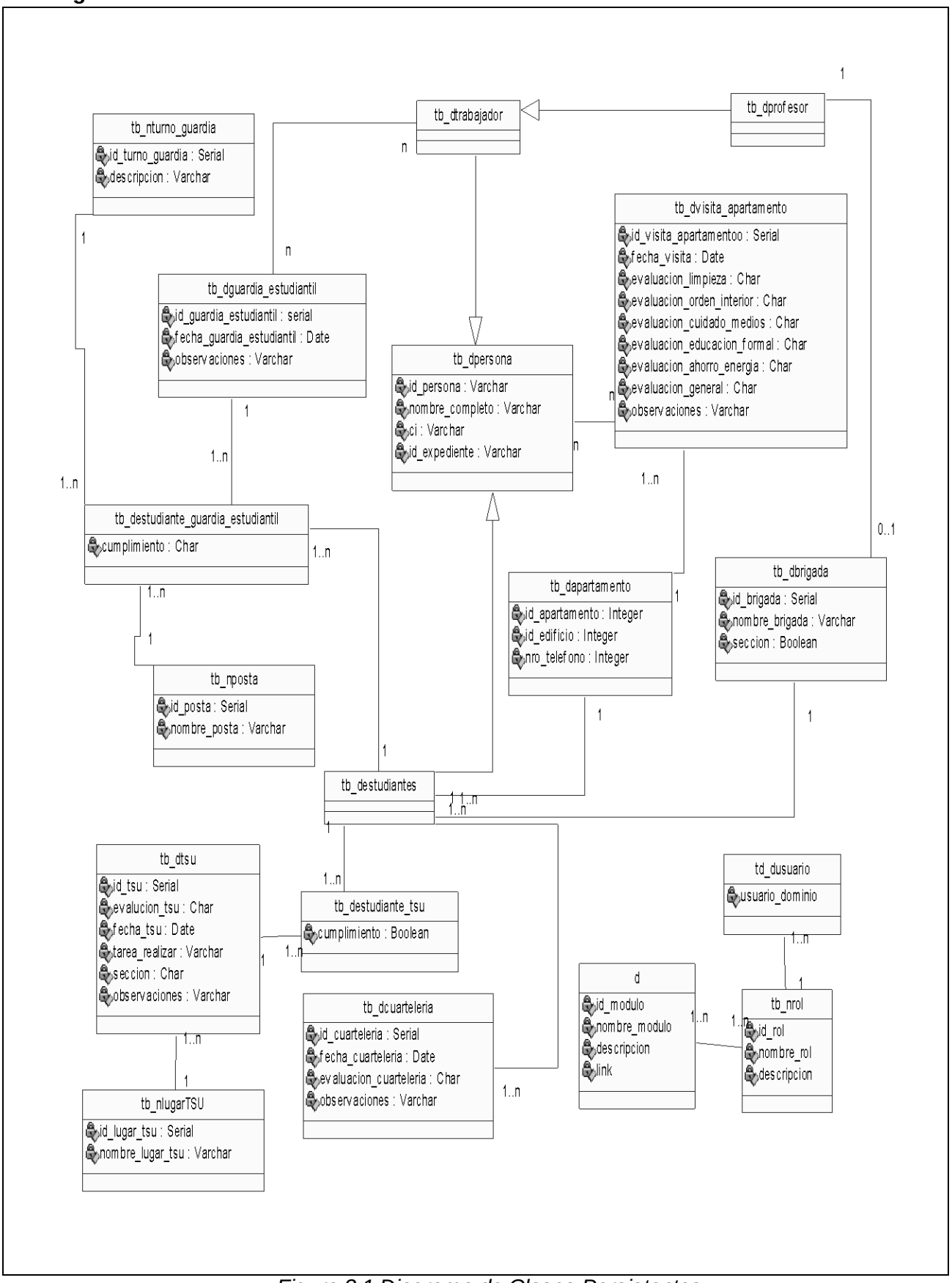


Figura 3.1 Diagrama de Clases Persistentes.

3.3.4 Diseño de la Base de Datos.

En el diagrama de base de datos se representan los atributos de las tablas de la base de datos, así como la distribución física de las mismas y las relaciones que existen entre ellas. En el Anexo 2 se encuentra el diagrama de la Base de Datos debido a su extensión.

3.3.5 Descripción de las Tablas de la Base de Datos.

Tabla tb_dapartamento

Nombre: Tabla tb_dapartamento		
Descripción: Esta tabla almacena los datos los apartamentos de la residencia.		
Atributo	**Tipo**	**Descripción**
id_apartamento	integer	Llave primaria, identificador del apartamento.
nro_edificio	integer	Edificio al que pertenece el apartamento.
nro_telefono	integer	Número de teléfono que posee el apartamento.

Tabla tb_dbrigada

Nombre: tb_dbrigada		
Descripción: Esta tabla almacena los datos de las brigadas de la facultad.		
Atributo	**Tipo**	**Descripción**
id_brigada	varchar	Llave primaria, identificador de la brigada.
nombre_brigada	varchar	Nombre de la brigada.
seccion	boolean	Sección en que da clases la brigada.
id_jefe_brigada	varchar	Llave foránea, identificador del jefe de brigada.
id_profesor_guia	varchar	Llave foránea, identificador del profesor guía.

Tabla tb_dcuarteleria

Nombre: tb_dcuarteleria		
Descripción: Esta tabla almacena las cuartelerías que se han realizado.		
Atributo	**Tipo**	**Descripción**
id_cuarteleria	integer	Llave primaria, identificador de la cuartelería.
observaciones	varchar	Observaciones realizadas a la cuartelería.
fecha_cuarteleria	date	Fecha en que se realizó la cuartelería.
evaluacion_cuarteleria	char	Evaluación que se le dio a la

		cuartelería.
id_estudiante	varchar	Llave foránea, identificador del estudiante que realizó la cuartelería.

Tabla tb_destudiante

Nombre: tb_destudiante		
Descripción: Esta tabla almacena los datos de los estudiantes de la facultad.		
Atributo	**Tipo**	**Descripción**
id_estudiante	varchar	Llave primaria, identificador del estudiante.
id_brigada	varchar	Llave foránea, identificador de la brigada a la que pertenece.
id_apartamento	integer	Llave foránea, identificador del apartamento al que pertenece.

Tabla tb_destudiante_guardia_estudiantil

Nombre: tb_destudiante_guardia_estudiantil		
Descripción: Esta tabla almacena las guardias realizadas, y los estudiantes que participaron en ella con su ubicación, turno de guardia y evaluación.		
Atributo	**Tipo**	**Descripción**
id_estudiante	varchar	Llave primaria, identificador del estudiante que participó en la guardia estudiantil.
id_guardia_estudiantil	integer	Llave primaria, identificador de una guardia.
id_posta	integer	Llave foránea, identificador de la posta en que realizó la guardia.
id_turno_guardia	integer	Llave foránea, identificador del turno en que realizó la guardia.
cumplimiento_guardia	char	Cumplimiento con la guardia.

Tabla tb_destudiante_tsu

Nombre: tb_destudiante_tsu		
Descripción: Esta tabla almacena los TSU realizados, y los estudiantes que participaron en ellos con su cumplimiento.		
Atributo	**Tipo**	**Descripción**
id_tsu	varchar	Llave primaria, identificador de un TSU.
id_estudiante	varchar	Llave primaria, identificador del estudiante que participó en el TSU.
cumplimiento_tsu	boolean	Cumplimiento con el TSU.

Tabla tb_dguardia_estudiantil

Nombre: tb_dguardia_estudiantil		
Descripción: Esta tabla almacena los datos de las guardias estudiantiles.		
Atributo	**Tipo**	**Descripción**
id_guardia_estudiantil	integer	Llave primaria, identificador de la guardia.
fecha_guardia_estudiantil	date	Fecha en que se realizó la guardia estudiantil.
observaciones	varchar	Observaciones que se realizaron en la guardia.

Tabla tb_dmodulo

Nombre: tb_dmodulo		
Descripción: Esta tabla almacena los datos de los módulos.		
Atributo	**Tipo**	**Descripción**
id_modulo	integer	Llave primaria, identificador del módulo
nombre_modulo	varchar	Nombre del módulo.
descripción	varchar	Breve descripción del módulo.
link	varchar	Vínculo del módulo.

Tabla tb_dpersona

Nombre: tb_dpersona		
Descripción: Esta tabla almacena los datos de las personas.		
Atributo	**Tipo**	**Descripción**
id_persona	varchar	Llave primaria, identificador de la persona
nombre_completo	varchar	Nombre y dos apellidos de la persona.
ci	varchar	Carnet de identidad de la persona.
id_expediente	varchar	Número de solapín del estudiante.
usuario_dominio	varchar	Usuario del dominio UCI que tiene la persona.

Tabla tb_dprofesor

Nombre: tb_dprofesor		
Descripción: Esta tabla almacena los datos de los profesores.		
Atributo	**Tipo**	**Descripción**
id_profesor	varchar	Llave primaria, identificador del profesor.

Tabla tb_dtrabajador

Nombre: tb_dtrabajador		
Descripción: Esta tabla almacena los datos de los trabajadores		
Atributo	**Tipo**	**Descripción**
id_trabajador	varchar	Llave primaria, identificador del trabajador.

Tabla tb_dtsu

Nombre: tb_dtsu		
Descripción: Esta tabla almacena los datos de las TSU.		
Atributo	**Tipo**	**Descripción**
id_tsu	integer	Llave primaria, identificador del TSU.
evaluacion_tsu	char	Evaluación del TSU.
fecha_tsu	date	Fecha en que se realizó el TSU.
tarea_realizar	varchar	Actividad que se realizó en el TSU
seccion	boolean	Sección en la que se realizó el TSU.
observaciones	varchar	Observaciones que se realizaron del TSU.
id_lugar_tsu	integer	Llave foránea, identificador del lugar donde se realizó el TSU.
id_brigada	varchar	Llave foránea, identificador de la brigada que realizó el TSU.

Tabla tb_dusuario

Nombre: tb_dusuario		
Descripción: Esta tabla almacena los datos de los usuarios del dominio registrados.		
Atributo	**Tipo**	**Descripción**
usuario_dominio	varchar	Llave primaria, identificador del usuario.
id_rol	integer	Llave foránea, identificador del rol que va a tener asignado el usuario.

Tabla tb_dvisita_apartamento

Nombre: tb_dvisita_apartamento		
Descripción: Esta tabla almacena los datos de las visitas realizadas a los apartamentos.		
Atributo	**Tipo**	**Descripción**
id_visita_apartamento	integer	Llave primaria, identificador de la visita.
id_apartamento	integer	Llave foránea, identificador del apartamento al que se le hizo la visita.
fecha_visita	date	Fecha en que se realizó la visita.
evaluacion_limpieza	char	Evaluación que obtuvo el apartamento por la limpieza.
evaluacion_orden_interior	char	Evaluación que obtuvo el apartamento por el orden interior.
evaluacion_cuidado_medios	char	Evaluación que obtuvo el apartamento por el cuidado de los medios.
evaluacion_educacion_formal	char	Evaluación que obtuvo el apartamento por la educación

		formal de sus estudiantes.
evaluacion_ahorro_energía	char	Evaluación que obtuvo el apartamento por el ahorro de energía.
evaluacion_general	char	Evaluación que es calculada a partir de las demás evaluaciones.
observaciones	varchar	Observaciones que se realizaron en la visita.

Tabla tb_nlugar_tsu

Nombre: tb_nlugar_tsu		
Descripción: Esta tabla almacena los datos de los lugares en donde se realizan los TSU.		
Atributo	**Tipo**	**Descripción**
id_lugar_tsu	integer	Llave primaria, identificador del lugar de TSU.
nombre_lugar_tsu	varchar	Nombre del lugar

Tabla tb_nposta

Nombre: tb_nposta		
Descripción: Esta tabla almacena los datos de las postas en donde se realizan las guardias estudiantiles.		
Atributo	**Tipo**	**Descripción**
id_posta	integer	Llave primaria, identificador de la posta de guardia.
nombre_posta	varchar	Nombre de la posta de guardia.

Tabla tb_nrol

Nombre: tb_nrol		
Descripción: Esta tabla almacena los datos de los roles que van a desempeñar los usuarios.		
Atributo	**Tipo**	**Descripción**
id_rol	integer	Llave primaria, identificador del rol.
nombre_rol	varchar	Nombre del rol.
descripción	varchar	Breve descripción de cada rol.

Tabla tb_nturno_guardia

Nombre: tb_nturno_guardia		
Descripción: Esta tabla almacena los datos de los turnos en que se realizan las guardias estudiantiles.		
Atributo	**Tipo**	**Descripción**
id_turno_guardia	integer	Llave primaria, identificador del turno de guardia.
descripción	varchar	Breve descripción de cada posta de guardia.

3.3.6 Principios de Diseño.

Al realizar una aplicación Web el diseño es una parte importante para lograr el éxito de la misma. Esto no se refiere exclusivamente a la apariencia estética, la combinación de colores o imágenes de forma acertada; sino que incluye además una buena navegabilidad, usabilidad y una adecuada distribución de los contenidos en las páginas. De todo esto depende que el usuario se sienta a gusto mientras navega por el sitio y logre su propósito.
Con el objetivo de lograr la mayor aceptación del sistema posible por parte de los usuarios, a la hora de diseñar se han tenido en cuenta los siguientes principios:

3.3.7 Seguridad

Para garantizar la seguridad del sistema y la de los datos a los que se tiene acceso en él, la aplicación cuenta con un sistema de autenticación de uso obligatorio para su posterior utilización. En dicho mecanismo el individuo deberá estar registrado en la base de datos del sistema, e insertar su usuario y contraseña del dominio UCI para poder acceder. Una vez autenticado sólo va a tener acceso a los módulos que el rol que tenga asignado dicho usuario le permitan, garantizando de esta forma que no realice ninguna acción fuera de su alcance.
Fueron validados los campos de los formularios para que los usuarios no entren datos que puedan poner en peligro al sistema.

3.3.8 Tratamiento de errores

El tratamiento de errores en un sistema influye directamente sobre la calidad del mismo, por esa razón en nuestro trabajo los hemos manejado de dos formas diferentes, y en ambas se persigue el correcto funcionamiento de la aplicación.
La primera se realiza a través del lenguaje JavaScript en la página cliente, mostrando diferentes mensajes de alerta en dependencia del error. Aplicando esto, garantizamos que los datos no lleguen a la página servidora, y que el usuario conozca de la situación actual de la operación. La segunda forma tiene lugar en las páginas servidoras, esto ocurre cuando es solicitada por parte del usuario una operación donde los datos son correctos, pero que sigue sin ser válida por determinadas reglas del negocio, en este caso es la página servidora la encargada de manejar el error y de informarle al usuario el estado de la operación.

3.3.9 Interfaz

La interfaz de la aplicación está confeccionada de la forma Cabecera-Navegador-Contenido-Pie de Página, esto se hizo para lograr un diseño consistente. La cabecera está ubicada en la esquina superior derecha del Sistema, y en ella se muestra el logo y el nombre de nuestra aplicación, puesto que es ahí a donde primero el usuario dirige la vista según estudios realizados.

El navegador se encuentra justamente en la parte inferior de la cabecera, y en él es que el usuario va a tener la posibilidad de seleccionar cada acción que desee realizar, permitiéndole una navegabilidad completa, dentro de sus posibilidades según su rol.

El contenido es lo que está a continuación del menú en orden descendente, y en él se van a mostrar todos formularios, componentes e informaciones necesarios para que el usuario realice sus funciones en el sistema.

El Pie de Página se encuentra al final de la interfaz, y en ella vamos a encontrar diferentes vínculos a sitios relevantes, y los contactos de los creadores de la aplicación, ver figura 3.2.

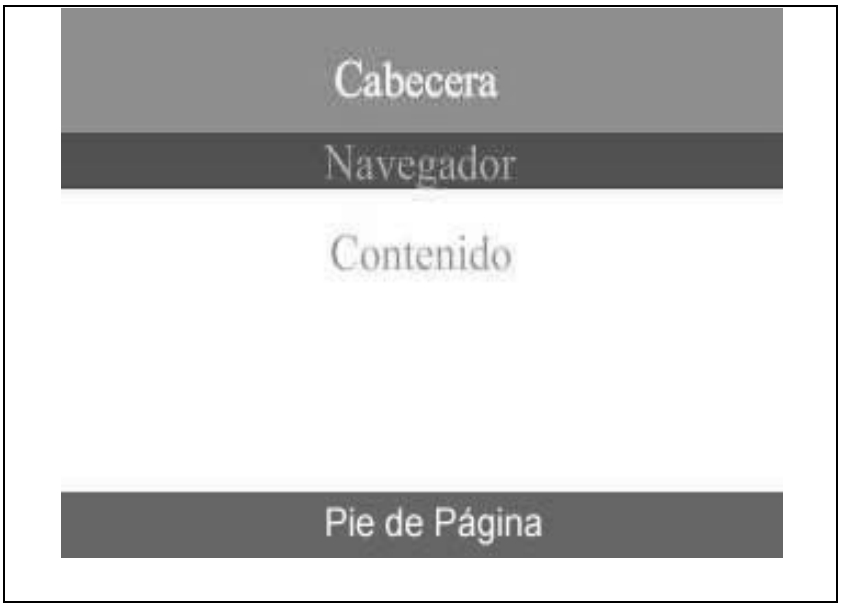


Figura 3.2 Esquema de la Interfaz.

La interfaz está vinculada con una hoja de estilos que ofrece colores, tamaño y tipo de fuente, en dependencia de la importancia de la información, tanto a tablas, vínculos, como texto. Predominan tonalidades de color verde claro combinadas con el color blanco para lograr una interfaz agradable a la vista del usuario, y se utilizó la menor cantidad de imágenes posibles para evitar las demoras en las peticiones al servidor.

La interfaz del sistema fue diseñada teniendo en cuenta los siguientes aspectos:

- Garantizar el funcionamiento de los vínculos.
- Garantizar la legibilidad de la aplicación, y el contraste entre el color de fondo y las letras, así como el tamaño adecuado de las mismas.
- Que le sean visibles al usuario sólo las opciones a las que tiene privilegio de uso.
- Facilitarle a los usuarios sus objetivos, exigiendo de ellos un mínimo de esfuerzo.
- Permitirle al usuario deshacer acciones fácilmente.
- La utilización de un mismo formato y estilo en cada una de las páginas.

3.3.10 Concepción de ayuda

La ayuda estará presente en cada página de la aplicación que lo requiera, y está concebido de esta forma para poder brindarle al usuario la información que necesite según las opciones a las que está accediendo. En cada página de ayuda se abordará sobre el caso de uso específico con que esté interactuando en ese momento, así como una explicación de los posibles mensajes de error que puede darle el sistema ante cualquier acción que se ejecute.

3.4 Conclusiones del Capítulo

En este Capítulo se desarrolló la modelación del sistema empleando para ello, los diagramas de clases del análisis y del diseño, dando a conocer la lógica del sistema que se implementó. Se realiza una breve descripción de la responsabilidad de las clases de acceso a datos, de las controladoras y de las interfaces con que cuenta el sistema, así como también la descripción de las tablas de la base de datos. Se confecciona el diagrama de clases persistentes y el modelo de datos. Por último fueron expuestos los principios de diseño seguidos durante la implementación del sistema, entre los que se encuentran: Interfaz de usuario, Tratamiento de errores, Seguridad y Ayuda.

CAPÍTULO 4 IMPLEMENTACIÓN

4.1 Introducción

En el presente Capítulo se pasará a la implementación del sistema basándose en lo expuesto en el capítulo anterior, aquí se muestra el diagrama de despliegue en el cual se representan los nodos del sistema, y para esto se tuvieron en cuenta las características y funcionalidades que debe cumplir el sistema. Se muestra también en el modelo de implementación los diagramas de los componentes que contiene la aplicación, en los cuales podemos apreciar por separado los componentes necesarios para la implementación de cada caso de uso.

4.2 Modelo de Despliegue.

En el diagrama de despliegue se muestran las relaciones físicas que existen entre los componentes de hardware y de software, es decir la configuración de los elementos de procesamiento en el tiempo que se ejecuta la aplicación. En dicho diagrama se muestra una PC Cliente en donde el usuario accede a la aplicación, una PC Servidor Web que es donde estará situada la aplicación, una PC Servidor de Base de Datos en donde se encontrará la base de datos del sistema y una impresora para que el usuario tenga la posibilidad de imprimir los reportes efectuados por la aplicación.

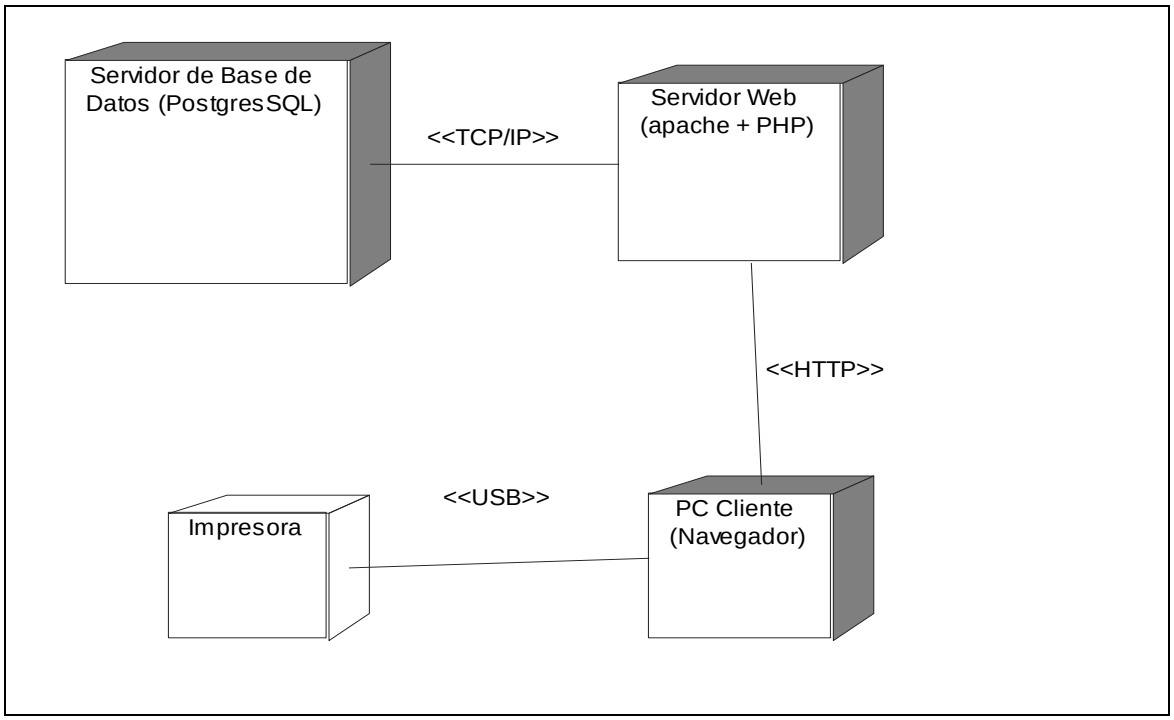


Figura 4.1 Diagrama de Despliegue.

4.3 Modelo de Implementación.

En el modelo de implementación se describe cómo es que los elementos del diseño son implementados en términos de componentes. Estos componentes incluyen: ficheros ejecutables, ficheros de código fuente, y todo otro tipo de ficheros necesarios para la implantación y despliegue del sistema.

4.3.1 Diagrama de componentes del sistema.

En el siguiente diagrama se muestra la relación que existe entre todos los paquetes de componentes del sistema.

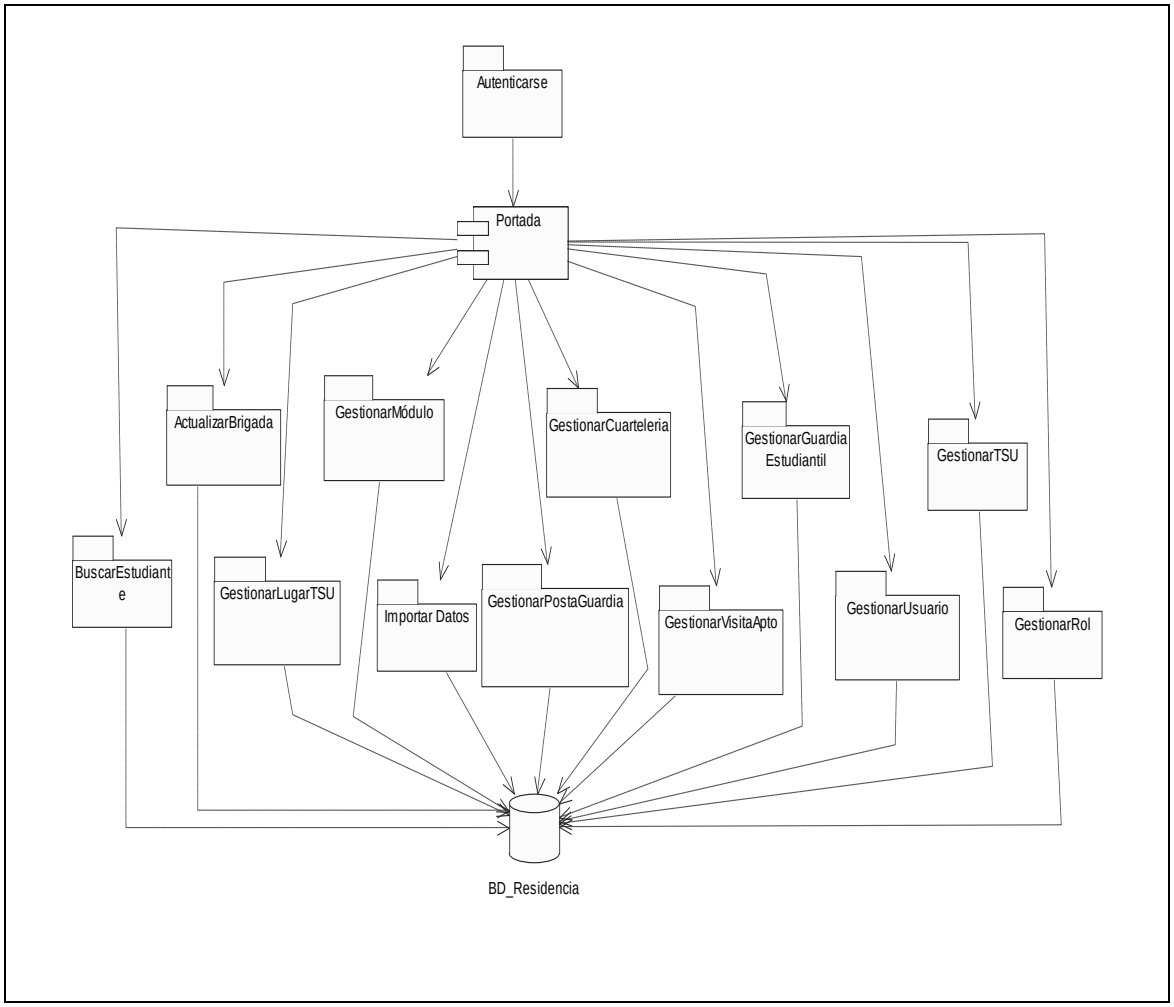


Figura 4.1 Diagrama de Componentes del Sistema.

4.3.2 Diagramas de compontes por casos de uso.

Diagrama de componentes del caso de uso "Gestionar cuartelería".

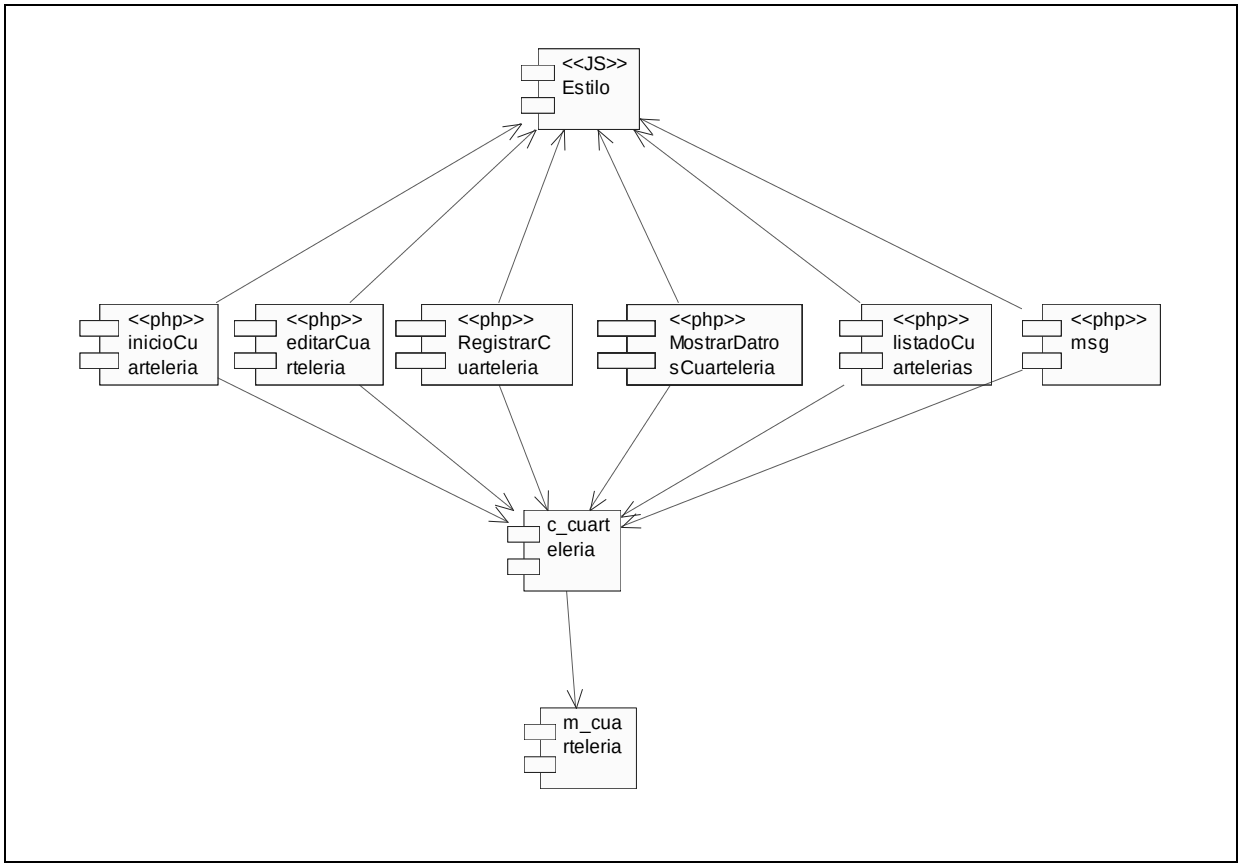


Figura 4.2 Diagrama de componentes del caso de uso "Gestionar cuartelería".

Diagrama de componentes del caso de uso "Gestionar guardia estudiantil".

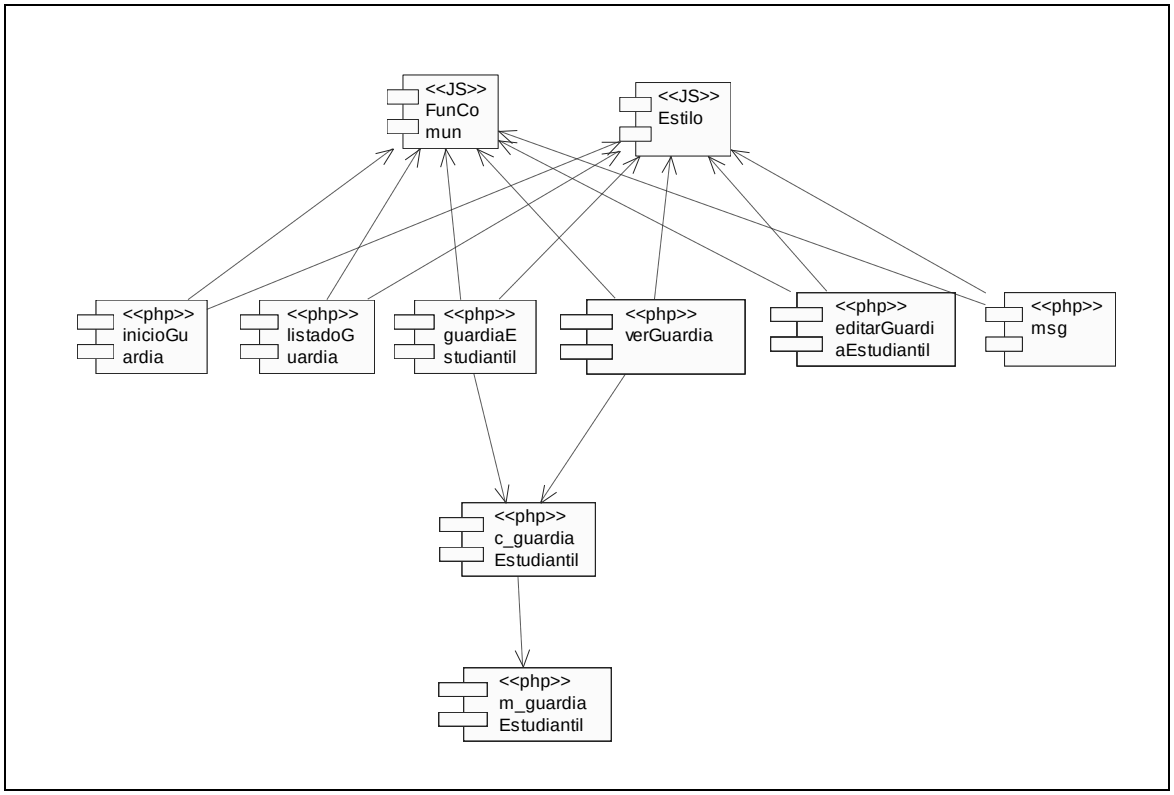


Figura 4.3 Diagrama de componentes del caso de uso "Gestionar guardia estudiantil".

Diagrama de componentes del caso de uso "Gestionar trabajo socialmente útil".

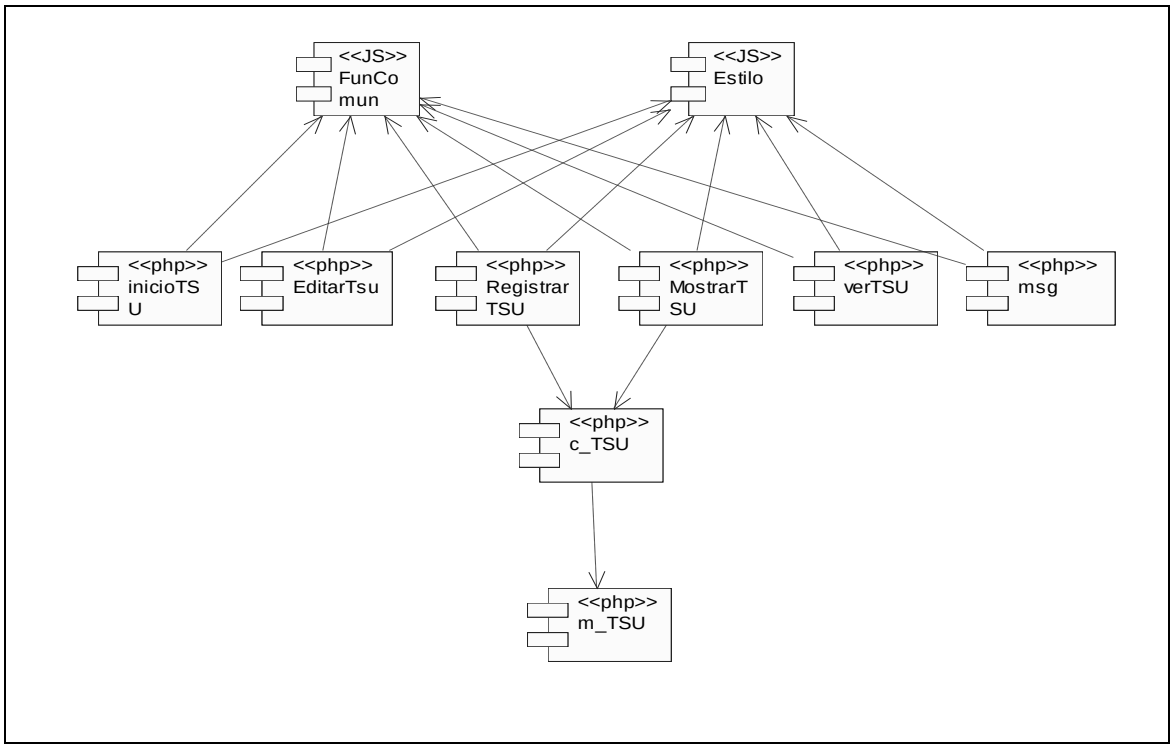


Figura 4.4 Diagrama de componentes del caso de uso "Gestionar trabajo socialmente útil".

Diagrama de componentes del caso de uso "Gestionar usuario".

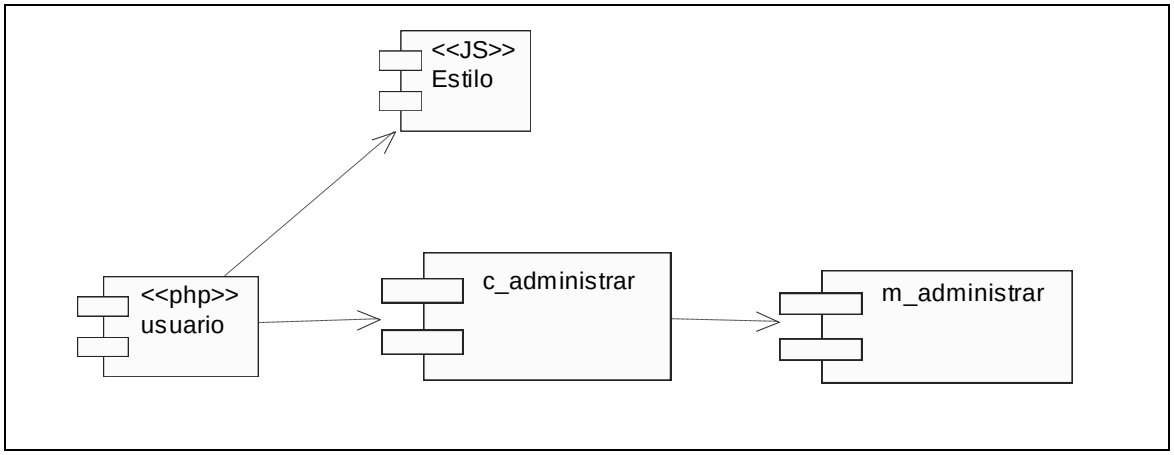


Figura 4.5 Diagrama de componentes del caso de uso "Gestionar usuario".

Diagrama de componentes del caso de uso "Gestionar rol".

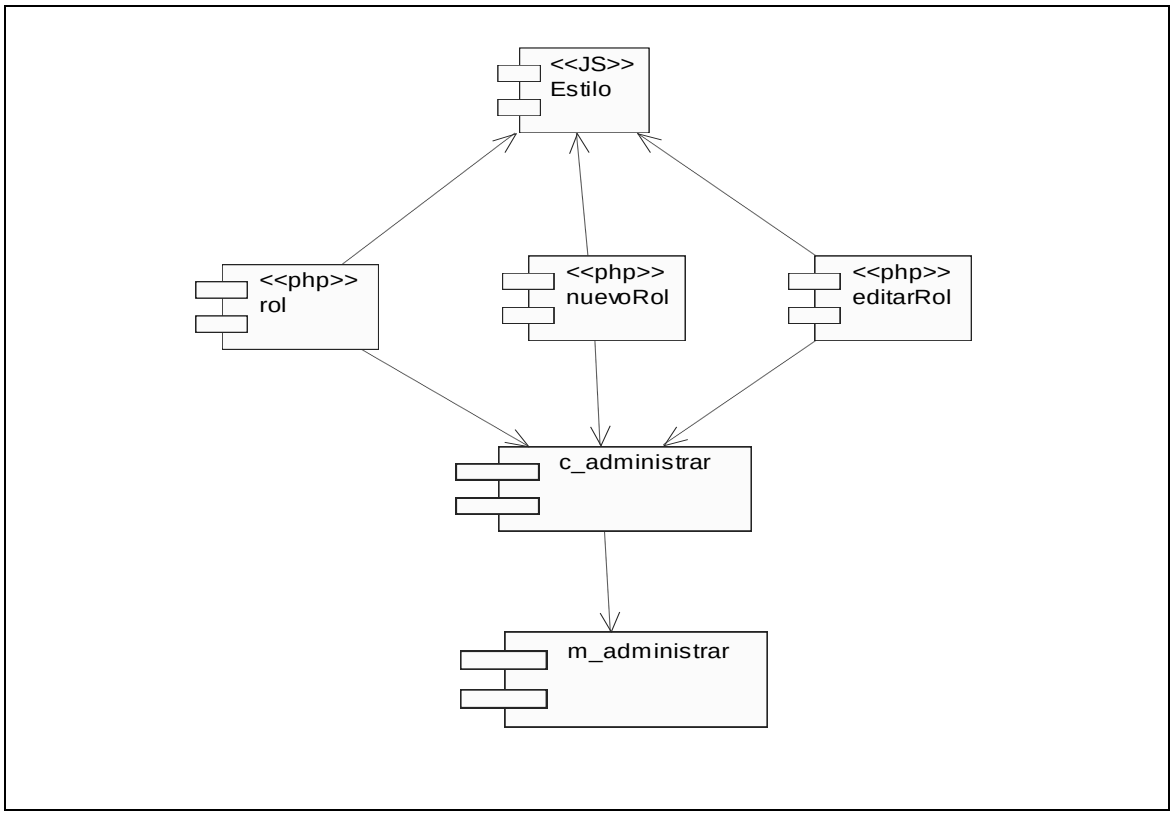


Figura 4.6 Diagrama de componentes del caso de uso "Gestionar rol".

Diagrama de componentes del caso de uso "Buscar estudiante".

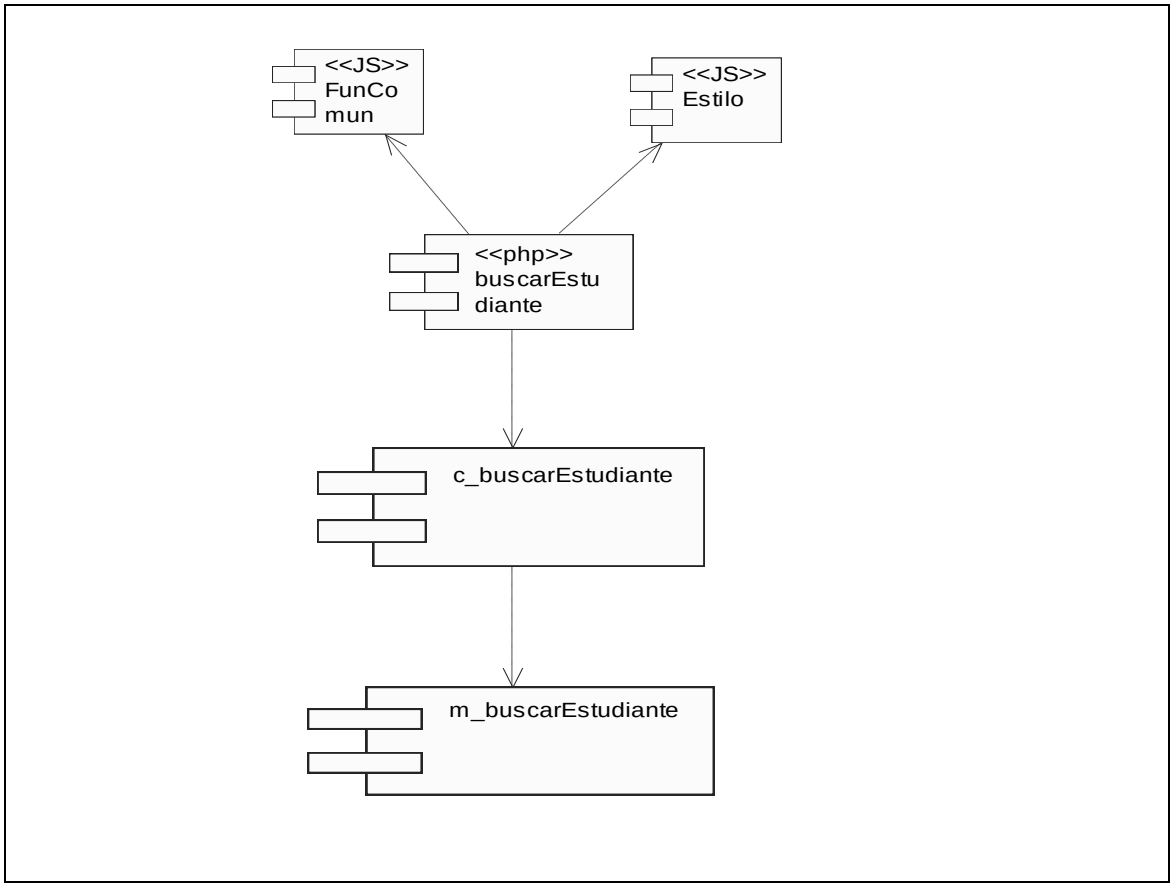


Figura 4.7 Diagrama de componentes del caso de uso "Buscar estudiante".

Diagrama de componentes del caso de uso "Gestionar módulo".

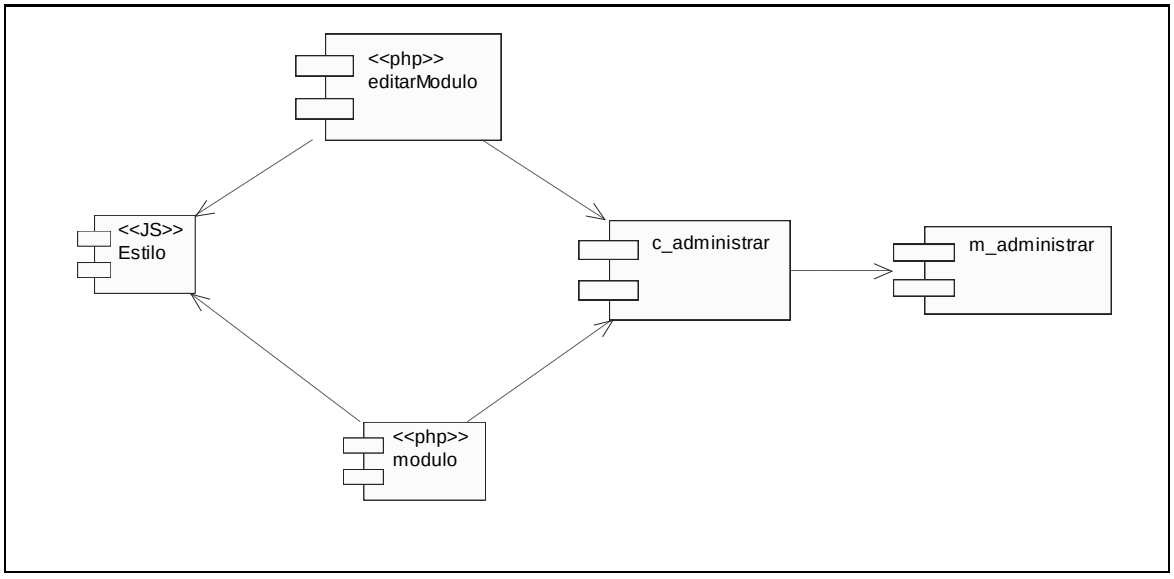


Figura 4.8 Diagrama de componentes del caso de uso "Gestionar módulo".

Diagrama de componentes del caso de uso "Actualizar brigada".

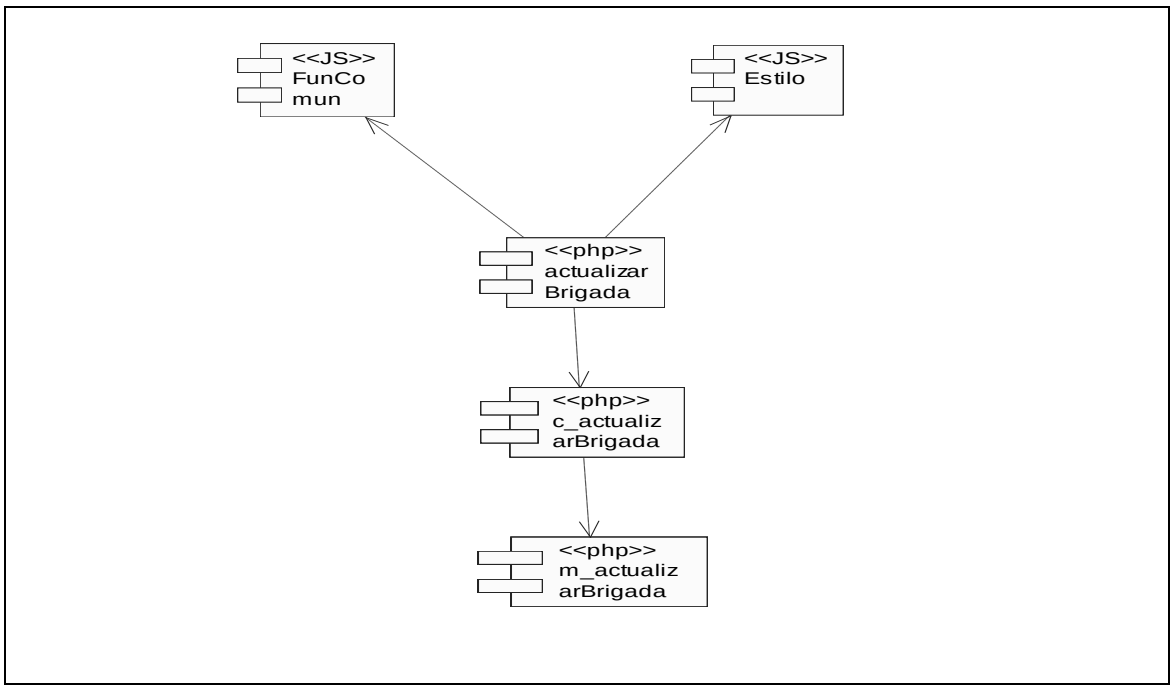


Figura 4.9 Diagrama de componentes del caso de uso "Actualizar brigada".

Diagrama de componentes del caso de uso "Gestionar Visita a Apartamento".

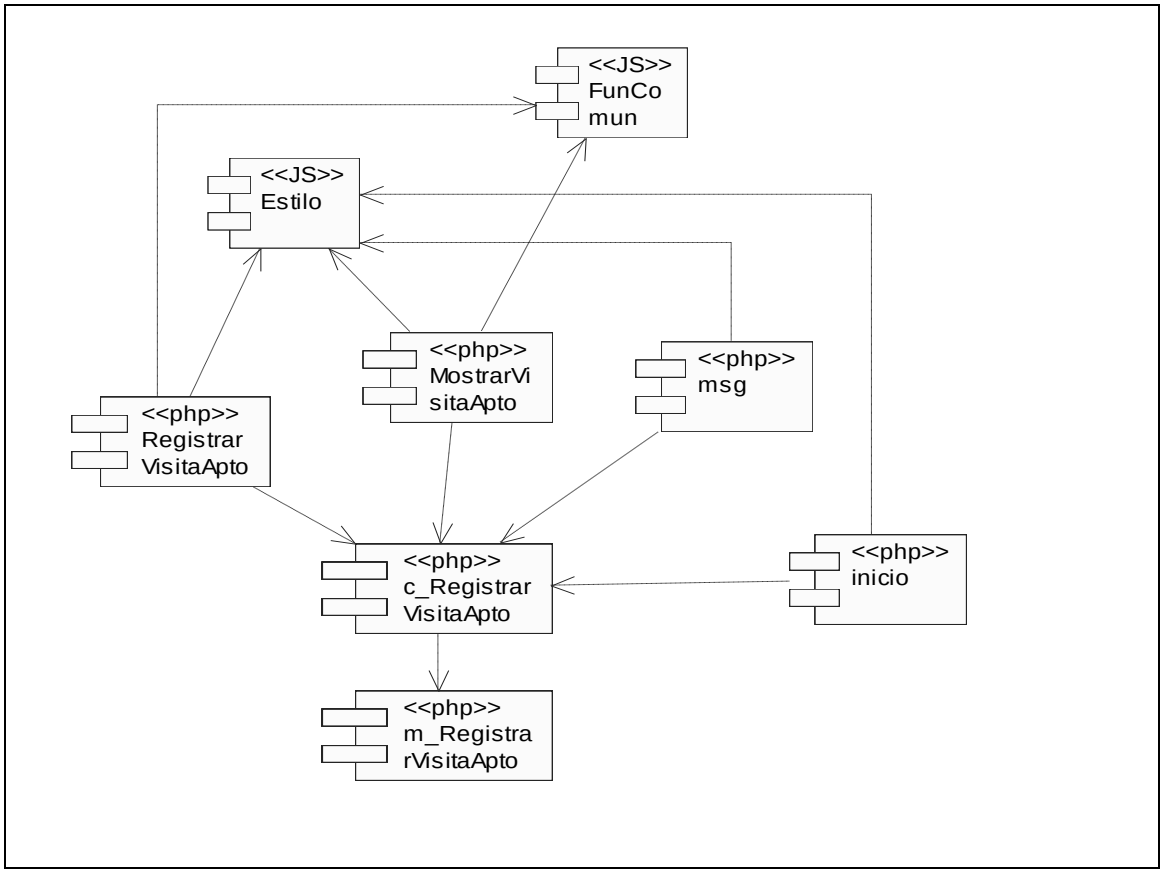


Figura 4.10 Diagrama de componentes del caso de uso "Gestionar Visita a Apartamento".

Diagrama de componentes del caso de uso "Gestionar Lugar de TSU".

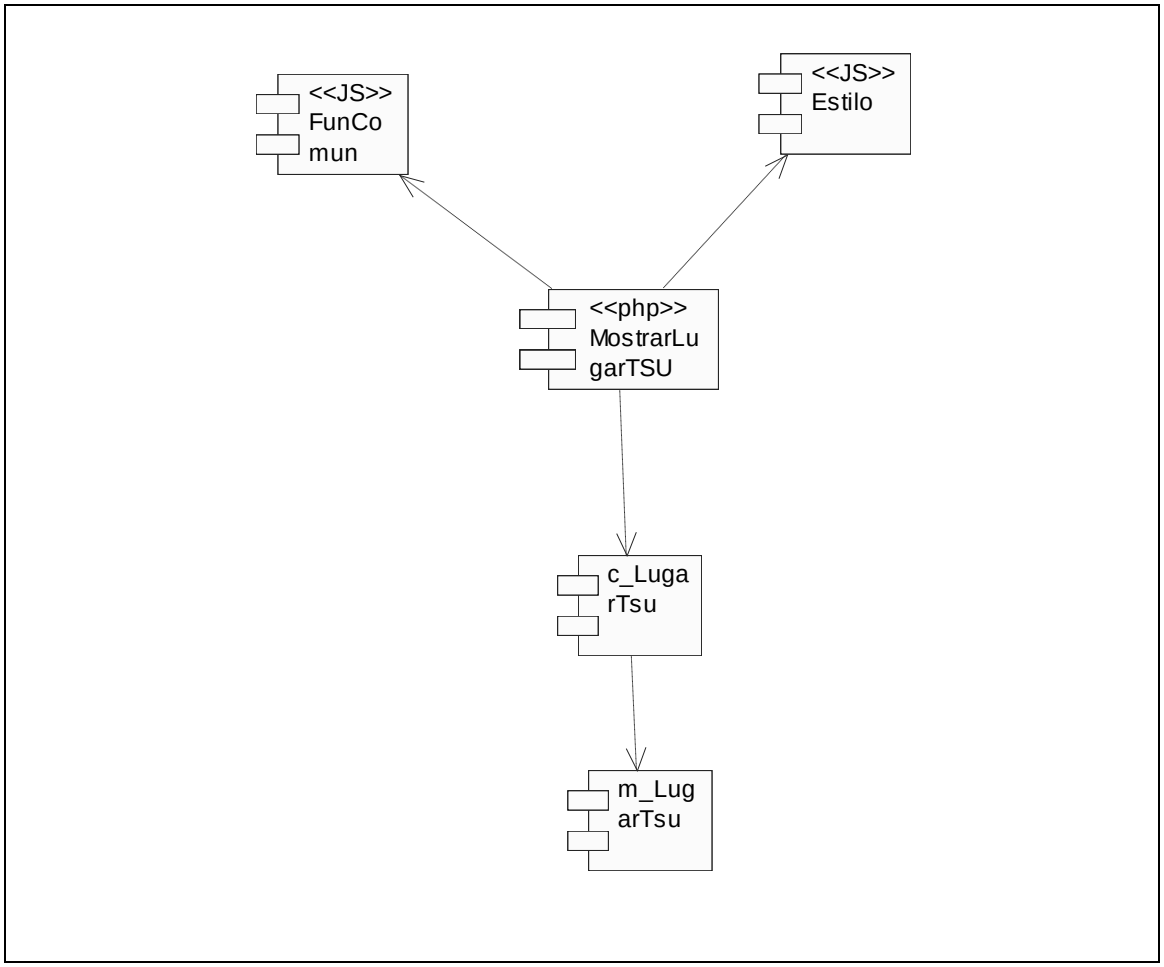


Figura 4.11 Diagrama de componentes del caso de uso "Gestionar Lugar de TSU".

Diagrama de componentes del caso de uso "Gestionar Posta de Guardia".

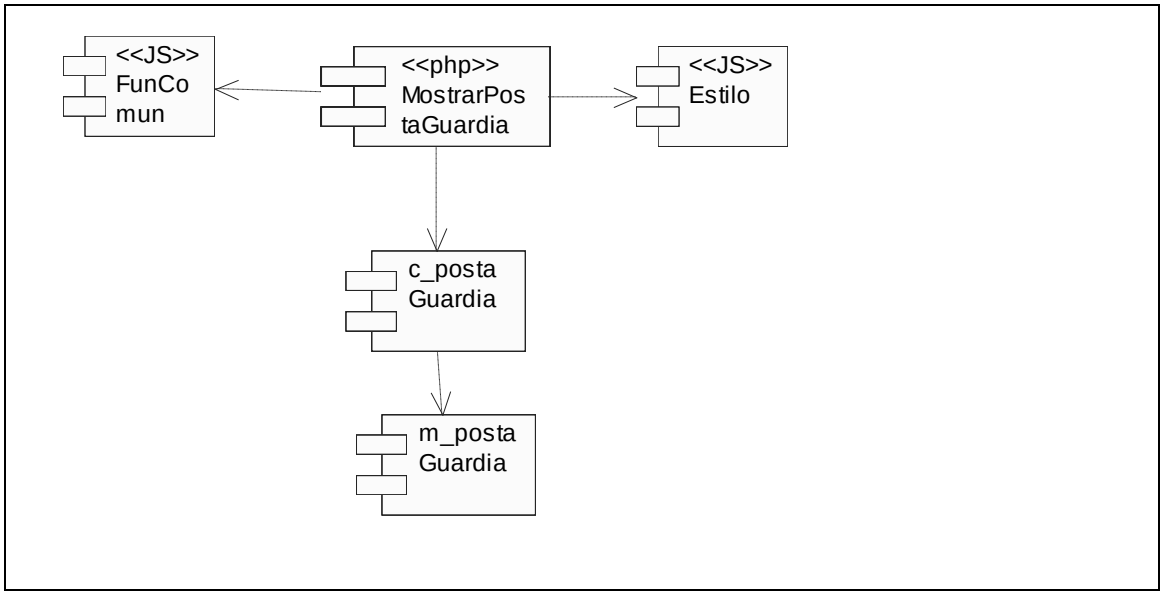


Figura 4.12 Diagrama de componentes del caso de uso "Gestionar Posta de Guardia".

4.5 Conclusiones del Capítulo.

En este capítulo se ha mostrado mediante el diagrama de despliegue los nodos necesarios para la utilización del sistema, garantizando que este funcione correctamente, lo que se traduce en la utilización de una PC servidora para emplazar la aplicación, una PC servidora de base de datos para la base de datos del sistema, las PC clientes en donde el usuario podrá interactuar con el sistema y una impresora para imprimir los reportes generados por el mismo. Además se mostró la relación existente entre todos los archivos que conforman el sistema, y su relación con la base de datos mediante el Diagrama de Componentes del Sistema.

CONCLUSIONES

La solución propuesta en este trabajo cumple con los objetivos trazados de forma satisfactoria, ya que la aplicación informática a la que se le realizó análisis, diseño e implementación está a la altura de las exigencias del cliente, en este caso el vicedecano de extensión universitaria de la facultad 8 de la Universidad de Ciencias Informáticas (UCI). Con esta nueva aplicación se contribuye a la informatización de los procesos que tienen lugar en la residencia estudiantil, permitiendo así un mayor control por parte de la dirección de la facultad sobre las actividades de la residencia como pueden ser guardias estudiantiles, trabajos socialmente útiles, cuartelerías y visitas realizadas a los apartamentos, garantizando de esta forma un fácil acceso a la información relacionada con dichos procesos. La aplicación está de acuerdo a los requerimientos, los cuales soportan al sistema y los casos de uso satisfacen las necesidades funcionales.

Durante la realización del sistema mediante el estudio y uso de las diferentes tecnologías aplicadas al mismo, se profundizaron los conocimientos en el ámbito profesional, lo cual contribuye en la preparación y formación como trabajadores de la informática para cumplir un determinado rol en la informatización de la sociedad cubana.

RECOMENDACIONES

Luego de haber cumplido con los objetivos propuestos y dado que el sistema se encuentra en su segunda versión se recomienda:

- Poner a prueba el sistema durante un período de tiempo con el objetivo de comprobar su funcionamiento.
- Perfeccionar en lo posible los criterios de búsqueda de la aplicación.
- Agregarle nuevas funcionalidades al sistema, con el objetivo de que el usuario pueda obtener otros tipos de reportes exportados para un formato digital.
- Proponer, luego de comprobar el buen funcionamiento del sistema, la utilización del mismo en las diferentes facultades de la UCI.
- Archivar el documento con el objetivo de que pueda ser consultado en un futuro para una modificación o actualización de la aplicación.

REFERENCIAS BIBLIOGRÁFICAS.

[1] Raúl Rodas Hinostroza, Características de PHP, http://www.linuxcentro.net/linux/staticpages/index.php?page=CaracteristicasPHP 06/12/2007.

[2] Anónimo, Introducción al lenguaje PERL, http://kataix.umag.cl/~mmarin/topinf/perl.html 10/01/2008

[3] Anónimo, ¿Qué es ASP.NET?, http://es.gotdotnet.com/quickstart/aspplus/doc/whatisaspx.aspx 11/01/2008

[4] Maryvonne Enjolras, Beneficios del uso de JAVA en las aplicaciones modernas de Bibliotecas, http://fesabid98.florida-uni.es/Comunicaciones/m_enjolras.htm 11/01/2008

[5] Anónimo, Herramientas Case, http://members.fortunecity.com/software1/herramie.htm 06/12/2007

[6] Robin Alberto Castro Gil, Estructura básica del proceso unificado de desarrollo de software, http://dspace.icesi.edu.co/dspace/bitstream/item/399/1/rcastro_estructura-bas-puds.pdf 13/01/2008

[7] Anónimo, Definición de metodología, http://definicion.de/metodologia/ 14/01/2008

[8] Araceli Torres Lecuanda**,** Metodologías modernas de desarrollo de Sistemas de Información, http://www.monografias.com/trabajos12/docmento/docmento.shtml#intro 14/01/2008

[9] Aurora Vizcaíno, Ismael Caballero, http://alarcos.inf-cr.uclm.es/per/fgarcia/isoftware/doc/LabTr1_RationalRose.pdf, 25/01/2008

[10] anónimo, http://www.dbrunas.com.ar/postgres/migrapg.pdf 25/01/2008

[11] anónimo, http://descargar.mp3.es/lv/group/view/kl61486/NuSphere_PhpED.htm , 04/02/2008.

[12] María Mercedes Marqués Andrés, Sistemas de gestión de bases de datos, http://www3.uji.es/~mmarques/f47/apun/node1.html 06/02/2008.

[13] Desconocido, Ventajas de PostgreSQL, http://soporte.tiendalinux.com/portal/Portfolio/postgresql_ventajas_html 04/01/2008.

[14] Anónimo, Conceptos básicos de Dreamweaver8 (I), http://www.aulaclic.es/dreamweaver8/t_1_1htm, 04/02/2008.

[15] Jennifer Taylor Título, Nuevas funciones y ventajas de Dreamweaver8, http://www.adobe.com/es/devnet/dreamweaver/articles/dw8_newfeatures.html, 04/01/08.

[16] Anónimo, Lenguajes de programación, http://www.frt.utn.edu.ar/sistemas/paradigmas/lenguajes.htm, 09/02/2008.

[17]Ing. Enrique Canseco, Metodología MSF, http://www.coworker.com.mx/nota.asp?id=28 ,10/02/2008

[18] Ing. Maria A. Mendoza Sánchez, Metodologías De Desarrollo De Software, http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html, 10/02/08.

[19] Daniel Pecos, PostgreSQL vs. MySQL, http://www.netpecos.org/docs/mysql_postgres/index.html 12/02/08.

[20] Desconocido, Introducción a CakePHP, http://cakephp-es.org/doku.php?id=introduccion_a_cakephp , 10/02/08.

[21] Desconocido, Framework PHP para desarrollo web http://codeigniter.uptodown.com/ , 12/02/08.

[22] Desconocido, Symfony en pocas palabras, http://www.librosweb.es/symfony/capitulo1/symfony_en_pocas_palabras.html , 12/02/08.

BIBLIOGRAFÌA

Anónimo. 2006. PostgresSQL 8.2 Documentation. [En línea] 2006. http://www.postgresql.org/docs/8.2/interactive/index.html..
C, larman. 2000. *UML y Patrones.* 2000.
Jacobson, Ivar, y otros. 2000. *El Proceso Unificado de Modelado de Desarrollo de Sofware.* 2000.
RUMBAUGH, y otros. 2000. *El Lenguaje Unificado de Modelado.* 2000.
UCI. Teleformación. [En línea] http://Teleformacion.uci.cu.

GLOSARIO DE TÉRMINOS

PHP: *Hypertext Preprocessor*. Es un ambiente script del lado del servidor que permite crear y ejecutar aplicaciones Web dinámicas e interactivas. Con PHP se pueden combinar páginas HTML y scripts. Con el objetivo de crear aplicaciones potentes.

HTML: Lenguaje usado para escribir documentos para servidores World Wide Web.

WEB (WWW): Red de documentos HTML intercomunicados y distribuidos entre servidores del mundo entero.

Script: es un programa usualmente simple, que generalmente se almacena en un archivo de texto plano. Los guiones son casi siempre interpretados, pero no todo programa interpretado es considerado un guión. El uso habitual de los guiones es realizar diversas tareas como combinar componentes, interactuar con el sistema operativo o con el usuario. Por este uso es frecuente que los shells sean a la vez intérpretes de este tipo de programas.

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.

SQL: *Structured Query Language*. Es un lenguaje declarativo de acceso a bases de datos que permite especificar diversos tipos de operaciones sobre las mismas. Aúna características del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar información de interés de una base de datos.

ASP: *Active Server Pages.* Es una tecnología del lado servidor de Microsoft para páginas Web generadas dinámicamente, que ha sido comercializada como un anexo a Internet Information Server (IIS).

XML: *Extensible Markup Language*. Es un lenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium. Orientado principalmente al almacenamiento, procesamiento y transmisión de mensajes.

UML: *Unified Modeling Language*. Es una notación estándar para modelar objetos del mundo real como primer paso en el desarrollo de programas orientados a objetos. Es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema de software.

RUP: *Rational Unified Process* (Proceso Unificado de desarrollo). Metodología para el desarrollo de Software.

Integer: tipo de dato que incluye los números naturales, sus opuestos y el cero.

MySQL: Es un sistema de gestión de bases de datos relacional que cuentan con todas las características de un motor de BD comercial: transacciones atómicas, triggers, replicación, llaves foráneas entre otras. Su ingeniosa arquitectura lo hace extremadamente rápido y fácil de personalizar.

PostgreSQL: es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS) libre.

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

Microsoft: Compañía que manufactura los sistemas de operación DOS y Windows.

HTTP: Es el conjunto de reglas para intercambiar archivos (texto, gráfica, imágenes, sonido, video y otros archivos multimedia) en la World Wide Web.

CASE: *Computer Aided Software Engineering.*

TCP/IP: Protocolo de Control de Transmisión (TCP) y Protocolo de Internet (IP).

Hardware: Componentes electrónicos, tarjetas, periféricos y equipo que conforman un sistema de computación; se distinguen de los programas (software) porque son tangibles.

TSU: Trabajo Socialmente Útil.

CUN: sigla que se utiliza para referirse al término Caso de Uso del Negocio.

CUS: sigla que se utiliza para referirse al término Caso de Uso del Sistema.

USB: Interfaz estándar que facilita la conexión de periféricos a un ordenador.

CSS: Hojas de Estilo en Cascada (Cascading Style Sheets): Es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura.

Licencia GNU/GPL: En español Licencia Pública General, es una licencia creada por la *Free Software Fundation* y orientada principalmente a los términos de distribución, modificación y uso de *software*. Su propósito es declarar que el software cubierto por esta licencia es *software* libre.