Instituto Superior Politécnico “José Antonio Echeverría”

Facultad de Ingeniería Industrial
Centro de Estudios de Ingeniería y Sistemas

Universidad de las Ciencias Informáticas
Dirección de Informatización

Módulo Alojamiento del Sistema Automatizado para la Gestión de Información de la Misión Milagro

Trabajo de Diploma para optar por el título de Ingeniero Informático

AUTOR

Cesar González Hernández

TUTOR

Ing. Anabel Parra Vázquez

Ciudad de La Habana, Cuba
Junio, 2006

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas (UCI) para que haga el uso que estimen pertinente con este trabajo.

Para que así conste firmo la presente a los __ días del mes de junio del 2006.

_____	_____
Firma del Autor	Firma del Tutor
Cesar González Hernández	Ing. Anabel Parra Vásquez

OPINIÓN DEL USUARIO DEL TRABAJO DE DIPLOMA

El Trabajo de Diploma, titulado “Módulo Alojamiento del Sistema Automatizado para la Gestión de Información de la Misión Milagro”, fue realizado en la Universidad de las Ciencias Informáticas (UCI) de la provincia de Ciudad Habana. Esta entidad considera que, en correspondencia con los objetivos trazados, el trabajo realizado le satisface

- □ Totalmente
- □ Parcialmente en un ____ %

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

Y para que así conste, se firma la presente a los ___ días del mes de junio del 2006

_____ _____
Representante de la entidad Cargo

_____ _____
Firma Cuño

OPINIÓN DEL TUTOR DEL TRABAJO DE DIPLOMA

Título: Módulo Alojamiento del Sistema Automatizado para la Gestión de Información de la Misión Milagro.

Autor: Cesar González Hernández.

El tutor del presente Trabajo de Diploma considera que durante su ejecución el estudiante mostró las cualidades que a continuación se detallan.

Por todo lo anteriormente expresado considero que el estudiante está apto para ejercer como Ingeniero Informático; y propongo que se le otorgue al Trabajo de Diploma la calificación de ____ .

_____ ____ de junio del 2006

Firma

"...Se es hombre para serlo. Hombre es algo más que ser torpemente vivo: es entender una misión: ennoblecerla, cumplirla..."

José Martí

Agradecimientos

*A la **Revolución** por oportunidad que me ha dado y la obra tan hermosa que realiza.*

*A mis **padres** por todo el amor y la dedicación que he recibido de ellos y por la fuerza que despiertan en mí.*

*A mi **hermano** por quererme y soportarme como soy.*

*A mis **abuelos** por su cariño y por las vivencias transmitidas.*

*A mi gran **familia** por toda la preocupación el apoyo que me han brindado.*

*A mis **compañeros de Holguín** por las experiencias compartidas y estar siempre ahí en los momentos difíciles.*

*A mis **compañeros de quinto año** por la unidad que los caracterizó y por los tantos momentos que compartimos.*

*A **Elsia**, por todo el amor y la comprensión que me ha dedicado.*

*A mis **profesores** por todos los conocimientos transmitidos y los valores que formaron en mí.*

*A mis **amigos** por estar siempre ahí brindarme su apoyo incondicional.*

*A **Jorge,** mi otro hermano, por toda la ayuda y el apoyo que me ha dado.*

*A mi **tutora** por su paciencia y los conocimientos que nos ha transmitido.*

*A **Yunier** por el apoyo que le ha brindado al proyecto.*

*A mis **compañeros de tesis** por la cooperación que los caracterizó.*

*A los **muchachos del proyecto** por la ayuda que me han brindado.*

A los que de una forma u otra forma han contribuido con la realización de este trabajo.

A todos, gracias.

Dedicatoria

A mis padres que fueron la luz primera, mis primeros maestros. A ellos que son la llama eternizada que vive en mis venas, que vive en mi alma.

RESUMEN

El presente trabajo se realizó en la Universidad de las Ciencias Informáticas, centro de apoyo a la Misión Milagro, donde se brindan diferentes servicios y recursos durante el desarrollo de esta tarea.

El motivo de este trabajo es brindar una solución automatizada, flexible y única a los diferentes problemas presentados en los procesos de alojamiento, durante el desarrollo de la pasada misión. Para ello se estructura en 5 capítulos donde se recoge la fundamentación teórica del trabajo, el modelo del negocio, los requisitos del sistema, la descripción de la solución propuesta y el estudio de factibilidad.

Se propone como solución una aplicación Web, basada en tecnología PHP5 y con gestor de base de datos PostgreSQL, la cual contribuye a la reducción del tiempo en las búsquedas de información, permite mejorar las condiciones de trabajo del personal de apoyo a la misión, evitándoles el agotamiento y demora que produce el procesamiento manual de la información al contribuir positivamente en el almacenamiento y control de ésta.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Con la creación colectiva de la Constitución de la República Bolivariana de Venezuela, refrendada por voluntad popular en diciembre del 2000, se impulsaron grandes cambios y soluciones en distintos sectores de la sociedad, el sector de la salud no estuvo ajeno a estos cambios.

Casos de parasitismo, diarrea, problemas respiratorios, hipertensión y diabetes, colmaban las salas de emergencia de los hospitales, y siendo estas patologías controlables o curables, estaban destinadas a no ser atendidas. Por esta razón fueron creados los consultorios populares en torno a una red de atención primaria llamada Misión Barrio Adentro. Este programa se inicia el 16 de abril del 2003, cuando arribó a Caracas una brigada de 58 médicos cubanos, con la intención de proveer atención médica a los sectores populares del municipio Libertador de la ciudad capital. [1]

La Misión Milagro surge en el marco de los acuerdos entre Caracas y La Habana en el 2004 y es parte de la llamada misión Barrio Adentro, por la cual varios miles de médicos cubanos trabajan en las zonas más humildes de Venezuela, en su mayoría barriadas de precarias viviendas en las que viven gentes que carecían de los servicios públicos más elementales.

Allí pudieron detectar enfermedades oftalmológicas de sencilla curación que mantenían en muchos casos en la ceguera a miles de venezolanos. Un puente aéreo entre Cuba y Venezuela para intervenciones quirúrgicas podía resolver muchas de esas enfermedades gracias al importante desarrollo de la sanidad cubana. El acuerdo contemplaba la gratuidad de todo el proceso para los enfermos. [2]

Desde los lugares más remotos de la geografía venezolana, más de diez mil personas de todas las edades, con sus respectivos acompañantes, han viajado a Cuba para ser intervenidos quirúrgicamente en casos de cataratas, desprendimiento de retina, retinitis pigmentaria, carnosidad, párpado caído, afecciones del iris, así como también casos de dermatología, traumatología y cáncer. Se estudia la apertura de nuevas especialidades

a ser tratadas por la Misión Milagro, como operaciones de corazón, cuello uterino y columna, aunque ésta es una misión especializada en patologías específicas de la vista. Asimismo, los presidentes de Venezuela y Cuba han extendido los horizontes de la Misión, planteando una cantidad meta de miles de operaciones para cada año. [1]

El convenio firmado entre los dos países amigos establece la meta de realizar este tipo de cirugía a 600 mil personas de toda América Latina por año, durante un período que se extenderá hasta el año 2016. De esa forma más de 6 millones de latinoamericanos estarán siendo beneficiados por esta misión humanitaria que los gobiernos de Cuba y Venezuela están desarrollando. [3]

Hasta febrero último más de 187 mil 275 personas, de los cuales 178 mil 100 son de Venezuela, 11 mil 811 del Caribe y 7 mil 364 de Latinoamérica, habían sido operadas en la Mayor de las Antillas a través de esta forma política del ALBA (Alternativa Bolivariana para las Américas), propiciadora del intercambio entre países. [4]

La Universidad de las Ciencias Informáticas (UCI) se ha convertido en unos de los centros de apoyo a esta misión brindando diferentes servicios (hospedaje, alimentación, etc.) y todos los recursos necesarios (ropa, aseo, dinero, tarjetas telefónicas, etc.) para su desarrollo. En este centro estudiantes y profesores cambian sus profesiones convirtiéndose en verdaderos trabajadores sociales.

El período vacacional 2005 constituyó la segunda ocasión en que la UCI se transformó en hospital para esta misión, el mayor de todo el país, para la cual se habilitaron cerca de 2000 habitaciones para hospitalizar a enfermos.

Los estudiantes del centro donan su residencia y parte de sus vacaciones para lograr el éxito de esta actividad y de esta forma contribuir con nuestra sociedad en una tarea tan importante y humanitaria.

Los procesos relacionados con el aseguramiento de los recursos y servicios que se les brindan a las personas hospitalizadas así como el aseguramiento y control del personal que trabaja en la misión constituyen la columna vertebral.

Dentro de los servicios prestados se encuentra el de alojamiento el cual resulta la base de la estancia de los hospitalizados así como del personal que trabaja en la misión. Para el control de la información relacionada con dicho proceso el personal se apoya en sistemas como el Care2x y el sistema de apoyo integral a la Misión Milagro (SAIMM).

Actualmente existe dificultad de integración de los sistemas que se utilizan en el proceso. Se detecta con frecuencia ubicaciones de pacientes y acompañantes duplicadas. Además, no existen mecanismos para controlar la ubicación del personal de la misión por lo que la localización se hace muy engorrosa. Se asignan ubicaciones en apartamentos que no están de servicio. Existe demora en la toma de decisiones a la hora de alojar a pacientes con discapacidad física.

El trabajo surge con la **necesidad** de hacer un software que garantice la rápida y segura gestión de la información, para controlar el proceso de alojamiento durante el desarrollo de la Misión Milagro en la Universidad de las Ciencias Informáticas. El **problema** se puede formular entonces de la siguiente manera: ¿Cómo gestionar y controlar la ubicación de todas las personas asociadas a la misión garantizando la centralización y veracidad de la información así como la agilización en los procesos de toma de decisiones?

Por tanto el **objeto de estudio** de este trabajo son: los diferentes procesos, servicios y recursos que se brindan en la UCI durante la Misión Milagro para garantizar la estancia de todo el personal que participa en la misión, los pacientes y los acompañantes que reciben atención médica en dicho centro, aparejado al estudio de plataformas que permitan su implementación.

De ello se deriva que el **campo de acción** comprende todo el proceso de gestión del alojamiento durante la Misión Milagro en la Universidad de las Ciencias Informáticas.

Como **hipótesis** se parte de la idea de que la implementación y puesta en marcha de un sistema informático único que de soporte al proceso de alojamiento, garantizará la gestión y control de la ubicación de forma centralizada, disminuirá el volumen de información en papel, simplificará el trabajo del personal que apoya la misión, garantizará de una forma rápida y factible el registro, gestión y protección de las informaciones tratadas durante el desarrollo de la Misión Milagro, todo esto proporcionará mayor calidad en los servicios brindados en la Universidad de las Ciencias Informáticas.

Aportes prácticos

- Centralización de toda la información referente al proceso.
- Rapidez en la comunicación y las búsquedas de información por parte del personal de apoyo disminuyendo su carga de trabajo.
- Disminución del consumo de material de oficina.

Se ha propuesto como **objetivo general** desarrollar una solución informática, robusta, flexible y única de software que dé soporte al proceso de gestión del alojamiento durante la misión

Como **objetivos específicos** se platean los siguientes:

- Obtener el diseño de una base de datos capaz de almacenar de manera organizada la información que se manipula.
- Desarrollar el análisis y diseño del sistema.
- Implementar el sistema con las características definidas en los procesos de análisis y diseño.

Para cumplir los objetivos se desarrollaron las siguientes **tareas**:

- Analizar el funcionamiento del alojamiento en el período de la misión.
- Entrevistar a trabajadores y estudiantes del centro que hayan trabajado en misiones anteriores para conocer el sistema de trabajo e identificar las necesidades del proceso.

- Investigar como se controla dicho proceso en otras entidades, así como las herramientas que se utilizan.
- Diseñar de una base de datos que soporte la mayoría de las funcionalidades del sistema.
- Chequear los datos de la residencia e introducirlos a la base de datos.
- Realizar el análisis y diseño de la aplicación.
- Implementar del Software haciendo uso de herramientas de código abierto como PostgreSQL 8.1.X como gestor de base de datos y PHP5 como lenguaje de programación.
- Documentar la información referente al análisis, diseño e implementación del sistema, así como todo lo referente al flujo de trabajo.
- Elaborar un documento que sirva de ayuda a los usuarios del sistema.

La presente propuesta contribuye a mejorar y a optimizar las tareas desarrolladas asociadas a la misión, de esta forma se garantiza la estancia de los pacientes y sus acompañantes hospedados en la Universidad de las Ciencias Informáticas. Se persigue obtener un producto de software que responda a la mayor parte de necesidades de la Misión Milagro.

El presente trabajo, estructurado en 5 capítulos, resume la siguiente información:

Capítulo 1. Fundamentación Teórica: descripción del objeto de estudio, sistemas existentes vinculados al campo de acción, tendencias y tecnologías actuales seleccionadas a emplear en el desarrollo de la propuesta y por qué su utilización.

Capítulo 2. Modelo del Negocio: descripción de los procesos, actores, trabajadores y casos de uso del negocio; y diagramas de clases del modelo de objetos del negocio.

Capítulo 3. Requisitos: definición de los requisitos funcionales y no funcionales; actores y casos de uso del sistema.

Capítulo 4. Descripción de la solución propuesta: descripción del diseño a través del diagrama de clases Web, que describen la relación entre las páginas.

Se definen, además, los principios de diseño seguidos en la aplicación y el modelo de implementación mediante el diagrama de despliegue.

Capítulo 5. Estudio de factibilidad: estudio de factibilidad económica realizado para este proyecto, en el que se determina si es factible o no el desarrollo del software propuesto, analizando los diferentes criterios que influyen en el cálculo del esfuerzo, tiempo de desarrollo y costo del proyecto.

CAPÍTULO 1
Fundamentación Teórica

1.1 Introducción.

Este capítulo contiene los principales problemas que fundamentan la propuesta de solución, y los objetivos generales y específicos que se persiguen. Además de brindar un enfoque general de sistemas automatizados existentes vinculados al campo de acción y el análisis comparativo de las soluciones existentes con la propuesta dada en este trabajo. Se describen además las tecnologías actuales de desarrollo utilizadas para el análisis, diseño e implementación del sistema sobre las cuales se apoya la propuesta.

1.2 Objeto de estudio.

El objeto de estudio de este trabajo son: los diferentes procesos que se desarrollan, servicios y recursos que se brindan en la UCI durante la Misión Milagro para garantizar la estancia de todo el personal que participa en la misión, los pacientes y los acompañantes que reciben atención médica en dicho centro, aparejado al estudio de plataformas que permitan su implementación.

El objetivo principal de la UCI en esta importante tarea es precisamente garantizar todos los recursos necesarios a todas las personas que participan en la misión, ya sean pacientes, acompañantes o personal de apoyo. Convencida del papel que le ha tocado jugar, acomete diversas tareas para así lograr con mejores resultados sus objetivos. Por ello debe realizar una serie de procesos importantes y que son de obligatorio cumplimiento por todas las personas involucradas en esta actividad. Para asegurar todo lo anteriormente dicho la UCI en tiempo de misión se estructura de forma diferente: la residencia universitaria se descompone en clínicas, se crea un puesto de mando general al que se subordinan otros puestos de mando y otras entidades. A continuación se muestra la figura referente a la estructura de la UCI en tiempo de misión.

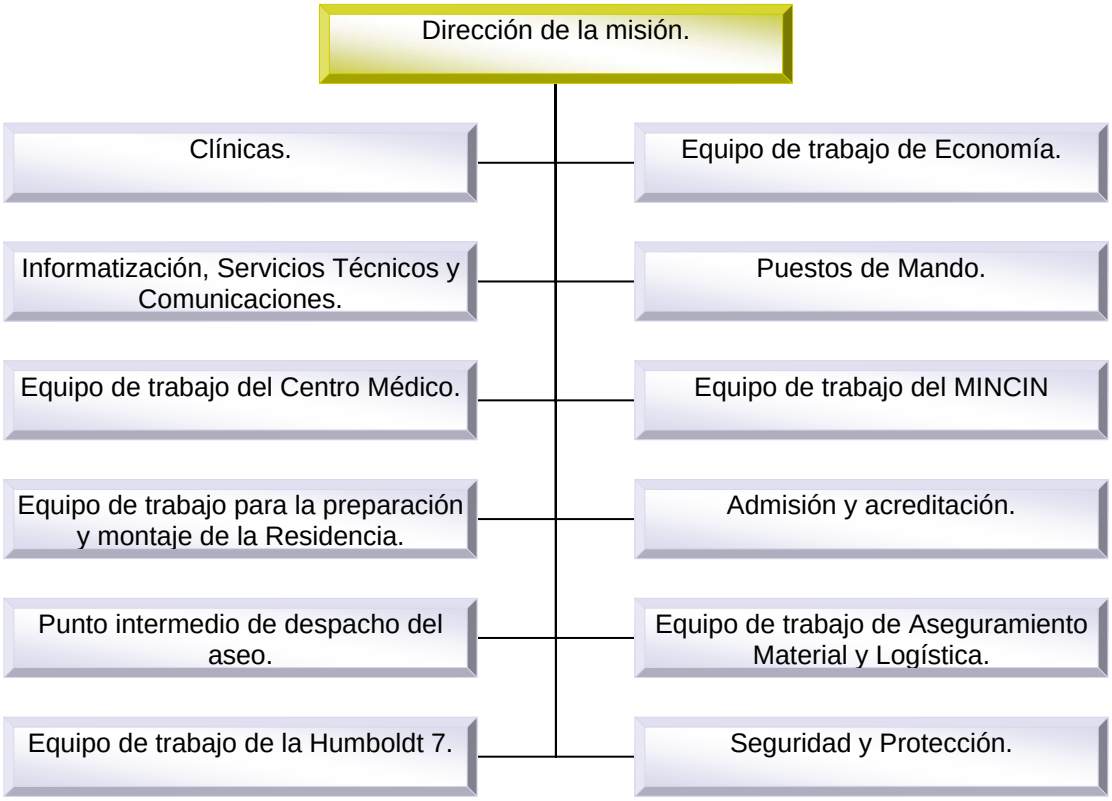


Figura 1.1 Estructura de la UCI en tiempo de misión.

1.2.1 Flujo actual de los procesos.

El análisis del flujo de procesos permite reconocer como funciona realmente el negocio para producir uno o varios resultados. El resultado puede ser un producto, un servicio, una información o combinaciones de ellos. Analizar el flujo de los procesos permite revelar problemas potenciales tales como: los cuellos de botella, los pasos innecesarios, la circulación doble de la información, la duplicación del trabajo, solo por citar algunos.

Tras el arribo al centro del personal vinculado a la misión con necesidad de alojamiento, se inicia inmediatamente en los puntos de admisión el proceso de ubicación cuyo objetivo final es asignar una capacidad a dicha persona. Una vez concluido el proceso de ubicación, la persona se traslada a la clínica o a la residencia, teniendo la posibilidad de reubicarse en caso de que presente algún tipo de inconformidad con la ubicación asignada, en este caso se inicia un proceso de reubicación. El último proceso en la

gestión del alojamiento es la liberación de la capacidad asignada, el cual se lleva a cabo una vez que la persona ha abandonado el centro.

A continuación se muestra una figura que representa el flujo de ejecución de los procesos, vinculado a los momentos por los que transita el beneficiado.

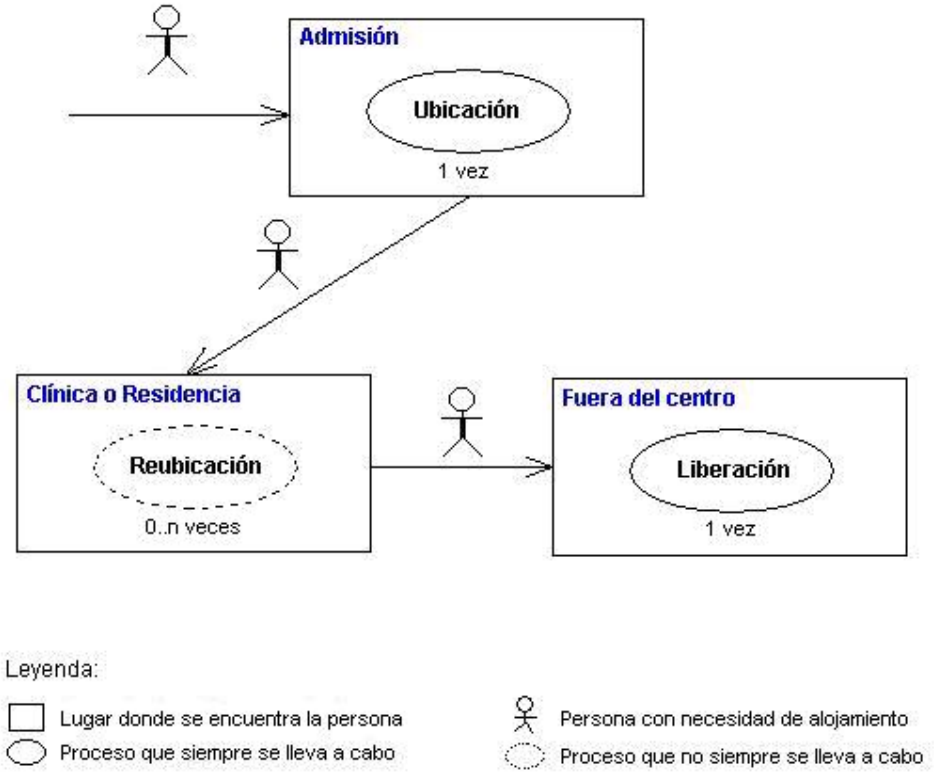


Figura 2.1 Flujo de los procesos.

1.2.2 Análisis crítico de la ejecución de los procesos.

Aunque el flujo actual de los procesos ha permitido satisfacer las necesidades básicas de alojamiento durante el período de misión cabe destacar aspectos deficientes que se han detectado y que han generado la situación problémica que se trata de resolver con el presente trabajo.

Una vez que el puesto de mando de vuelo informa de la asignación de un vuelo, el puesto de mando de la UCI avisa a las clínicas que tienen posibilidades de ubicación atendiendo a la cantidad de pasajeros que trae dicho vuelo. Cuando los hospitalizados arriban al centro, son conducidos a los puntos de acreditación donde esperan los

responsables de alojamiento de las clínicas que son los encargados de asignarles una ubicación. Luego son trasladados a los mesas de acreditación donde se les recogen los datos personales y se actualiza el listado de las capacidades, finalmente el hospitalizado es conducido a la ubicación que le fue asignada.

Actualmente existen deficiencias cuando el responsable de alojamiento trata de asignar una capacidad a un hospitalizado debido a que esta actividad se lleva a cabo de forma manual lo que genera descontrol en cuanto a las ubicaciones que han sido asignadas trayendo consigo duplicidad en la entrega de las mismas y errores en cuanto al tipo de ubicación que se asigna atendiendo al sexo del hospitalizado, sus condiciones físico mentales y en ocasiones se asignan apartamentos que están fuera de servicio por presentar algún tipo de problema. Realizar la actividad de ubicar antes de acreditar implica inevitablemente llevar el control del proceso mediante dos copias del listado de las capacidades, donde, para mantener la veracidad de la información que se manipula en el proceso se debe garantizar la igualdad de los datos que contienen los listados, tanto la copia dura que se lleva en la ubicación como el contenido del Care2x en su base de datos. Este proceso se lleva a cabo de forma manual para el caso de los trabajadores de servicio que se alojan en el centro en el período de misión, trayendo consigo la existencia de un bajo control de las capacidades asignadas.

El proceso de reubicación abarca un conjunto de actividades que se activan una vez que el hospitalizado muestra al médico algún tipo de inconformidad respecto a la ubicación que le fue asignada o contrae una enfermedad contagiosa, motivo por el cual debe ser aislado del resto de los hospitalizados. El médico es la persona que atiende la petición de cambio o detecta la enfermedad y valora la situación. Si el médico considera que el cambio es necesario solicita al jefe de clínica el traslado de ubicación. El jefe de clínica a través del responsable de alojamiento consulta el estado de las disponibilidades y si existen condiciones en la clínica para efectuar la reubicación se procede a ello. En caso de que no estén creadas las condiciones informa al paciente la imposibilidad del cambio. En el proceso de reubicación resultan muy lentas las búsquedas, lo cual genera un cuello de botella si se desea reubicar un gran número de hospitalizados, en tanto este proceso se realiza de forma manual para el caso de los

trabajadores de servicio, provocando pérdidas de información y manejo de información desactualizada.

Una vez que los hospitalizados parten en un vuelo hacia su país de origen, el puesto de mando de vuelo informa de dicha salida al puesto de mando general quien a su vez informa al médico encargado de dar las altas médicas. El médico procede a conceder el alta médica al paciente y actualiza el listado de las capacidades. Si el paciente resulta ser trasladado a otro hospital el médico es el responsable de indicar el traslado y actualiza el listado de capacidades. De igual manera las búsquedas se tornan lentas a la hora de liberar las capacidades una vez que se ha dado a conocer la salida de un vuelo. Este proceso debe llevarse a cabo con cada uno de los hospitalizados que parten en el vuelo, generándose un gran cuello de botella, mientras que para los trabajadores de servicio el proceso se realiza completamente manual, trayendo consigo los mismos problemas detectados en el proceso de reubicación.

1.3 Procesos objeto de automatización.

Con el sistema se pretende automatizar los siguientes procesos:

- Ubicación
  - Todas las personas que llegan a la UCI con necesidad de alojamiento (pacientes, acompañantes, personal que va a trabajar en la misión) durante el desarrollo de la Misión Milagro deben ser alojados, este proceso no se llevará a cabo solo para el personal externo. En el caso de los hospitalizados debe tenerse en cuenta el estado físico mental del mismo.
- Reubicación
  - Todas las personas que se hayan alojado pueden pedir una reubicación, que es valorada y en caso de que sea aprobada se a cabo.
- Liberación de capacidades
  - Cuando una persona (paciente, acompañante, personal que trabaja en la misión) abandona el centro debe liberarse la capacidad que ocupaba para luego asignarla a otras personas.

1.4 Sistemas automatizados existentes vinculados al campo de acción.

Actualmente existen algunos sistemas vinculados al campo de acción, cada uno de ellos trata el problema del alojamiento siguiendo una estructuración específica del mismo, por ejemplo, la estructura sala–cama, común en muchos centros hospitalarios. Esta limitante dificulta su aplicación en otros centros como es el caso de la UCI, con una estructuración del alojamiento muy particular.

1. El sistema Misión Milagro CUJAE es una aplicación basada Access que surge por la necesidad de automatizar la gestión logística y médica en los diferentes centros que pertenecen a la misión. Se desarrolla con el objetivo de controlar el estado médico y los datos personales de los pacientes así como los artículos entregados a estos. Este sistema es implementado por el Departamento de Informática del centro hospitalario CUJAE, el cual permite la actualización de las informaciones y la obtención de las estadísticas referente a todo lo relacionado con los pacientes. En el mismo, el proceso de alojamiento se trata siguiendo las particularidades estructurales de dicha institución haciéndolo poco práctico para aplicarlo en un centro con otro tipo de estructura. Por otra parte no está desarrollado sobre herramientas de software libre y su funcionamiento está muy sujeto al tipo de plataforma.

2. El Care2x es un sistema basado íntegramente en software libre, multiplataforma, nace en el año 2002 y rápidamente se posiciona como un sistema modular confiable para propósitos asistenciales y educativos. Integra datos, funciones y flujo de tareas en un entorno de cuidados de la salud. Al momento, está compuesto de cuatro componentes principales: Sistema de Información Hospitalaria/Servicios de la Salud, Administración del Ejercicio Médico, Servidor Central de Datos, Protocolo de Intercambio de Datos de la Salud. Cada uno de estos componentes también puede funcionar de manera individual. Al igual que el sistema anterior trata el problema del alojamiento para la estructura sala-cama lo cual dificulta su aplicación en otro tipo de estructura. Pese a que está desarrollado sobre herramientas de software libre y presenta un alto nivel de portabilidad, muestra demora en los tiempos de respuesta cuando la concurrencia y el número de datos son elevados.

3. El sistema SAIMM (Sistema de Apoyo Integral a la Misión Milagro) surge en el centro hospitalario UCI y recoge diferentes procesos, no solo abarca parte de la logística, también gestiona tareas adicionales como las actividades recreativas, dietas, pasaportes y otros. Su objetivo es complementar las funcionalidades del sistema Care2x tras su aplicación en la UCI. En el tema del alojamiento no automatiza ninguno de los procesos básicos, sino que se utiliza fundamentalmente para la generación de reportes, el mismo gestiona la información del alojamiento mediante consultas que realiza directamente sobre la base de datos del Care2x.

El sistema propuesto resuelve el problema estructural en el tema del alojamiento, el mismo hace uso de gestores de bases de datos superiores con el objetivo de disminuir los tiempos de respuesta cuando son altos los volúmenes de datos así como la concurrencia a los mismos. Es importante destacar el uso de herramientas de Software Libre, pues confirma las ventajas en comparación con otros sistemas, debido a que esta práctica es gratuita y los requerimientos de hardware son relativamente bajos. La nueva propuesta es un sistema único en el cual se van a integrar todos los procesos y actividades que se desarrollan durante la misión en el Hospital UCI.

1.5 Tendencias y tecnologías actuales.

Teniendo en cuenta las necesidades vistas y las características del entorno donde se aplicará la solución propuesta, se realizó un estudio de las tendencias y tecnologías actuales posibles a emplear, descritas a continuación.

1.5.1 Las aplicaciones Web.

Las aplicaciones Web son una especialización y concreción de las aplicaciones cliente-servidor, o sea, su arquitectura general es la de un sistema cliente/servidor, donde tanto el cliente (el navegador) como el servidor (el servidor Web), y el protocolo mediante el que se comunican (el HTTP: HyperText Transfer Protocol) son estándar, y no han de ser creados por el desarrollador. [5]

La parte del cliente de las aplicaciones Web está formada por el código HTML (HyperText Markup Language) que forma la página Web, con opción a código

ejecutable mediante los lenguajes script de los navegadores (JavaScript, VBScript, PerlScript) o mediante pequeños programas (applets) en Java. La parte de servidor está formada por un programa o script que es ejecutado por el servidor Web, y cuya salida se envía al navegador del cliente. [5]

La creciente popularidad de las aplicaciones Web se debe a sus múltiples ventajas, entre las cuales podemos citar: [6]

- Multiplataforma: Con un solo programa, un único ejecutable, nuestras aplicaciones pueden ser utilizada a través de múltiples plataformas, tanto de hardware como de software.
- Actualización instantánea: Debido que todos los usuarios de la aplicación hacen uso de un sólo programa que radica en el servidor, los usuarios siempre utilizarán la versión más actualizada del sistema.
- Suave curva de aprendizaje: Los usuarios, como utilizan la aplicación a través de un navegador, hacen uso del sistema tal como si estuvieran navegando por Internet, por lo cual su acceso es más intuitivo.
- Fácil de integrar con otros sistemas: Debido a que se basa en protocolos estándares, la información manejada por el sistema puede ser accedida con mayor facilidad por otros sistemas.
- Acceso móvil: El usuario puede acceder a la aplicación con la única restricción de que cuente con un acceso a la red privada de la organización o a Internet, dependiendo de las políticas de dicha organización; puede hacerlo desde una computadora de escritorio, una laptop o desde una agenda electrónica; desde su oficina, hogar u otra parte del mundo.

El desarrollo de aplicaciones Web está siendo utilizado en muchas organizaciones, ésta situación va ir creciendo indefinidamente. Es por ello que día a día se requieran más programadores capacitados para desarrollos basados en el World Wide Web (WWW).

No obstante a la serie de ventajas que presenta tiene además algunas desventajas, las cuales son:

- Acceso limitado, la necesidad de conexión permanente y rápida a Internet hacen que el acceso a estas aplicaciones no esté al alcance de todos.
- La interactividad no se produce en tiempo real, en las aplicaciones web cada acción del usuario conlleva un tiempo de espera algunas veces excesivo hasta que se obtiene la reacción del sistema.
- Elementos de interacción muy limitados. En comparación con el software de escritorio, las posibilidades de interacción con el usuario que ofrecen las aplicaciones web (mediante formularios principalmente) son muy escasas.
- Diferencias de presentación entre plataformas y navegadores. La falta de estándares ampliamente soportados dificulta el desarrollo de las aplicaciones.

1.5.2 Modelo Cliente Servidor.

Se dice que la arquitectura Cliente/Servidor es la integración distribuida de un sistema en red, con los recursos, medios y aplicaciones que, definidos modularmente en los servidores, administran, ejecutan y atienden las solicitudes de los clientes; todos interrelacionados física y lógicamente, compartiendo datos, procesos e información. Se establece así un enlace de comunicación transparente entre los elementos que conforman la estructura. Entre las principales características de la arquitectura Cliente/Servidor, se pueden destacar las siguientes: [7]

- El servidor presenta a todos sus clientes una interfaz única y bien definida.
- El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.
- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente.

Ventajas de la arquitectura cliente-servidor: [6]

- El servidor no necesita potencia de procesamiento, parte del proceso se reparte con los clientes.
- Se reduce el tráfico de red considerablemente. Idealmente, el cliente se conecta al servidor cuando es estrictamente necesario, obtiene los datos que necesita y cierra la conexión dejando la red libre.

Las arquitecturas de dos capas contienen tres componentes distribuidos en dos capas: cliente (solicitante de servicios) y servidor (proveedor de servicios). *Ver Anexo 1.*

Los tres componentes son: [6]

1. Interfaz de usuario al sistema. Tales como una sesión, entradas de texto, desplegado de menús, etc.
2. Administración de procesamiento. Tales como la ejecución de procesos, el monitoreado de los mismos y servicios de procesamiento de recursos.
3. Administración de bases de datos. Tales como los servicios de acceso a datos y archivos.

La arquitectura de software de tres capas emergió en la década de los noventas para solventar las limitaciones de la arquitectura de dos capas. La tercera capa (capa de servicios) se localiza entre la interfaz de usuarios (cliente) y el administrador de datos (servidor). Esta capa intermedia provee de servicios para la administración de procesos (tal como desarrollo, monitoreo y alimentación de procesos) que son compartidos por múltiples aplicaciones. [6]. *Ver Anexo 2.*

El servidor de la capa intermedia (también conocido como servidor de aplicaciones) centraliza la lógica de las aplicaciones, haciendo que la administración de cambios sea más sencilla. En arquitecturas más simples, cualquier cambio en la lógica, implica reescribir todas las aplicaciones que dependan de ésta. [6]

1.5.3 PHP (PHP: Hypertext Prepocessor).

El PHP, es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor. El PHP originalmente diseñado en Perl, seguidos por la escritura de un grupo de CGI binarios escritos en el lenguaje C por el programador Danés-Canadiense Rasmus Lerdorf en el año 1994 para mantener un controlsobre quien visitaba su currículum y guardar ciertos datos, como la cantidad de tráfico que su página Web recibía. En los siguientes tres años, se fue convirtiendo en lo que se conoce como PHP/FI 2.0. Esta forma de programar llegó a muchos usuarios, pero el lenguaje no tomó el peso actual hasta que dos programadores israelíes de Technion, Zeev Suraski y Andi Gutmans reescribieron el analizador gramatical en el año 1997, y crearon la base del PHP 3, cambiando el nombre del lenguaje a la forma actual. Para 1999, Suraski y Gutmans reescribieron el código de PHP, produciendo lo que hoy se conoce como Zend Engine o motor Zend. En mayo del 2000, PHP 4 fue lanzado bajo el poder del motor Zend Engine 1.0. El 13 de julio de 2004, PHP 5 fue lanzado, utilizando el motor Zend Engine II. La versión más reciente de PHP es la 5.1.4, que incluye el novedoso PDO (PHP Data Objects) y mejoras utilizando las ventajas que provee el nuevo Zend Engine 2. Según estudios, más de un millón de servidores tienen esta capacidad implementada y los números continúan creciendo. [6]

Una de sus características más potentes es su soporte para gran cantidad de bases de datos. Entre las que se pueden mencionar InterBase, mSQL, MySQL, Oracle, Informix,

PostgreSQL, entre otras. PHP también ofrece la integración con varias bibliotecas externas, que dan al desarrollador la posibilidad de realizar cualquier tarea, desde generar documentos en pdf (Portable Document Format) hasta analizar código XML (eXtensible Markup Language) y últimamente también para la creación de otro tipo de programas incluyendo aplicaciones con interfaz gráfica usando la librería GTK+. [6]

Es software libre, lo que implica menos costes y servidores más baratos que otras alternativas. Es muy rápido y su integración con la base de datos MySQL y el servidor Apache, le permite constituirse como una de las alternativas más atractivas del mercado. [6]

Es multiplataforma, funciona tanto para Unix (con Apache) como para Windows (con Microsoft Internet Information Server) de forma que el código que se haya creado para una de ellas no tiene porqué modificarse al pasar a la otra.

Su sintaxis está inspirada en C, ligeramente modificada para adaptarlo al entorno en el que trabaja, de modo que si se está familiarizado con esta sintaxis, le resultará muy fácil aprender PHP. Su librería estándar es realmente amplia, lo que permite reducir los llamados "costes ocultos", uno de los principales defectos de ASP (Active Server Pages). [6]

PHP tiene una de las comunidades más grandes en Internet, con lo que no es complicado encontrar ayuda, documentación, artículos, noticias, y más recursos. Ofrece una solución simple y universal para las paginaciones dinámicas del Web de fácil programación. Su diseño elegante lo hace perceptiblemente más fácil de mantener y ponerse al día, a diferencia con el código de otros lenguajes. [6]

Debido a su amplia distribución PHP esta perfectamente soportado por una gran comunidad de desarrolladores. Como producto de código abierto, PHP goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y se reparan rápidamente. El código se pone al día

continuamente con mejoras y extensiones de lenguaje para ampliar las capacidades de PHP. [6]

El funcionamiento del PHP se puede describir a través de los pasos siguientes: [8]

- Escribir en las páginas HTML pero con el código PHP dentro.
- Guardar la página en el servidor Web.
- Un navegador solicita una página al servidor.
- El servidor interpreta el código PHP.
- El servidor envía el resultado del conjunto de código HTML y el resultado del código PHP que también es HTML.

En ningún caso se envía código PHP al navegador, por lo que todas las operaciones realizadas son transparentes al usuario, el código PHP es ejecutado en el servidor y el resultado enviado al navegador. El resultado es normalmente una página HTML. Por lo que al usuario le parecerá que está visitando una página HTML que cualquier navegador puede interpretar. [8]

Al ser PHP un lenguaje que se ejecuta en el servidor no es necesario que el navegador lo soporte, es independiente del navegador, pero sin embargo para que sus páginas PHP funcionen, el servidor donde están alojadas debe soportar PHP. PHP se encuentra libre en el mercado y se puede acceder a él por medio de Internet. [8]

Después de seis años, y después que la comunidad ha revisado el paquete de legados que ha dejado el PHP, se han realizado cambios estructurales en el lenguaje para ofrecer innovación en el nuevo PHP 5 y solucionar muchos de los problemas encontrados en PHP 4.

Afortunadamente, lo nuevo de PHP 5 mejora muchas áreas en el lenguaje y su ejecución, como por ejemplo: [9]

- Programación orientada a objetos (OOP).
- MySQL.
- XML.

- Integración nativa con el Zend Engine.

Los diseñadores de PHP5 han realizado un cambio radical en el tratamiento de las variables objeto: en PHP5 todas las variables que nombran objetos son en realidad referencias. No hay que usar el operador '&' ni en las asignaciones, ni en el paso de parámetros que son objetos, ahorrándose con ello gran cantidad de potenciales errores. [10]

La principal novedad en las clases de PHP5 es la inclusión de modificadores de control de acceso para implementar la encapsulación --piedra angular en la programación orientada a objetos de la que adolecía PHP4--.

PHP5 introduce tres palabras clave (public, private y protected) que sustituyen a *var* en la definición de variables miembro --atributos-- de la clase, y que preceden a la definición de funciones miembro --métodos-- .

Otros lenguajes como Perl (Practical Extraction and Report Language), ASP (Active Server Pages) y JSP (Java Server Pages) tienen características similares al PHP aunque poseen rasgos que los marcan y por ello los distingue, entre ellos podemos encontrar: [11]

- **Características multiplataformas:** Menos el ASP, que es solamente soportado por la plataforma Windows, los demás lenguajes están soportados en múltiples plataformas.
- **Velocidad de ejecución:** la velocidad es mayor en PHP, seguidos por PERL y JSP.
- **Disponibilidad de recursos:** actualmente los más utilizados en la Internet son el PHP y el JSP, siendo más utilizado en la publicación de artículos y códigos de ejemplos. PHP tiene una de las comunidades más grandes en Internet, al igual que la de Java.
- **Familiaridad con el lenguaje:** En la universidad los lenguajes más utilizados por los programadores es el ASP y el PHP.

De acuerdo a estas comparaciones, el PHP resulta mucho más favorecido, por tanto pensamos que es el adecuado para implementar la propuesta de sistema de este trabajo, particularmente PHP5.

1.5.4 Servidor Web Apache.

Un servidor de páginas Web es un programa que permite acceder a páginas Web alojadas en un ordenador. Hoy en día Apache es el servidor web más utilizado del mundo, encontrándose muy por encima de sus competidores, tanto gratuitos como comerciales. Es un software de código abierto que funciona sobre cualquier plataforma. Desde su origen ha evolucionado hasta convertirse en uno de los mejores servidores en términos de eficiencia, funcionalidad y velocidad, surgió en abril de 1996 y ya en julio del 2002 era utilizado por el 57% de los sitios Web de Internet. [11]

Tiene capacidad para servir páginas tanto de contenido estático, para lo que nos serviría sencillamente un viejo ordenador 486, como de contenido dinámico a través de otras herramientas soportadas que facilitan la actualización de los contenidos mediante bases de datos, ficheros u otras fuentes de información, es muy potente y altamente configurable. [11]

Los servidores Web suministran páginas web a los navegadores que lo solicitan. En términos más técnicos, los servidores Web soportan el Protocolo de Transferencia de Hypertexto como HTTP (HyperText Transfer Protocol), el estándar de Internet para comunicaciones web. Usando HTTP, un servidor Web envía páginas web en HTML y Common Gateway Interface (CGI), así como otros tipos de scritps a los navegadores o browsers cuando éstos los requieren. Cuando un usuario hace clic sobre un enlace a una página web, se envía una solicitud al servidor web para localizar los datos nombrados por ese enlace. El servidor web recibe esta solicitud y suministra los datos que le han sido solicitados o bien devuelve un mensaje de error. [11]

El servidor Apache es un software que esta estructurado en módulos, es decir, está dividido en muchas porciones de código que hacen referencia a diferentes aspectos o funcionalidades del servidor web. Esta modularidad es intencionada ya que la

configuración de cada módulo se hace mediante la configuración de las directivas que están contenidas dentro del módulo. Los módulos del Apache se pueden clasificar en tres categorías: [11]

- Módulos Base: Módulo con las funciones básicas del Apache.
- Módulos Multiproceso: Son los responsables de la unión con los puertos de la máquina, aceptando las peticiones y enviando a los hijos a atender a las peticiones.
- Módulos Adicionales: Cualquier otro módulo que le añada una funcionalidad al servidor.

Las funcionalidades más elementales se encuentran en el módulo base, siendo necesario un módulo multiproceso para manejar las peticiones. Se han diseñado varios módulos multiprocesos para cada uno de los sistemas operativos sobre los que se ejecuta el Apache, optimizando el rendimiento y rapidez del código. [11]

El resto de funcionalidades del servidor se consigue por medio de módulos adicionales que se pueden cargar. Para añadir un conjunto de utilidades al servidor, simplemente hay que añadirle un módulo, de forma que no es necesario volver a instalar el software. [11]

1.5.5 AJAX

AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript y XML asíncronos), es una técnica de desarrollo web para crear aplicaciones interactivas. Estas se ejecutan en el cliente, es decir, en el navegador del usuario, y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la velocidad de interacción en la misma. [21]

AJAX no constituye una tecnología en sí, sino que es un término que engloba a un grupo de éstas que trabajan conjuntamente y que se muestra a continuación: [21]

- XHTML (o HTML) y hojas de estilos en cascada (CSS) para el diseño que acompaña a la información.

- Document Object Model (DOM) accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones ECMAScript como JavaScript y JScript, para mostrar e interactuar dinámicamente con la información presentada.
- El objeto XMLHttpRequest para intercambiar datos asincrónicamente con el servidor web. En algunos frameworks y en algunas situaciones concretas, se usa un objeto IFrame en lugar del XMLHttpRequest para realizar dichos intercambios.
- XML es el formato usado comúnmente para la transferencia de vuelta al servidor, aunque cualquier formato puede funcionar, incluyendo HTML preformateado, texto plano, JSON y hasta EBML.

Existen diferencias significativas entre las aplicaciones web tradicionales y las aplicaciones desarrolladas en AJAX (*Ver Anexo 3*) así como sus patrones de interacción sincrónica para una aplicación Web tradicional y asincrónica para una aplicación AJAX (*Ver Anexo 4*).

Ventajas del AJAX:

- Interactividad

  Las aplicaciones AJAX se ejecutan en la máquina del usuario, manipulando la página actual dentro de sus navegadores usando métodos de Document Object Model. Puede ser usado para multitud de tareas como actualizar o eliminar registros, expandir formularios web, devolver peticiones simples de búsqueda, o editar árboles de categorías; todo sin tener la necesidad de recargar toda la página de HTML cada vez que se realiza un cambio. Generalmente solo requiere enviar pequeñas peticiones al servidor, y se devuelven respuestas relativamente cortas. Esto permite el desarrollo de aplicaciones interactivas con más interfaces de usuario más responsivas gracias al uso de las técnicas DHTML. [21]
- Portabilidad

  Las aplicaciones construidas con AJAX utilizan características bien documentadas presentes en todos los navegadores importantes en la mayoría de las plataformas existentes. Aunque esta situación podría cambiar en el futuro, en este momento, los usos de AJAX son efectivos entre plataformas. Mientras que la plataforma de AJAX está más restringida que la plataforma de Java, las

aplicaciones actuales de AJAX llenan con eficacia la parte de los Java Applets: ampliar el navegador con mini-aplicaciones ligeras. [21]

Desventajas del AJAX:

- Críticas de usabilidad

  Una de las mayores críticas contra el uso de AJAX en aplicaciones web es que puede fácilmente acabar con el comportamiento normal del botón atrás del navegador. Las diversas expectativas entre volver a una página que se ha modificado dinámicamente y la vuelta a una página estática pueden ser sutiles. Otro problema relacionado es que las actualizaciones dinámicas hacen difícil al usuario agregar a los marcadores/favoritos un momento particular de la aplicación. [21]

- JavaScript

  Aunque AJAX no necesita ningún tipo de plug-in para el navegador, requiere que los usuarios tengan el JavaScript activado. Esto se aplica a todos los navegadores que soportan esta tecnología excepto para Microsoft Internet Explorer 6 y anteriores los cuales necesitan también tener el ActiveX activado, ya que el objeto XMLHttpRequest está implementado junto con el ActiveX en este navegador.

  Como ocurre con las aplicaciones DHTML, las de AJAX deben de ser probadas rigurosamente para adaptarse a los diferentes navegadores y plataformas. [21]

1.5.6 Patrones de Arquitectura.

Un patrón es un modelo que podemos seguir para realizar algo. Los patrones surgen de la experiencia de seres humanos de tratar de lograr ciertos objetivos. Los patrones capturan la experiencia existente y probada para promover buenas prácticas. [19]

Los patrones de arquitectura expresan el esquema fundamental de organización para sistemas de software. Proveen un conjunto de subsistemas predefinidos; especifican sus responsabilidades e incluyen reglas y guías para organizar las relaciones entre ellos. Los patrones de arquitectura representan el nivel más alto en el sistema de

patrones propuesto en Pattern Oriented Software Architecture - Volume 1. Ayudan a especificar la estructura fundamental de una aplicación. Cada actividad de desarrollo es gobernada por esta estructura; por ejemplo, el diseño detallado de los subsistemas, la comunicación y colaboración entre diferentes partes del sistema, etc. Cada patrón de arquitectura ayuda a conseguir una propiedad específica en el sistema global; por ejemplo, la adaptabilidad de la interfaz de usuario. [22]
Un ejemplo de este tipo de patrón lo constituye el Modelo Vista Controlador.

1.5.6.1 Modelo Vista Controlador (MVC).

Son muchas las empresas que deciden pasar sus aplicaciones a la arquitectura modelo vista controlador para documentar más fácilmente el código, ahorrar espacio y en caso de no disponer de diseñadores web, poder contratar los servicios de un diseñador que no sepa mucho de programación que les haga las vistas. [20]

El Modelo Vista Controlador (MVC) se introduce inicialmente en la comunidad de desarrolladores de Smalltalk-80. MVC divide una aplicación interactiva en 3 áreas: procesamiento, salida y entrada. Para esto, utiliza las siguientes abstracciones: Modelo, Vista y Controlador. [22]

- El **Modelo** es todo acceso a datos, y las funciones que llevan lo que llaman "lógica de negocio", o sea datos y reglas de negocio. Lleva un registro de las vistas y controladores del sistema. Cada acceso a datos se pone en su función individual porque, de esta forma, si se cambia de gestor de bases de datos este cambio sólo afecta a estas funciones, no al resto de la aplicación. Tener el modelo bien delimitado permite la existencia de varias aplicaciones que compartan el mismo modelo (por ejemplo, una aplicación "tienda" y una "contabilidad" que accedan a las bases de datos de inventario, ventas,.. etc.). [20]
- La **Vista**, en una aplicación web, es el HTML y lo necesario para convertir datos en HTML. O sea muestra la información del modelo al usuario. Tienen un registro de su controlador asociado (normalmente porque además lo instancia). Pueden dar el servicio de "Actualización()", para que sea invocado por el controlador o por el modelo. Tener la vista separada del controlador permite cambiar la

aplicación para que genere, en lugar de HTML, algo distinto (por ejemplo, WML), sin tener que tocar más que una parte completamente delimitada del código. [20]

- El **Controlador** es lo que une la vista y el modelo. Por ejemplo, son las funciones que toman los valores de un formulario, consultan la base de datos (a través del modelo) y producen valores, que la vista tomará y convertirá en HTML. En resumen gestiona las entradas del usuario. Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.). Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. De este modo, el código que "hace algo" está perfectamente separado del código dedicado a crear HTML, lo que ayuda a evitar el spaghetti. [20]

Las Vistas y los Controladores conforman la interfaz de usuario. Un mecanismo de propagación de cambios asegura la consistencia entre la interfaz y el modelo. La separación del modelo de los componentes vista y del controlador permite tener múltiples vistas del mismo modelo. Si el usuario cambia el modelo a través del controlador de una vista, todas las otras vistas dependientes deben reflejar los cambios. Por lo tanto, el modelo notifica a todas las vistas siempre que sus datos cambien. Las vistas, en cambio, recuperan los nuevos datos del modelo y actualizan la información que muestran al usuario.

Un ejemplo típico del funcionamiento de este patrón se puede observar en el *Anexo 5*. La estructura de este patrón se puede observar en el *Anexo 6*.

Este patrón es muy popular y ha sido portado a una gran cantidad de entornos y frameworks como entre los que se encuentran WinForms, ASP .Net, etc. Las herramientas de programación visual como Visual Basic, Visual Studio .Net, etc., siguen también alguna variante de este esquema. El MVC es un patrón ampliamente utilizado en múltiples plataformas y lenguajes. Algunos de sus principales beneficios son: [22]

- Menor acoplamiento
    - Desacopla las vistas de los modelos

    - Desacopla los modelos de la forma en que se muestran e ingresan los datos
- Mayor cohesión
    - Cada elemento del patrón esta altamente especializado en su tarea (la vista en mostrar datos al usuario, el controlador en las entradas y el modelo en su objetivo de negocio)
- Las vistas proveen mayor flexibilidad y agilidad
    - Se puede crear múltiples vistas de un modelo
    - Se puede crear, añadir, modificar y eliminar nuevas vistas dinámicamente
    - Las vistas pueden anidarse
    - Se puede cambiar el modo en que una vista responde al usuario sin cambiar su representación visual
    - Se puede sincronizar las vistas
    - Las vistas pueden concentrarse en diferentes aspectos del modelo
- Mayor facilidad para el desarrollo de clientes ricos en múltiples dispositivos y canales
    - Una vista para cada dispositivo que puede varias según sus capacidades
    - Una vista para la Web y otra para aplicaciones de escritorio
- Más claridad de diseño
- Facilita el mantenimiento
- Mayor escalabilidad

Un patrón de arquitectura puede contener varios patrones de diseño, por ende el patrón de arquitectura MVC contiene (o puede contener) los siguientes patrones de diseño: [22]

- **Observer:** Para el mecanismo de publicación y suscripción que permite la notificación de los cambios en el modelo a las vistas.
- **Composite:** Para la creación de vistas compuestas. Utilizando este patrón podemos crear una jerarquía de vistas y tratar a cada vista compuesta igual que una a una vista normal.

- **Strategy:** En la relación entre las vistas y los controladores. Utilizando este patrón podemos cambiar dinámicamente o en tiempo de compilación los algoritmos del controlador mediante los cuales responde a su entorno.
- **Factory Method:** Para especificar la clase controlador predeterminada de una vista.
- **Decorator:** Para añadir capacidades adicionales a una vista (por ejemplo, scroll).
- **Proxy:** Para distribuir la arquitectura (Modelo y Vista-Controlador) en diferentes emplazamientos.

Los Patrones de Diseño (Design Patterns) son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. [19]

Estos se dividen en tres grandes categorías:

- Patrones Creacionales
  Solucionan problemas de creación de instancias. Nos ayudan a encapsular y abstraer dicha creación.
- Patrones Estructurales
  Solucionan problemas de composición (agregación) de clases y objetos.
- Patrones de Comportamiento
  Soluciones respecto a la interacción y responsabilidades entre clases y objetos, así como los algoritmos que encapsulan.

Los patrones de diseño han contribuido a dar flexibilidad y extensibilidad a nuestros diseños. Pero en adición, han demostrado ser una forma muy útil (exitosa) de reutilizar diseño, ya que ellos no sólo nombran, abstraen e identifican aspectos claves de estructuras comunes de diseño, sino que generalmente son descritos en una forma específica documental, haciendo su comprensión y aplicación fácil para el conjunto de desarrolladores. [19]

1.5.7 Sistemas de Gestión de Base de Datos.

Un Sistema de Gestión de Bases de Datos (SGBD) puede definirse como un paquete generalizado de software, que se ejecuta en un sistema computacional anfitrión, centralizando los accesos a los datos y actuando de interfaz entre los datos físicos y el usuario. Las principales funciones que debe cumplir un SGBD se relacionan con la creación y mantenimiento de la base de datos, el control de accesos, la manipulación de datos de acuerdo con las necesidades del usuario, el cumplimiento de las normas de tratamiento de datos, evitar redundancias e inconsistencias y mantener la integridad.

Un SGBD tiene los siguientes objetivos específicos: [8]

- Independencia de los datos y los programas de aplicación.
- Minimización de la redundancia.
- Integración y sincronización de las bases de datos.
- Integridad de los datos.
- Seguridad y protección de los datos.
- Facilidad de manipulación de la información.
- Control centralizado.

La información es representada a través de tuplas, las cuales describen el fenómeno, proceso o ente de la realidad objetiva que se está analizando y se representan a través de tablas. [8]

Entre los SGBD comúnmente utilizados en el mundo tenemos Oracle, MySQL, Microsoft SQL Server, PostgreSQL, Interbase, entre otros. Todos estos presentan un enfoque relacional con un buen basamento matemático centrado en el Álgebra Relacional. [6]

Todos los sistemas mencionados anteriormente facilitan el trabajo con la base de datos y tienen características que los diferencian, por ejemplo: [11]

- **Oracle**: requiere de una licencia para poderlo utilizar, es decir, es necesario pagar para poder utilizarlo.
- **Microsoft SQL Server**: no es multiplataforma, solo puede ser utilizado con el sistema operativo Windows que está patrocinado por la compañía Microsoft.

- **MySQL**: PostgreSQL soporta un subconjunto de SQL92 MAYOR que el que soporta MySQL.

Como SGBD se seleccionó el PostgreSQL por las ventajas que ofrece y por requerimientos del cliente.

1.5.7.1 PostgreSQL.

PostgreSQL es un sistema de gestión de bases de datos Objeto-Relacionales (ORDBMS) libre, liberado bajo la licencia BSD (Berkeley Software Distribution). Es una alternativa a otros sistemas de bases de datos de código abierto (como MySQL, Firebird y MaxDB), así como sistemas propietarios como Oracle y SQLServer. El mismo ha sido desarrollado de varias formas desde 1977.

En 1994, Andrew Yu y Jolly Chen añadieron un intérprete de lenguaje SQL a Postgres. Postgres95 fue lanzada a continuación a la Web para que encontrara su propio hueco en el mundo como un descendiente de dominio público y código abierto del código original Postgres de Berkeley.

En 1996, debido a un nuevo esfuerzo de código abierto y a la incrementada funcionalidad del software, *Postgres* fue renombrado a *PostgreSQL*, tras un breve periplo como *Postgres95*. El proyecto *PostgreSQL* sigue actualmente un activo proceso de desarrollo a nivel mundial gracias a un equipo de desarrolladores y contribuidores de código abierto. PostgreSQL está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo. Posee muchas características que tradicionalmente sólo se podían ver en productos comerciales de alto calibre.

La siguiente es una breve lista de algunas de esas características, a partir de PostgreSQL 7.1.x.

<u>DBMS Objeto-Relacional</u>

PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transacciones, optimización de consultas, herencia, y arreglos.

Altamente Extensible

PostgreSQL soporta operadores, funciones métodos de acceso y tipos de datos definidos por el usuario.

Soporte SQL Comprensivo

PostgreSQL soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92.

Integridad Referencial

PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.

Lenguajes Procedurales

PostgreSQL tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL.

Otra ventaja de PostgreSQL es su habilidad para usar Perl, Python, o TCL (Tool Command Language) como lenguaje procedural embebido.

MVCC

MVCC, o Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control), es la tecnología que PostgreSQL usa para evitar bloqueos innecesarios. Los bloqueos son provocados por usuarios que están escribiendo en la base de datos. Resumiendo, el lector está bloqueado por los escritores que están actualizando registros.

Mediante el uso de MVCC, PostgreSQL evita este problema por completo. MVCC está considerado mejor que el bloqueo a nivel de fila porque un lector nunca es bloqueado por un escritor. En su lugar, PostgreSQL mantiene una ruta a todas las transacciones realizadas por los usuarios de la base de datos. PostgreSQL es capaz entonces de manejar los registros sin necesidad de que los usuarios tengan que esperar a que los registros estén disponibles.

Cliente/Servidor

PostgreSQL usa una arquitectura proceso-por-usuario cliente/servidor. Esta es similar al método del Apache 1.3.x para manejar procesos. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL.

Las principales mejoras en PostgreSQL incluyen: [12]

- Los bloqueos de tabla han sido sustituidos por el control de concurrencia multi-versión, el cual permite a los accesos de sólo lectura continuar leyendo datos consistentes durante la actualización de registros, y permite copias de seguridad en caliente desde pg_dump mientras la base de datos permanece disponible para consultas.
- Se han implementado importantes características del motor de datos, incluyendo subconsultas, valores por defecto, restricciones a valores en los campos (constraints) y disparadores (triggers).
- Se han añadido características adicionales que cumplen el estándar SQL92, incluyendo claves primarias, identificadores entrecomillados, forzado de tipos cadenas literales, conversión de tipos y entrada de enteros binarios y hexadecimales.
- Los tipos internos han sido mejorados, incluyendo nuevos tipos de fecha/hora de rango amplio y soporte para tipos geométricos adicionales.
- La velocidad del código del motor de datos ha sido incrementada aproximadamente en un 20-40%, y su tiempo de arranque ha bajado el 80% desde que la versión 6.0 fue lanzada.

1.5.8 Proceso de Desarrollo.

Cada día la producción de software busca adecuarse más a las necesidades del usuario, esto trae como consecuencia que aumente en tamaño y complejidad.

Para lograr la productividad del software se necesita un proceso que integre las múltiples facetas del desarrollo del mismo. Se hace necesario definir la metodología de ingeniería del software que guiará el proceso de automatización, se ha escogido el Proceso Unificado de Desarrollo de Software (RUP). El Proceso Unificado de Rational, (Rational Unified Process, de ahí las siglas RUP), fue publicado en 1998 como resultado de varios años de experiencia.[7].

RUP es un proceso de desarrollo de software, o sea, conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software. Es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos. [14]

Es un proceso basado en componentes, que utiliza el Lenguaje Unificado de Modelado (UML) para preparar todos los esquemas de un sistema software. No obstante, los verdaderos aspectos definitorios del Proceso Unificado se resumen en que está dirigido por casos de uso, este avanza a través de una serie de flujos de trabajo, los cuales se muestran en el A*nexo 7*, que parten de los casos de uso; centrado en la arquitectura y es iterativo e incremental. [14]

Está acompañado de una herramienta muy buena que soporta cada uno de los procesos que necesitamos: Rational Rose Enterprise Edition 2003. Además cubre el ciclo de vida de desarrollo de un proyecto y toma en cuenta las mejores prácticas a utilizar en el modelo de desarrollo de software.

Lenguaje Unificado de Modelado (UML).

UML (Unified Modeling Language) es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema de software. [15]

Sus creadores pretendieron con este lenguaje, unificar las experiencias acumuladas sobre técnicas de modelado e incorporar las mejores prácticas en un acercamiento estándar.

El UML permite a los creadores de sistemas generar diseños que capturen sus ideas en una forma convencional y fácil de comprender para comunicarlas a otras personas que estén involucradas en el proceso de desarrollo de los sistemas, esto se lleva a cabo mediante un conjunto de símbolos y diagramas. [16]

El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas y proporciona un estándar que permite al analista de sistemas generar un anteproyecto de varias facetas que sean comprensibles para los clientes, desarrolladores y todos aquellos que estén involucrados en el proceso de desarrollo. Un modelo UML indica que es lo que supuestamente hará el sistema pero no como lo hará. [16]

De forma general las principales características son: [6]

- Lenguaje unificado para la modelación de sistemas.
- Tecnología orientada a objetos.
- El cliente participa en todas las etapas del proyecto.
- Corrección de errores viables en todas las etapa.s
- Aplicable para tratar asuntos de escala inherentes a sistemas complejos de misión crítica, tiempo real y cliente/servidor.

Existen varias herramientas CASE (Computer-Aided Systems Engineering), que dan asistencia a analistas, ingenieros de software y desarrolladores durante el ciclo de vida de desarrollo de un software, pero es Rational Rose líder en el modelado del desarrollo de los proyectos y es esta precisamente la que se utiliza en la modelación de este proyecto. La herramienta fue desarrollada por los creadores de UML, utilizando la notación estándar en la arquitectura de software. Esta herramienta integra todos los

elementos que propone la metodología RUP para cubrir el ciclo de vida de un proyecto. [7]

1.5.9 Herramientas utilizadas.

1.5.9.1 Diseño de interfaz.

Macromedia Dreamweaver MX es uno de los editores de desarrollo Web más utilizado a nivel profesional para la creación de sitios Web. Su amplio abanico de herramientas permite crear desde la más simple página Web personal hasta el sitio Web más completo y complejo para una gran empresa y utilizar casi todos los recursos de la Web. [7]

Este editor de HTML profesional para el diseño, codificación y desarrollo de páginas, sitios y aplicaciones Web; permite la edición visual, o sea, crear páginas rápidamente sin escribir una línea de código, así como también la codificación manual. [7]

Dreamweaver ayuda además a construir aplicaciones Web dinámicas apoyadas en bases de datos. [7]

Es completamente personalizable. Se pueden crear objetos y comandos propios, modificar los accesos directos de teclado, e incluso escribir código JavaScript para extender las capacidades del Dreamweaver con nuevos comportamientos. Soporta varias tecnologías del servidor para la construcción de aplicaciones Web, tales como: Macromedia ColdFusion, Microsoft ASP, Microsoft ASP.NET, Sun JavaServer Pages (JSP) y PHP. [13]

1.5.9.2 Zend Studio.

Zend Studio es uno de los ambientes de desarrollo integrado o Integrated Development Environment (IDE) disponible para desarrolladores profesionales que agrupa todos los componentes de desarrollo necesarios para ciclo de desarrollo de aplicaciones PHP. A través de un comprensivo conjunto de herramientas de edición, depurado, análisis, optimización y bases de datos, Zend Studio acelera los ciclos de desarrollo y simplifica los proyectos complejos. [18]

1.5.9.3 Adobe Photoshop.

Adobe Photoshop CS para el tratamiento de los gráficos. Es una herramienta muy poderosa para crear cualquier tipo de gráficos, su integración con Adobe ImageReady hacen que crear complicados gráficos para la Web sea una tarea muy fácil.

1.5.9.4 Rational Rose.

Existen herramientas CASE de trabajo visuales como el Analise, el Designe, el Rational Rose, que permiten realizar el modelado del desarrollo de los proyectos, en la actualidad la mejor y más utilizada en el mercado mundial es Rational Rose y es la que se utiliza en la modelación de este proyecto. [17]

Rational Rose cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases y entregables.

Es la herramienta CASE que comercializan los desarrolladores de UML y que soporta de forma completa la especificación del UML.

Rose es una herramienta con plataforma independiente que ayuda a la comunicación entre los miembros de equipo, a monitorear el tiempo de desarrollo y a entender el entorno de los sistemas. Una de las grandes ventajas de Rose es que utiliza la notación estándar en la arquitectura de software(UML), la cual permite a los arquitectos de software y desarrolladores visualizar el sistema completo utilizando un lenguaje común, además los diseñadores pueden modelar sus componentes e interfaces en forma individual y luego unirlos con otros componentes del proyecto. [17]

Se decidió que se utilizaría el Rational Rose Enterprise Edition 2003, para sustentar la documentación, como modelador visual de la notación UML (Unified Modeling Language) para la confección de los diagramas que se ilustran en este documento. Esta herramienta es muy completa y ofrece amplias potencialidades.

1.5.9.5 PgAdmin.

Es desarrollado por una comunidad de los expertos de PostgreSQL de varias partes del mundo y está disponible en más de una docena de idiomas. Se trata de una herramienta para la administración de bases de datos PostgreSQL. Su uso se puede extender hacia las plataformas de Linux, de FreeBSD, de Solaris, del Mac OSX y de Windows.

1.6 Conclusiones.

En este capítulo se exponen las condiciones y problemas que rodean el objeto de estudio; y a través de los conceptos y definiciones planteadas. Se evidencia la necesidad de implementar un software que permita controlar las tareas desarrolladas durante la Misión Milagro. Para desarrollar el sistema se hace uso de la tecnología para la programación de páginas dinámicas el lenguaje PHP5 y con soporte de base de datos en PostgreSQL. El proceso de desarrollo es RUP, el cual está basado en la orientación a objetos y el modelamiento visual usando UML, lo cual permite incorporar al proceso de desarrollo de software un mejor control de los requerimientos y cambios.

CAPÍTULO 2
Modelo del Negocio

2.1 Introducción.

Antes de comenzar a desarrollar un sistema es necesario comprender la organización bajo estudio y los procesos que en ella tienen lugar, a fin de lograr una mejor comprensión del problema a resolver y el común entendimiento entre clientes y desarrolladores; para lo cual se realiza la modelación del negocio.

El modelo del negocio posibilita obtener una visión más clara del proceso en cuestión, por ello en este capítulo se exponen las políticas y condiciones que deben cumplirse, entendidas como reglas del negocio asociadas al campo de acción. Se describen los actores y trabajadores del negocio y el modelo de objetos.

2.2 Modelo del negocio propuesto.

El primer paso del modelado del negocio consiste en capturar y definir los procesos de negocio de la organización bajo estudio. En el capítulo anterior se hizo una descripción general de los procesos identificados en el negocio actual, así como un análisis crítico de la ejecución de los mismos. Teniendo en cuenta las deficiencias detectadas y bajo un análisis profundo de las fuentes de problemas potenciales se ha elaborado una propuesta de negocio que mantiene invariable el flujo de los procesos pero incurre en cambios de la ejecución de los mismos. A continuación se muestra la descripción detallada del negocio propuesto, en la misma se describe la ejecución de los siguientes procesos:

2.2.1 Proceso de Ubicar al hospitalizado.

Este proceso tiene como objetivo asignarle una ubicación al hospitalizado para de esta forma satisfacer sus necesidades de alojamiento durante su estancia en el centro.

Una vez que el puesto de mando de vuelo informa de la asignación de un vuelo, el puesto de mando de la UCI avisa a las clínicas que tienen posibilidades de ubicación atendiendo a la cantidad de pasajeros que trae dicho vuelo. Cuando los hospitalizados

arriban al centro, son conducidos a los puntos de acreditación donde se les recogen los datos personales, siendo luego trasladados a los puntos de ubicación donde los responsables de alojamiento de las clínicas son encargados de asignarles una ubicación de acuerdo a las condiciones del paciente.

2.2.2 Proceso de Reubicar al hospitalizado.

Este proceso abarca un conjunto de actividades que se activan una vez que el hospitalizado muestra al médico algún tipo de inconformidad respecto a la ubicación que le fue asignada o contrae una enfermedad contagiosa, motivo por el cual debe ser aislado del resto de los hospitalizados.

El médico es la persona que atiende la petición de cambio del hospitalizado o detecta la enfermedad y valora la situación. Si el médico considera que el cambio es necesario solicita al jefe de clínica el traslado de ubicación. El jefe de clínica a través del responsable de alojamiento consulta el estado de las disponibilidades y si existen condiciones en la clínica para efectuar la reubicación se procede a ello. En caso de que no estén creadas las condiciones informa al paciente la imposibilidad del cambio.

2.2.3 Proceso de Liberación de capacidades de hospitalizados.

Una vez que el paciente parte en un vuelo hacia su país de origen, el puesto de mando de vuelo informa de dicha salida al puesto de mando general y procede a conceder el alta médica al paciente y actualiza el listado de las capacidades. Si el paciente es trasladado a otro hospital el médico es el responsable de indicar el traslado y actualizar el listado de capacidades.

2.2.4 Proceso de Ubicación del personal de servicio.

Este proceso tiene como objetivo asignar una ubicación al personal de servicio de la misión.

Cuando arriba un trabajador de servicio a la UCI con necesidad de alojamiento, procede a acreditarse e inmediatamente el responsable de alojamiento es el encargado de asignarle una ubicación y actualizar el listado de las capacidades.

2.2.5 Proceso de Reubicación de personal de servicio.

Una vez que se le asigna una ubicación a un trabajador de servicio, existe la posibilidad de reubicarlo, para ello el responsable de alojamiento es quien autoriza el cambio y actualiza el listado de las capacidades.

2.2.6 Proceso de Liberación de capacidades del personal de servicio.

Una vez que concluye el período de misión del trabajador, el responsable de alojamiento verifica el estado de la capacidad donde estuvo alojado el trabajador con el objetivo de chequear la existencia y el estado de los medios entregados y autoriza la salida del mismo, actualizando luego el listado de las capacidades.

2.3 Reglas del negocio a considerar.

- El puesto de mando de vuelo se encarga de informar de la asignación de un vuelo a la UCI, así como la cantidad de hospitalizados que trae dicho vuelo.
- El puesto de mando de la UCI debe consultar el listado de capacidades para seleccionar las clínicas que tienen posibilidades reales de ubicación, enviándoles luego la información de que serán utilizadas para alojar a los nuevos ingresos.
- El responsable del alojamiento tiene la responsabilidad de asignarle una ubicación al personal que arribe al centro con necesidad de alojamiento.
- Los hospitalizados con discapacidad física deben ser alojados en las plantas bajas de los edificios.
- Las personas de distintos sexos sin parentesco familiar no deben ser ubicadas en el mismo cuarto.
- Si un acompañante viene con varios pacientes debe tratarse por todos los medios que sean ubicados lo más cerca posible.
- El médico es la persona que valora la posibilidad de reubicación de un hospitalizado.
- El responsable de alojamiento consulta el listado de las capacidades y determina si es posible el cambio de ubicación atendiendo al estado en que se encuentra la clínica.
- El médico es el único autorizado a dar de baja al hospitalizado y es la persona que además libera la capacidad en el listado de capacidades.

- El responsable de alojamiento es el encargado de gestionar el alojamiento para los trabajadores de servicio de la misión.

2.4 Actores del negocio.

Un actor del negocio es cualquier individuo, grupo, organización, máquina o sistema de información externo que interactúa con el negocio. El término *actor* significa el rol que algo o alguien juega cuando interactúa con el negocio para beneficiarse de sus resultados. De acuerdo con esta idea un actor del negocio representa un tipo particular de usuario del negocio más que un usuario físico, ya que varios usuarios físicos pueden realizar el mismo papel en relación al negocio, o sea, ser instancias de un mismo actor. [6]

A continuación se muestra en la tabla 2.1, los actores del negocio y su correspondiente justificación:

Actores del negocio	Justificación
Hospitalizado	Es el principal beneficiado de los servicios que se prestan en la misión. Hospitalizado puede ser paciente o acompañante.
Trabajador de Servicio	Es el encargado de velar por la salud y bienestar del paciente durante su estancia en el hospital. Puede ser un trabajador de servicios médicos, donde se incluyen técnicos de salud, enfermeras y médicos. Y puede ser un trabajador de servicios varios, donde se incluyen profesores, estudiantes, trabajadores de la UCI, etc.

Tabla 2.1 Descripción de los actores del negocio.

2.4.1 Diagrama de casos de uso del negocio.

El diagrama de casos de uso del negocio representa gráficamente los procesos del negocio y su interacción con los actores del negocio. A continuación se muestra la figura 2.1 correspondiente al diagrama de casos de uso del negocio.

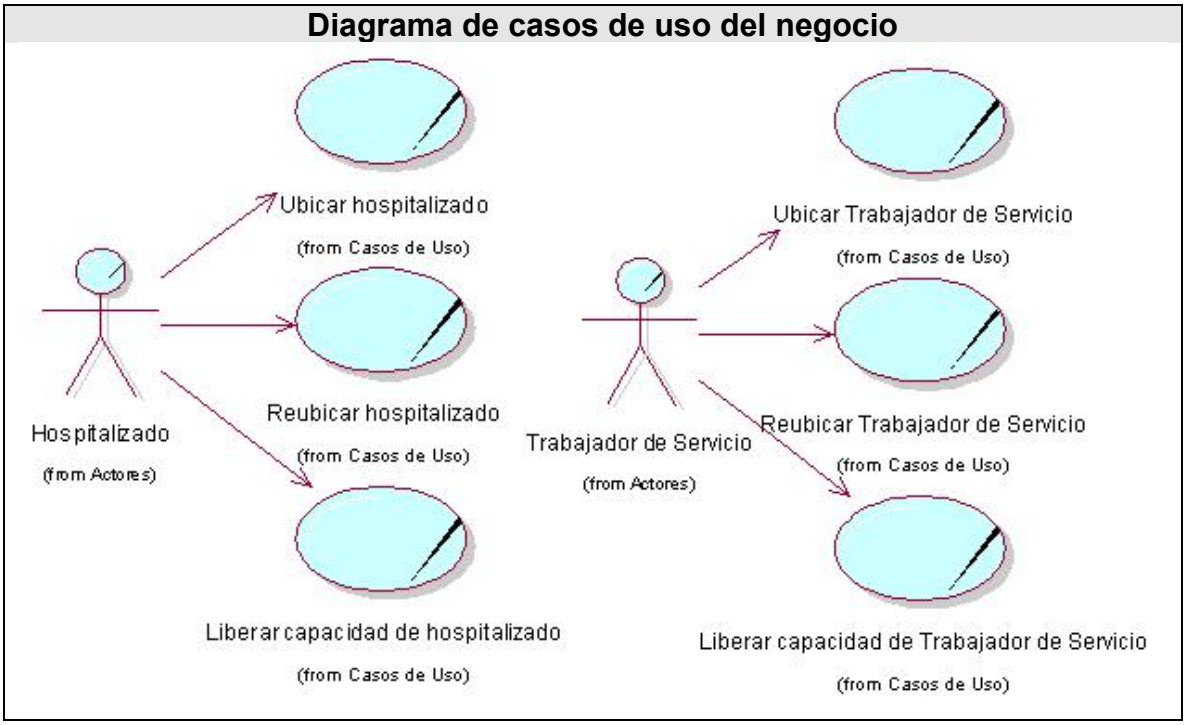


Figura 2.1 Diagrama de casos de uso del negocio.

2.5 Trabajadores del negocio.

Un trabajador define el comportamiento y las responsabilidades de un individuo que actúa en el negocio realizando una o varias actividades, interactuando con otros trabajadores del negocio y manipulando entidades del negocio. [6]

A continuación se muestran en la tabla 2.2, los trabajadores del negocio y su correspondiente justificación:

Trabajadores del negocio	Justificación
Responsable del Puesto de Mando de vuelo de la UCI	Es quien recibe las informaciones relacionadas con los vuelos asignados a la UCI, y toma las decisiones relacionadas a la conformación de los vuelos de salida.
Responsable del Puesto de Mando de la UCI	Es quien controla el funcionamiento del hospital, coordina con las instituciones involucradas, actualiza la estadística de capacidades e informa para la conformación de entrada del hospital y elabora y envía el

	Sistema de Partes.
Responsable de alojamiento	Es el encargado de gestionar los procesos relacionados con el alojamiento.
Acreditador	Es el encargado de recoger los datos de las personas que se acreditan en la misión.

Tabla 2.2 Descripción de los trabajadores del negocio.

2.6 Descripción de los casos de uso del negocio.

2.6.1 Caso de uso "Ubicar Hospitalizado".

Especificación textual en formato general

Caso de uso del negocio: Ubicar Hospitalizado	
Actores del negocio: Hospitalizado.	
Propósito: Permitir que el hospitalizado pueda ubicarse una vez presente en la Universidad.	
Resumen: El caso de uso se inicia cuando el Hospitalizado llega al país. Luego se siguen una serie de procedimientos hasta que es ubicado por el Responsable de alojamiento de la clínica correspondiente, culminando así el caso de uso.	
Curso normal de los eventos:	
Acción del actor	**Respuesta del proceso de negocio**
1. El hospitalizado arriba al país.	1.1. El Puesto de mando de vuelo informa al puesto de mando de la UCI sobre la llegada del vuelo así como la cantidad de hospitalizados que trae. 1.2. El puesto de mando de la UCI recibe dicha información. 1.3. Seguidamente consulta el listado de las capacidades de cada clínica y determina que clínicas se van a utilizar para

	alojar a los hospitalizados del vuelo. 1.4. Luego informa a las clínicas seleccionadas. 1.5. El responsable de alojamiento de la clínica recibe la información y se persona en los puntos de acreditación, en espera del arribo al centro de los hospitalizados.
2. Arriba a la universidad y es trasladado a los puntos de acreditación.	2.1. El acreditador recoge los datos personales de los hospitalizados. 2.2. El responsable de alojamiento de cada clínica analiza las capacidades. 2.3. Le asigna la ubicación al hospitalizado. 2.4. Actualiza el listado de las capacidades. 2.5. Informa la ubicación asignada.
3. Recibe la asignación.	
Prioridad: Alta.	
Mejoras: Gestionar a partir de una fuente de datos central las capacidades disponibles en las clínicas mostrando un conjunto de datos útiles para la toma rápida de decisiones por parte del responsable de alojamiento.	
Otras secciones:	

Tabla 2.3 Especificación textual del caso de uso del negocio “Ubicar hospitalizado”.

2.6.1.1 Diagrama de actividades.

Un diagrama de actividad demuestra la serie de actividades que deben ser realizadas en un proceso del negocio, así como las distintas rutas que se pueden ir desencadenando. Este es dividido en canales, donde cada canal representa el actor que está llevando a cabo la actividad y muestra cómo se utilizan las entidades del negocio. A continuación se muestra en la figura 2.2 el Diagrama de actividades del caso de uso del negocio “Ubicar hospitalizado”.

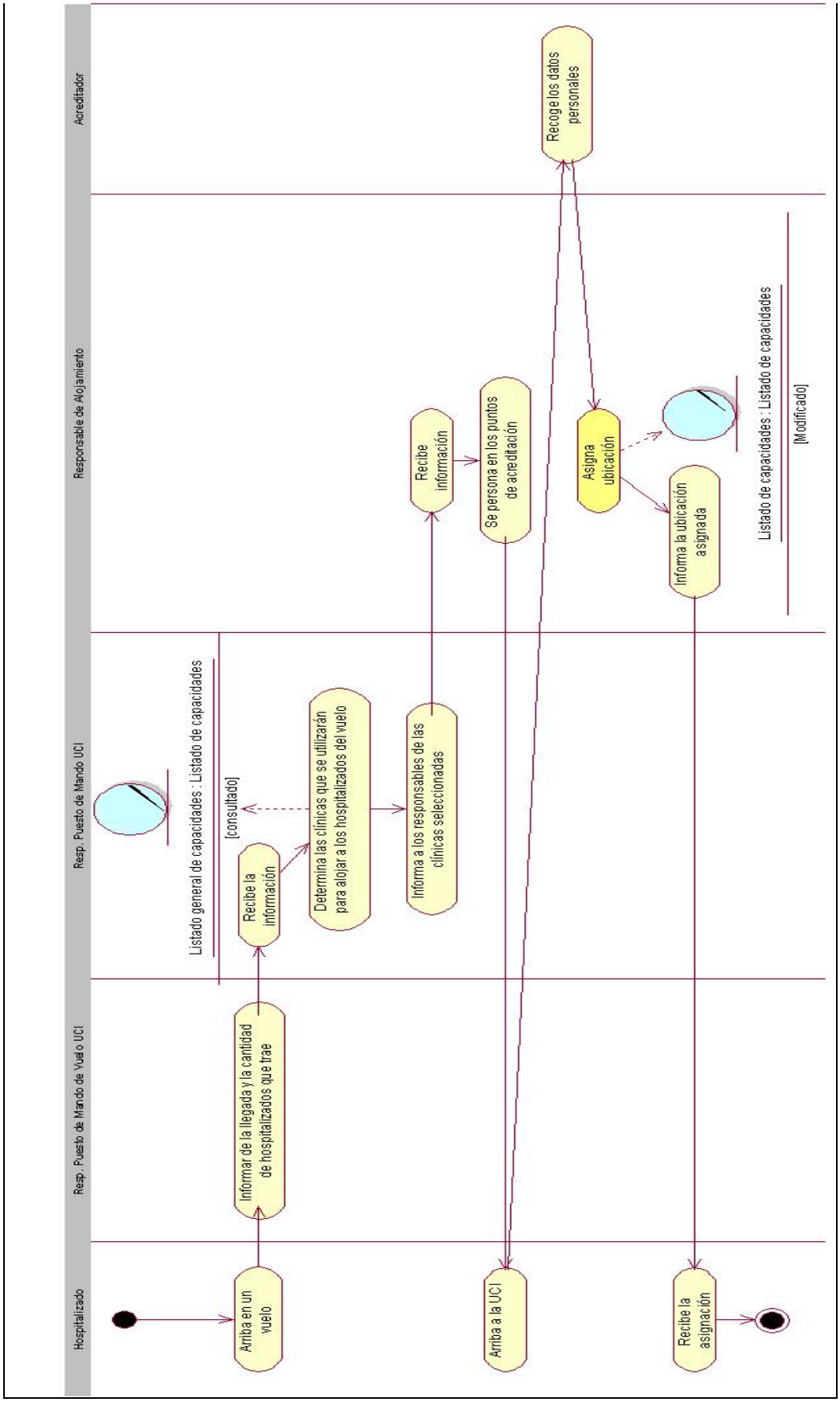

Hospitalizado
Resp. Puesto de Mando de Vuelo UCI
Resp. Puesto de Mando UCI
Responsable de Alojamiento
Acreditador
Arriba en un vuelo
Informar de la llegada y la cantidad de hospitalizados que trae
Recibe la información
[consultado]
Listado general de capacidades : Listado de capacidades
Determina las clínicas que se utilizarán para alojar a los hospitalizados del vuelo
Informa a los responsables de las clínicas seleccionadas
Recibe información
Se persona en los puntos de acreditación
Recoge los datos personales
Asigna ubicación
Informa la ubicación asignada
Listado de capacidades : Listado de capacidades
[Modificado]
Arriba a la UCI
Recibe la asignación

Figura 2.2 Diagrama de actividades del caso de uso del negocio “Ubicar hospitalizado”.

2.6.1.2 Diagrama de clases del modelo de objeto.

Un modelo de objetos del negocio es un modelo interno a un negocio. Describe como cada caso de uso del negocio es llevado a cabo por parte de un conjunto de trabajadores que utilizan un conjunto de entidades del negocio y unidades de trabajo. [14]

A continuación se muestra en la figura 2.3 el diagrama de clases del modelo de objetos del caso de uso del negocio “Ubicar hospitalizado”.

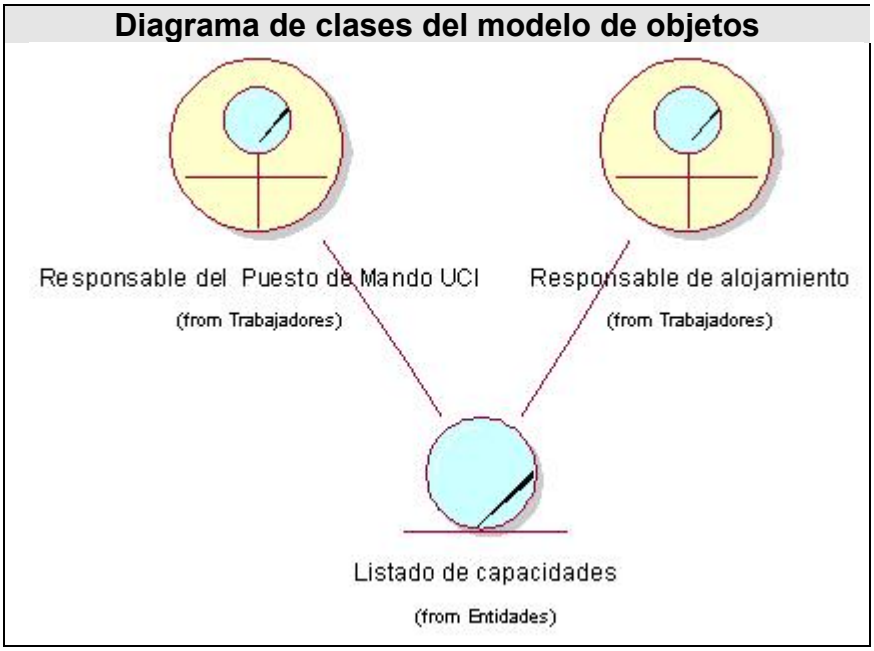


Figura 2.3 Diagrama de clases del modelo de objetos del caso de uso del negocio “Ubicar hospitalizado”.

2.6.2 Caso de uso “Reubicar hospitalizado”.

Especificación textual en formato general

Caso de uso del negocio: Reubicar Hospitalizado
Actores del negocio: Hospitalizado.
Propósito: Permitir la reubicación de un hospitalizado.
Resumen: El caso de uso se inicia cuando un paciente solicita un cambio de

ubicación o presenta algún tipo de enfermedad contagiosa, a partir de ahí comienza un proceso de toma de decisiones que concluye con la actualización del listado general de las capacidades si se reubica o cuando se le comunica al hospitalizado la imposibilidad del cambio, concluyendo así el caso de uso.	
Curso normal de los eventos:	
Acción del actor	**Respuesta del proceso de negocio**
1. Solicita cambio de ubicación.	1.1. El médico recibe la solicitud. 1.2. Valora si es necesario el cambio. 1.3. Solicita el cambio al Jefe de la clínica. 1.4. El jefe de la clínica recibe la solicitud de cambio. 1.5. Indica al responsable de alojamiento la asignación de una nueva ubicación de ser posible. 1.6. El responsable de alojamiento recibe la indicación. 1.7. Consulta el listado de capacidades de la clínica. 1.8. Asigna una nueva ubicación al hospitalizado de acuerdo a sus necesidades. 1.9. Actualiza el listado de las capacidades. 1.10. Informa la nueva ubicación.
2. Recibe la información	
Flujo alterno de los eventos:	
Acción del actor:	**Respuesta del proceso de negocio:**

	1.4. El médico informa al hospitalizado que el cambio no se efectuará.
2. Recibe la información.	
	1.8. El responsable de alojamiento informa al hospitalizado la imposibilidad de efectuar el cambio.
2. Recibe la información.	
1. Contrae una enfermedad contagiosa.	1.1. El médico detecta la enfermedad del hospitalizado. 1.2. Solicita al jefe de clínica el cambio necesario de ubicación. 1.3. El jefe de clínica indica al responsable de alojamiento el proceder a realizar el cambio. 1.4. El responsable de alojamiento asigna una nueva ubicación. 1.5. Actualiza el listado de las capacidades. 1.6. Informa la nueva ubicación.
2. Recibe la información.	
Prioridad: Media	
Mejoras: Gestionar de forma rápida y confiable la ubicación del paciente que se desea reubicar, así como la búsqueda asistida de una nueva ubicación.	
Otras secciones:	

Tabla 2.4 Especificación textual del caso de uso del negocio “Reubicar hospitalizado”.

2.6.2.1 Diagrama de actividades.

A continuación se muestra en la figura 2.4 el Diagrama de actividades del caso de uso del negocio “Reubicar hospitalizado”.

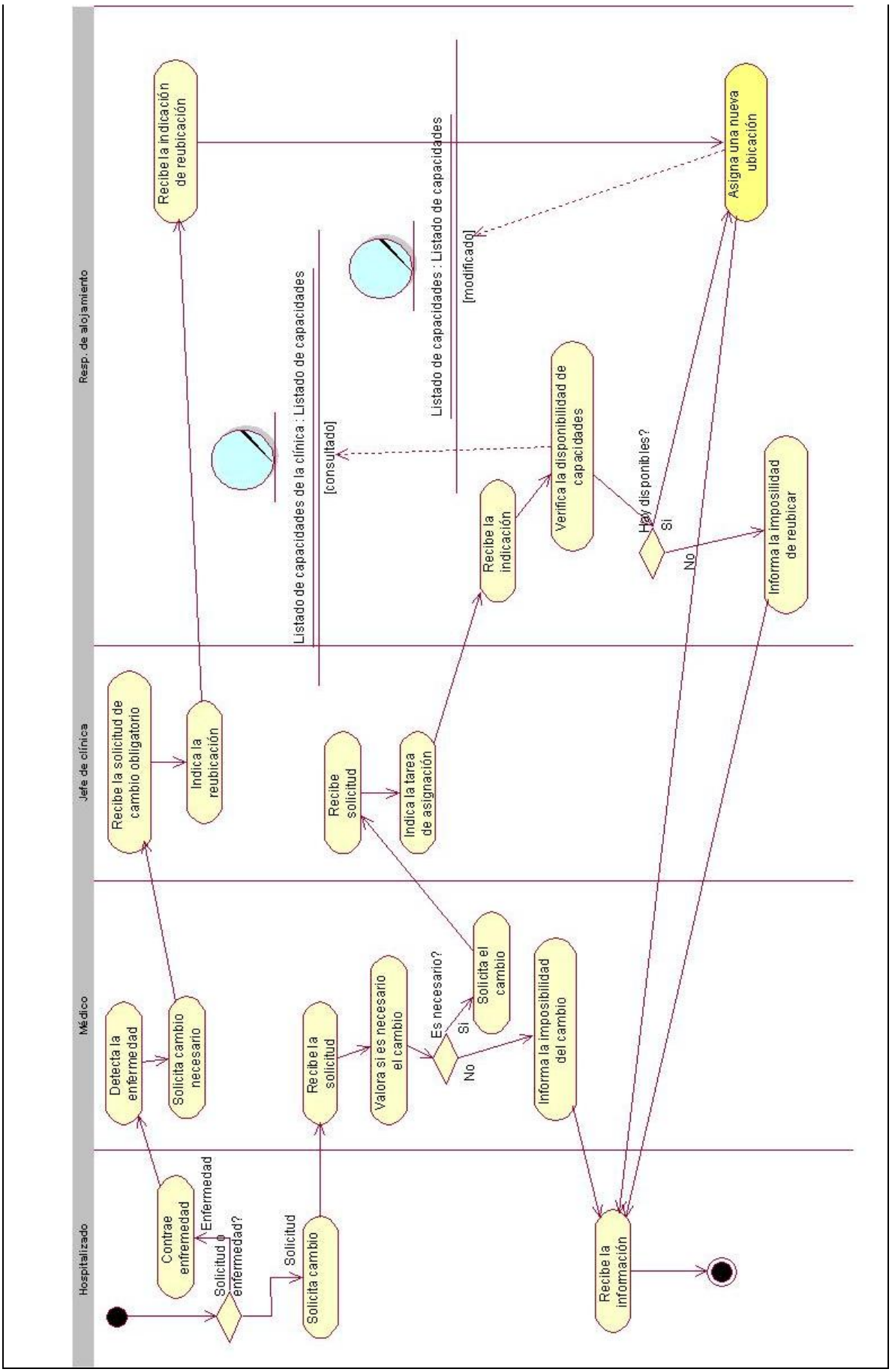

Hospitalizado
Médico
Jefe de clínica
Resp. de alojamiento
Contrae enfermedad
Enfermedad
Solicitud o enfermedad?
Solicitud
Solicita cambio
Detecta la enfermedad
Solicita cambio necesario
Recibe la solicitud de cambio obligatorio
Indica la reubicación
Recibe la indicación de reubicación
Recibe la solicitud
Valora si es necesario el cambio
Es necesario?
Si
No
Solicita el cambio
Recibe solicitud
Indica la tarea de asignación
Informa la imposibilidad del cambio
Recibe la información
Listado de capacidades de la clínica : Listado de capacidades
[consultado]
Recibe la indicación
Verifica la disponibilidad de capacidades
Listado de capacidades : Listado de capacidades
[modificado]
Hay disponibles?
Si
No
Asigna una nueva ubicación
Informa la imposilidad de reubicar

Figura 2.4 Diagrama de actividades del caso de uso del negocio “Reubicar hospitalizado”.

2.6.2.2 Diagrama de clases del modelo de objeto.

A continuación se muestra en la figura 2.5 el diagrama de clases del modelo de objetos del caso de uso del negocio “Reubicar hospitalizado”.

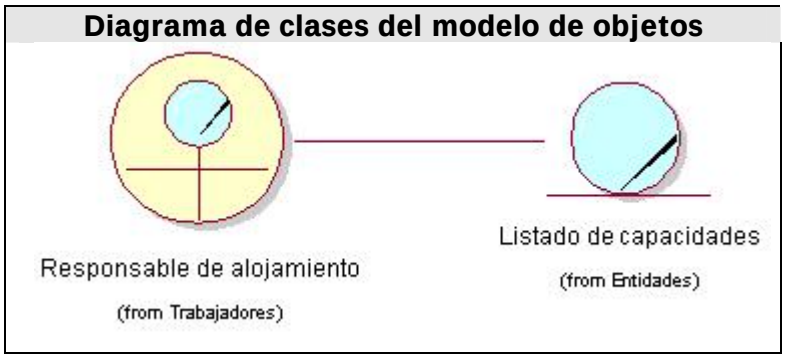


Figura 2.5 Diagrama de clases del modelo de objetos del caso de uso del negocio “Reubicar hospitalizado”.

2.6.3 Caso de uso “Liberar capacidades de hospitalizados”.

Especificación textual en formato general

Caso de uso del negocio: Liberar capacidades de hospitalizados	
Actores del negocio: Hospitalizado.	
Propósito: Permitir que cuando un hospitalizado deje la Universidad, se libere la capacidad que estaba siendo ocupada por el mismo para luego otorgarla a otro hospitalizado.	
Resumen: El caso de uso se inicia cuando el hospitalizado regresa a su país o se traslada hacia otro centro. En ambos casos el médico es el encargado de dar el alta médica y actualizar el listado de capacidades, finalizando así el caso de uso.	
Curso normal de los eventos:	
Acción del actor	**Respuesta del proceso de negocio**
1. El hospitalizado regresa a su país de origen.	1.1. El puesto de mando de vuelo indica la salida del vuelo. 1.2. El puesto de mando de vuelo

	actualiza el listado de las capacidades, finalizando así el caso de uso.
Flujo alterno de los eventos:	
Acción del actor:	**Respuesta del proceso de negocio:**
1. El hospitalizado se traslada de la universidad hacia otro centro hospitalario.	1.1. El médico indica el traslado del paciente. 1.2. Actualiza el listado de capacidades terminando así el caso de uso
Prioridad: Alta	
Mejoras: El sistema garantizará que una vez que se reconozca la salida hospitalizado del centro, de forma automática y confiable se libere la capacidad que ocupaba.	
Otras secciones:	

Tabla 2.5 Especificación textual del caso de uso del negocio “Liberar capacidades de hospitalizados”.

2.6.3.1 Diagrama de actividades.

A continuación se muestra en la figura 2.6 el diagrama de actividades del caso de uso del negocio “Liberar capacidades de hospitalizados”.

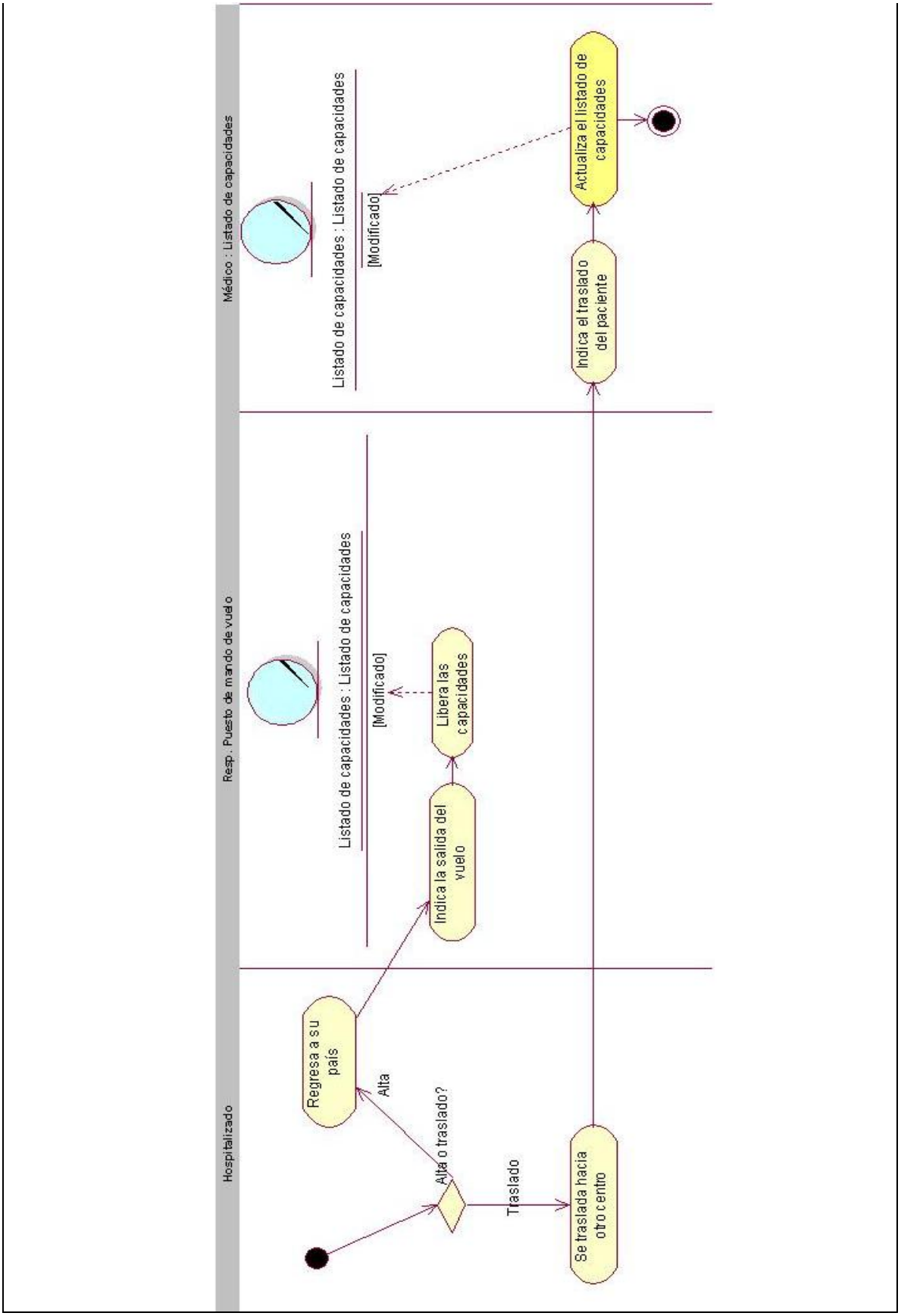

Hospitalizado
Resp. Puesto de mando de vuelo
Médico : Listado de capacidades
Regresa a su país
Alta
Alta o traslado?
Traslado
Indica la salida del vuelo
Libera las capacidades
Listado de capacidades : Listado de capacidades
[Modificado]
Se traslada hacia otro centro
Indica el traslado del paciente
Actualiza el listado de capacidades
Listado de capacidades : Listado de capacidades
[Modificado]

Figura 2.6 Diagrama de actividades del caso de uso del negocio “Liberar capacidades de hospitalizados”.

2.6.3.2 Diagrama de clases del modelo de objeto.

A continuación se muestra en la figura 2.7 el diagrama de clases del modelo de objetos del caso de uso del negocio “Liberar capacidades de hospitalizados”.

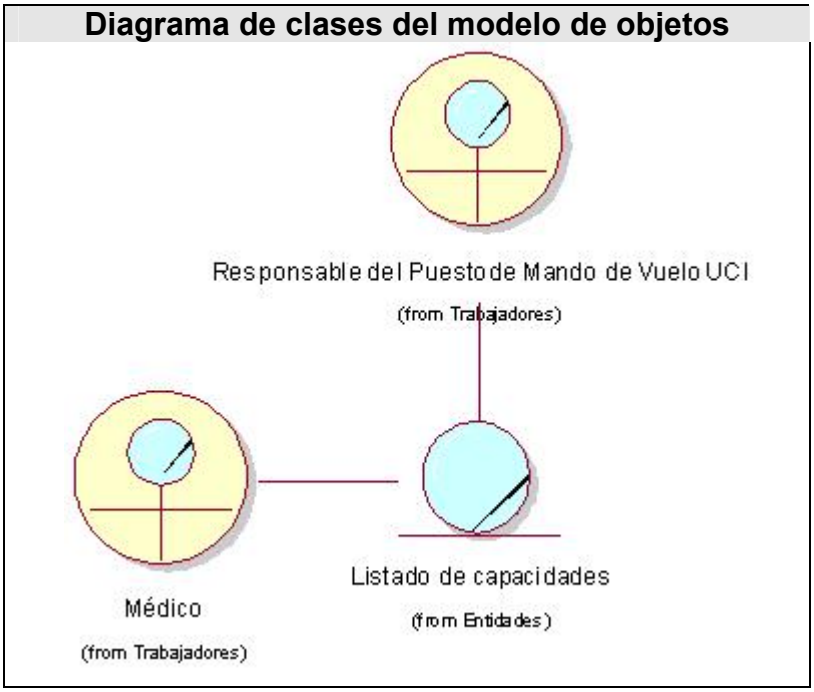


Figura 2.7 Diagrama de clases del modelo de objetos del caso de uso del negocio “Liberar capacidades de hospitalizados”.

2.6.4 Caso de uso “Ubicar personal de servicio”.

Especificación textual en formato general

Caso de uso del negocio: Ubicar personal de servicio
Actores del negocio: Trabajador de servicio.
Propósito: Permitir que el trabajador pueda ubicarse una vez presente en la Universidad en caso de tener necesidad de alojamiento.
Resumen: El caso de uso se inicia cuando el trabajador llega a la universidad y solicita ubicación, es atendido por el responsable de alojamiento que es el

encargado de otorgarle una ubicación.	
Curso normal de los eventos:	
Acción del actor	**Respuesta del proceso de negocio**
1. El trabajador arriba a la universidad y solicita ubicación.	1.1. El responsable de alojamiento recibe la solicitud. 1.2. Asigna ubicación al trabajador. 1.3. Actualiza el listado de las capacidades. 1.4. Informa al trabajador de servicio de la ubicación asignada.
2. Recibe la ubicación y termina el caso de uso	
Mejoras: El sistema podrá registrar la ubicación exacta de cada trabajador asociado a la misión, esto permite mantener un mejor control de la ubicación de dichos trabajadores.	
Otras secciones:	

Tabla 2.6 Especificación textual del caso de uso del negocio “Ubicar trabajador de servicio”.

2.6.4.1 Diagrama de actividades.

A continuación se muestra en la figura 2.8 el diagrama de actividades del caso de uso del negocio “Ubicar trabajador de servicio”.

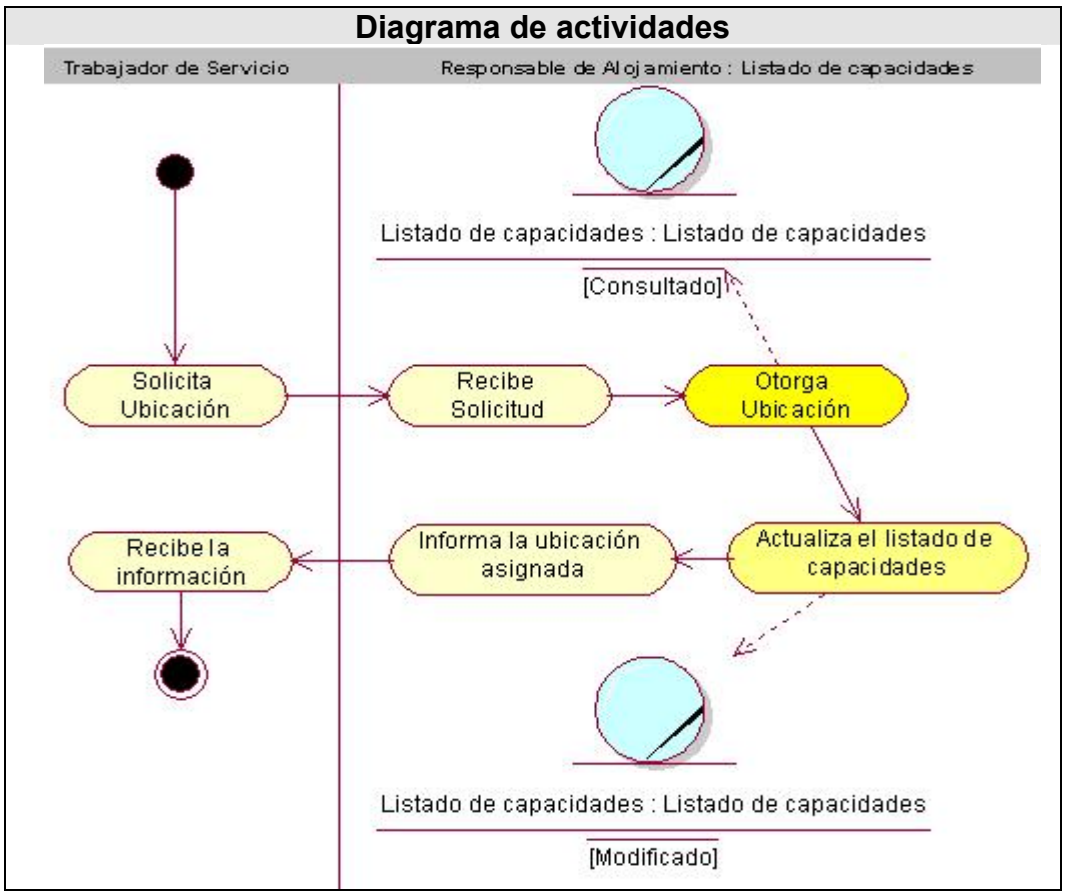


Figura 2.8 Diagrama de actividades del caso de uso del negocio “Ubicar trabajador de servicio”.

2.6.4.2 Diagrama de clases del modelo de objeto.

A continuación se muestra en la figura 2.9 el diagrama de clases del modelo de objetos del caso de uso del negocio “Ubicar trabajador de servicio”.

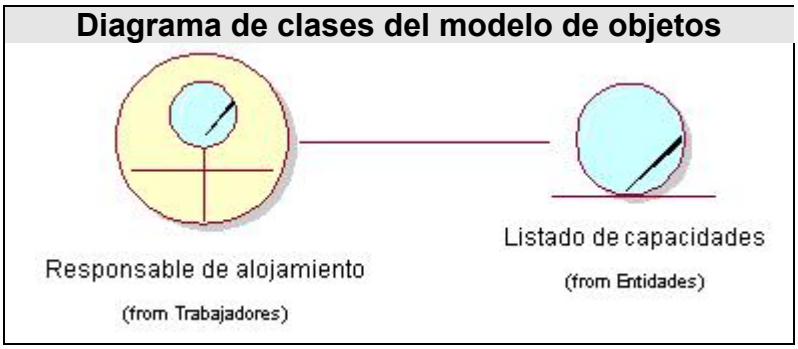


Figura 2.9 Diagrama de clases del modelo de objetos del caso de uso del negocio “Ubicar trabajador de servicio”.

2.6.5 Caso de uso "Reubicar trabajador de servicio".

Especificación textual en formato general

<table>
<tr><td colspan="2">Caso de uso del negocio: Reubicar trabajador de servicio</td></tr>
<tr><td colspan="2">Actores del negocio: Trabajador de servicio.</td></tr>
<tr><td colspan="2">Propósito: Permitir que un trabajador de servicio que no pueda residir en su lugar de alojamiento original por cualquier tipo problemas ya sea de salud u otro caso, pueda cambiar su ubicación.</td></tr>
<tr><td colspan="2">Resumen: El caso de uso se inicia cuando el trabajador de servicio presenta al responsable de alojamiento la situación por la cuál no puede residir en la ubicación asignada. El responsable de alojamiento hace una valoración de la situación. En caso de que decida cambiarlo procede a efectuar el cambio terminando así el caso de uso.</td></tr>
<tr><td colspan="2">Curso normal de los eventos:</td></tr>
<tr><td>Acción del actor</td><td>Respuesta del proceso de negocio</td></tr>
<tr><td>1. El trabajador de servicio solicita el cambio de ubicación.</td><td>1.1. El responsable de alojamiento analiza la situación.
1.2. Efectúa el cambio.
1.3. Actualiza el listado de las capacidades.
1.4. Informa al trabajador de servicio.</td></tr>
<tr><td>2. Recibe la información.</td><td></td></tr>
<tr><td colspan="2">Flujo alterno de los eventos:</td></tr>
<tr><td>Acción del actor:</td><td>Respuesta del proceso de negocio:</td></tr>
<tr><td></td><td>1.2. Informa la imposibilidad de cambio.</td></tr>
<tr><td>2. Recibe la información.</td><td></td></tr>
</table>

Prioridad: Media
Mejoras: El nuevo sistema permite registrar los cambios de ubicación que se efectúen, manteniendo el control sobre las capacidades.

Tabla 2.7 Especificación textual del caso de uso del negocio “Reubicar trabajador de servicio”.

2.6.5.1 Diagrama de actividades.

A continuación se muestra en la figura 2.10 el diagrama de actividades del caso de uso del negocio “Reubicar trabajador de servicio”.

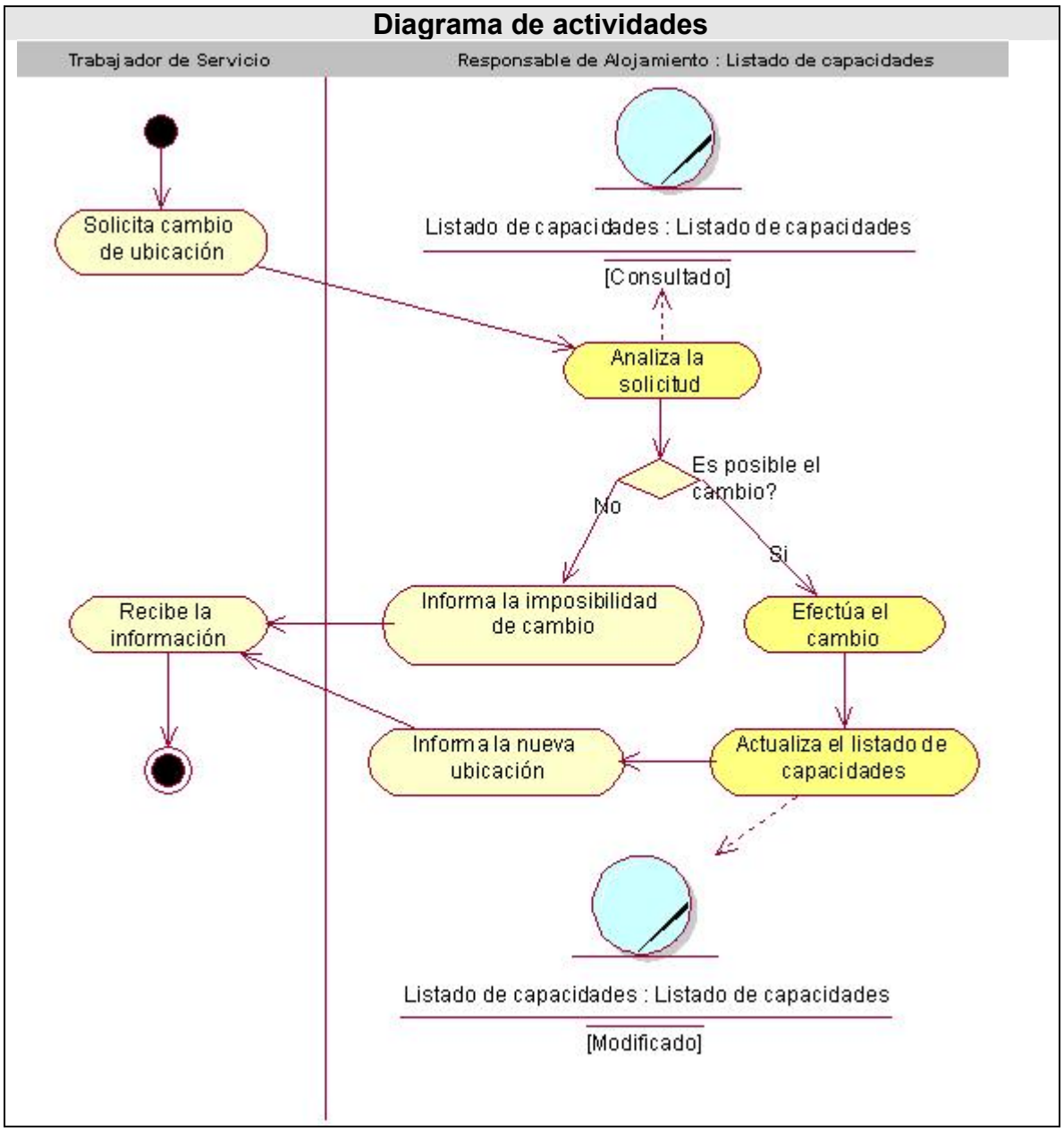


Figura 2.10 Diagrama de actividades del caso de uso del negocio “Reubicar trabajador de servicio”.

2.6.5.2 Diagrama de clases del modelo de objeto.

A continuación se muestra en la figura 2.11 el diagrama de clases del modelo de objetos del caso de uso del negocio “Reubicar trabajador de servicio”.

Diagrama de clases del modelo de objetos

Figura 2.11 Diagrama de clases del modelo de objetos del caso de uso del negocio "Reubicar trabajador de servicio".

2.6.6 Caso de uso "Liberar capacidad de personal de servicio".

Especificación textual en formato general

Caso de uso del negocio: Liberar capacidad de trabajador de servicio	
Actores del negocio: Trabajador de servicio.	
Propósito: Permitir que cuando el trabajador de servicio deje la Universidad, se libere la capacidad que estaba siendo ocupada por el mismo para luego otorgarla a otro trabajador.	
Resumen: El caso de uso se inicia cuando el trabajador concluye su período de trabajo en la misión. El responsable de alojamiento verifica no existe ningún problema da la autorización y actualiza el listado de capacidades. Si hay algún problema lo analiza y luego entonces se determina si el trabajador puede irse o no.	
Curso normal de los eventos:	
Acción del actor	**Respuesta del proceso de negocio**
1. El trabajador de servicio concluye su etapa de trabajo en la misión.	1.1. El responsable de alojamiento verifica que no existe ningún problema de responsabilidad material donde estuvo alojado el trabajador y da el autorizo.
2. Recibe el autorizo.	2.1. Actualiza el listado de las capacidades.
Flujo alterno de los eventos:	

Acción del actor:	**Respuesta del proceso de negocio:**
	1.2. El responsable de alojamiento procede a resolver el problema detectado. 1.3. Informa la decisión tomada.
2. Recibe la información.	2.1. Informa al trabajador de servicio que puede marcharse.
3. Recibe el autorizo.	3.1. Actualiza el listado de capacidades.
Prioridad: Alta	
Mejoras: Con el sistema se puede determinar con exactitud el listado de las capacidades disponibles.	
Otras secciones:	

Tabla 2.8 Especificación textual del caso de uso del negocio “Liberar capacidad de trabajador de servicio”.

2.6.6.1 Diagrama de actividades.

A continuación se muestra en la figura 2.12 el diagrama de actividades del caso de uso del negocio “Liberar capacidad de trabajador de servicio”.

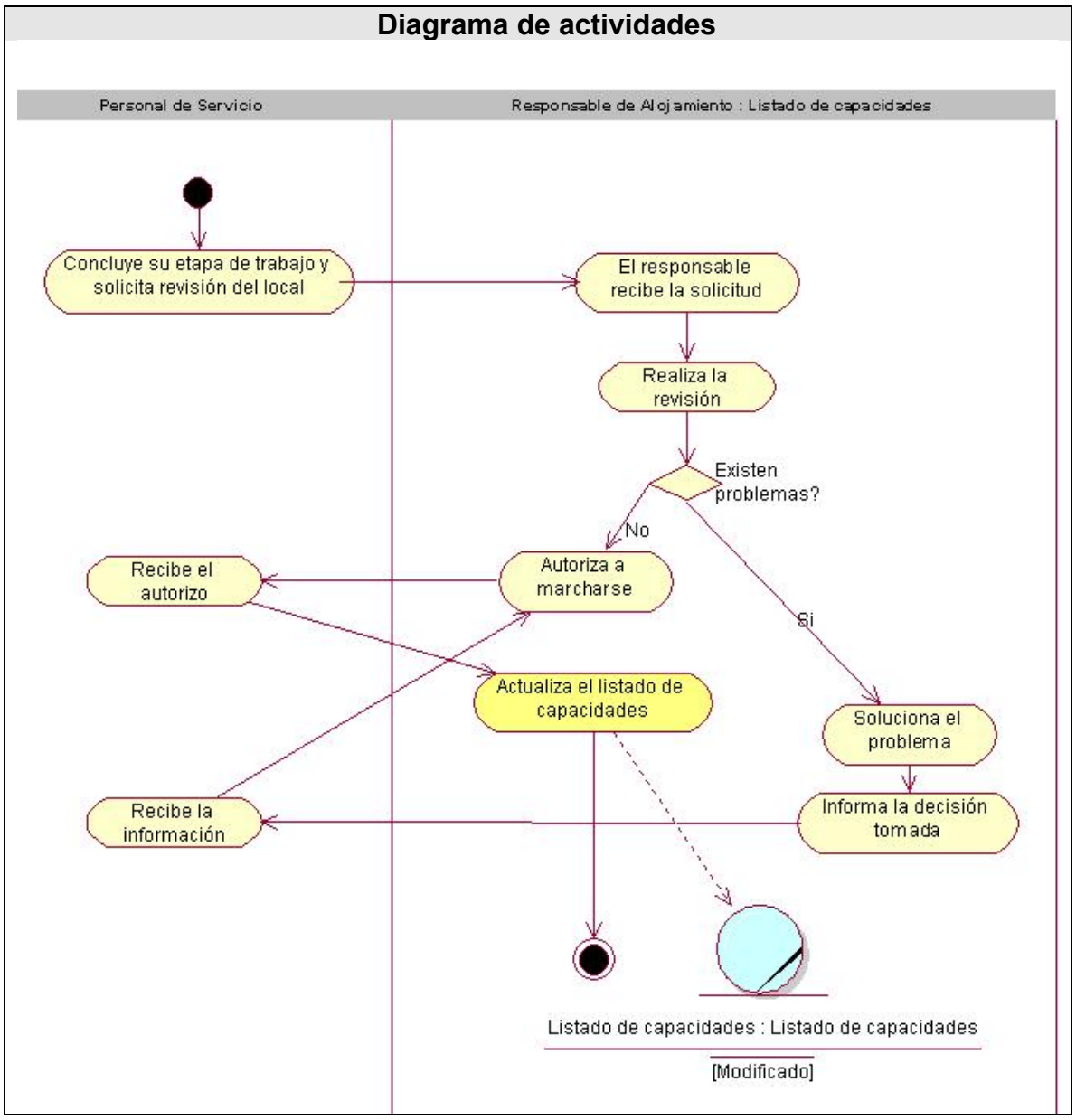


Figura 2.12 Diagrama de actividades del caso de uso del negocio “Liberar capacidad de trabajador de servicio”.

2.6.5.2 Diagrama de clases del modelo de objeto.

A continuación se muestra en la figura 2.13 el diagrama de clases del modelo de objetos del caso de uso del negocio “Liberar capacidad de trabajador de servicio”.

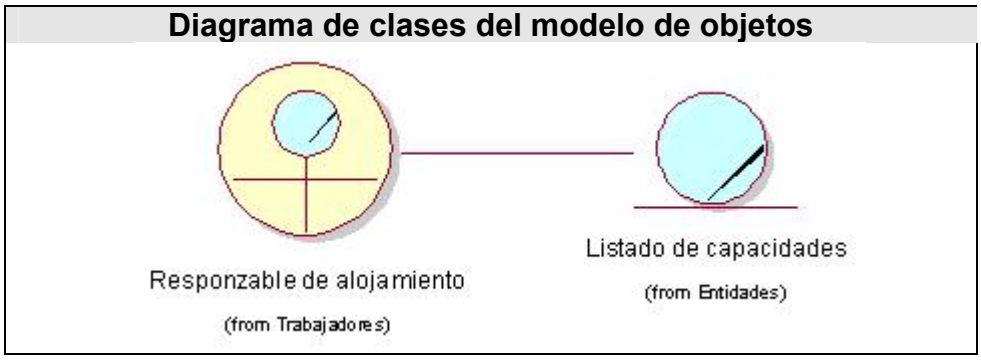


Figura 2.13 Diagrama de clases del modelo de objetos del caso de uso del negocio "Liberar capacidad de trabajador de servicio".

2.7 Conclusiones.

En este capítulo fue descrita la propuesta de negocio donde se detalla como deben ejecutarse los procesos que se llevan a cabo en la gestión de alojamiento en la UCI durante la Misión Milagro; siendo identificados, además, los roles y entidades u objetos del negocio, así como su relación en esos procesos. Esta descripción fue realizada mediante el modelo del negocio, para lo cual se elaboraron los modelos de casos de uso y objetos del negocio.

Tras realizar el modelado del negocio se pudo lograr una mejor comprensión del problema que el sistema tiene que resolver.

CAPÍTULO 3
Requisitos

3.1 Introducción.

En este capítulo se identifican los requisitos funcionales y no funcionales del sistema que dará solución al problema planteado; quiénes interactuarán con él (actores del sistema) y las distintas funcionalidades que ofrecerá a cada uno de los actores.

3.2 Definición de los requisitos funcionales.

Los requerimientos funcionales son aquellos requisitos que, desde el punto de vista de las necesidades del usuario, debe cumplir el sistema y que están fuertemente ligados a las opciones del programa.

Para cumplir con los objetivos propuestos se prevé que el sistema tenga las siguientes funcionalidades:

R1. Administrar los datos de la residencia.

- R1.1. Gestionar datos de las direcciones de residencias.
  - R1.1.1. Listar residencias.
  - R1.1.2. Buscar residencia.
  - R1.1.3. Adicionar dirección de residencia.
  - R1.1.4. Modificar dirección de residencia.
  - R1.1.5. Eliminar dirección de residencia.
- R1.2. Gestionar los datos de las clínicas.
  - R1.2.1. Listar clínicas.
  - R1.2.2. Buscar clínicas.
  - R1.2.3. Adicionar clínica.
  - R1.2.4. Modificar clínica.
  - R1.2.5. Eliminar clínica.
- R1.3. Gestionar los datos de las manzanas.
  - R1.3.1. Listar manzanas.
  - R1.3.2. Buscar manzanas.

R1.3.3. Adicionar manzana.

R1.3.4. Modificar manzana.

R1.3.5. Eliminar manzana.

R1.4. Gestionar los datos de los edificios.

R1.4.1. Listar edificios.

R1.4.2. Buscar edificios.

R1.4.3. Adicionar edificio.

R1.4.4. Modificar edificio.

R1.4.5. Eliminar edificio.

R1.5. Gestionar los datos de los apartamentos.

R1.5.1. Listar apartamentos.

R1.5.2. Buscar apartamentos.

R1.5.3. Adicionar apartamento.

R1.5.4. Modificar apartamento.

R1.5.5. Eliminar apartamento.

R1.6. Gestionar los datos de los cuartos.

R1.6.1. Listar cuartos.

R1.6.2. Buscar cuartos.

R1.6.3. Adicionar cuarto.

R1.6.4. Modificar cuarto.

R1.6.5. Eliminar cuarto.

R1.7. Gestionar los datos de las capacidades.

R1.7.1. Listar capacidades.

R1.7.2. Buscar capacidades.

R1.7.3. Adicionar capacidades.

R1.7.4. Modificar capacidades.

R1.7.5. Eliminar capacidades.

R2. Gestionar ubicación.

R2.1. Gestionar ubicación para estudiante.

R2.1.1. Listar capacidades para estudiantes.

R2.1.2. Ubicar estudiante de forma automática.

R2.1.3. Reubicar estudiante de forma manual.

R2.1.4. Reubicar estudiante de forma asistida.

R2.2. Gestionar ubicación para trabajador.

R2.2.1. Listar capacidades para trabajadores.

R2.2.2. Ubicar trabajador de forma manual.

R2.2.3. Ubicar trabajador de forma asistida.

R2.2.4. Reubicar trabajador de forma manual.

R2.2.5. Reubicar trabajador de forma asistida.

R2.3. Gestionar ubicación para hospitalizado.

R2.3.1. Listar capacidades para hospitalizados.

R2.3.2. Ubicar hospitalizado de forma manual.

R2.3.3. Ubicar hospitalizado de forma asistida.

R2.3.4. Reubicar hospitalizado de forma manual.

R2.3.5. Reubicar hospitalizado de forma asistida.

R2.4. Liberar capacidades.

3.3 Definición de los requisitos no funcionales.

Los requerimientos no funcionales son características que describen alguna forma o restricción para la realización de algún requerimiento (funcionalidad) o conjunto de ellas e inclusive todos los requerimientos. Se consideran los atributos del sistema, propiedades que debe tener el producto.

A continuación se muestran los requerimientos no funcionales:

- **Apariencia o interfaz externa**

La interfaz no contiene muchas imágenes para no demorar las respuestas al usuario. El diseño de la interfaz es sencillo y claro de usar con reconocimiento visual a través de elementos visibles que identifiquen cada una de sus acciones. Es formal, serio y con una navegación sugerente, todo esto teniendo en cuenta el fin con el que se desarrolla la aplicación.

- **Usabilidad**

El sistema puede ser usado por cualquier persona, comprendida en edad laboral de 18 a 60 años, que posea conocimientos básicos en el manejo de la computadora y de un ambiente Web en sentido general. Instalar el sistema trae consigo una mayor rapidez de trabajo y por consiguiente un ahorro de materiales y personal.

- **Rendimiento**

La disponibilidad de trabajo en red contra el servidor es constante.

Se garantiza que la respuesta a solicitudes de los usuarios del sistema sea en un período de tiempo breve (de segundos) para evitar la acumulación de trabajo por parte de los responsables y público en los puntos de admisión. El sistema deberá de ser lo más estable y confiable posible.

- **Soporte**

Se requiere que el producto reciba mantenimiento ante cualquier fallo que ocurra. El sistema es de fácil instalación.

- **Ayuda y documentación en línea**

El sistema brinda a los usuarios una buena ayuda en línea de modo de si el usuario presenta algún problema pueda acudir al mismo, así como una documentación apropiada para el mejor trabajo con el mismo.

- **Software**

Para el funcionamiento del sistema en el servidor es necesario el S.O. Windows 98 o superior, Linux o Unix, en sus versiones de S.O. servidores. Para el funcionamiento del sistema en las terminales cliente es necesario el S.O. Windows 95 o superior, Linux o Unix.

- **Hardware**

Se necesitan como requerimientos mínimos una PC con procesador Pentium II o superior.

- **Portabilidad**

El producto es usado bajo los S.O. Windows, Linux y Unix.

El producto corre sobre una plataforma Web, codificada en “PHP5” y sus sistemas de bases de datos en PostgreSQL.

- **Seguridad**

El sistema se encarga de controlar los diferentes niveles de acceso y funcionalidad de usuarios al sitio, de identificar al usuario antes de que pueda realizar cualquier acción sobre el sistema. Garantiza que la información sea vista únicamente por quien tiene derecho a verla.

Existe un primer nivel o nivel básico donde están las funciones asociadas al usuario general o común, que requieren poca responsabilidad. El segundo nivel esta compuesto por funciones de mayor complejidad y que pueden destruir información relacionada a las entidades del sistema. El tercer nivel esta conformado con las funciones administrativas del sitio y del sistema por tanto es el nivel de mayor complejidad.

Se usan mecanismos de encriptación (Base64) de los datos que por cuestiones de seguridad no deben viajar al servidor en texto claro, como es el caso de las contraseñas.
Se hacen validaciones de la información tanto en el cliente como en el servidor, no obstante los usuarios acceden de manera rápida y operativa al sistema sin que los requerimientos de seguridad se conviertan en un retardo para ellos.
Debido a la importancia que tiene este requerimiento para el sistema de la Misión Milagro se decidió hacer un módulo para manejar todo lo referente a la misma.

- **Confidencialidad**

Toda la información está protegida del acceso no autorizado, los administradores de sistema son los únicos que podrán transformar la información, los operadores solo podrán ver los listados de información.

- **Disponibilidad**

Se garantiza a los usuarios del sistema el acceso a la información solicitada en todo momento (si tiene permiso para ello).

- **Políticos-Culturales**

Toda modificación al funcionamiento establecido en los requerimientos será realizada por la Dirección del Puesto Mando Informatización conjuntamente con el personal de la misión.

- **Restricciones en el diseño y la implementación**

Es una aplicación Web desarrollada con la tecnología para creación de páginas Web dinámicas PHP5 y base de datos en PostgreSQL.

- **Legales**

El sistema se basa en un estándar que se rige por normas internacionales y cumple con las normas y leyes establecidas en nuestro país.

La plataforma escogida para el desarrollo de la aplicación, está basada en la licencia GNU/GPL.

- **Confiabilidad**

La herramienta de implementación a utilizar tiene soporte para recuperación ante fallos y errores.

- **Restricciones**

Se utiliza UML para lograr una mejor documentación del sistema y como herramienta de apoyo Rational Rose. Se utiliza como lenguaje de programación PHP5 y el gestor de base de datos PostgreSQL.

3.4 Actores del sistema a automatizar.

Los actores del sistema pueden representar el rol que juega una o varias personas, un equipo o un sistema automatizado, son parte del sistema, y pueden intercambiar información con él o ser recipientes pasivos de información. En este caso los actores que interactúan con el sistema se definen a continuación en la tabla 3.1.

Actores	Justificación
Responsable de alojamiento	Se encarga de seleccionar y registrar en el sistema la ubicación asignada a la persona. La persona puede ser un trabajador o un hospitalizado.
Administrador	Se encarga de adicionar, modificar o eliminar los datos referentes a la composición de la estructura del alojamiento de la UCI (datos referentes a las manzanas, edificios, apartamentos…).
Agente externo	Es un representante de otro módulo que tras

	realizar una acción propia de su negocio inicia el proceso de liberación de capacidades. Ej. Responsable del puesto de mando de vuelos.

Tabla 3.1 Descripción de los actores del sistema

3.5 Paquetes y sus relaciones.

Un sistema grande se debe dividir en unidades más pequeñas, de modo que pueda ser entendido por las personas que necesiten consultarlo y además para que los equipos de trabajo puedan trabajar de manera independiente.

Para satisfacer los objetivos propuestos al inicio de este trabajo el sistema que se propone debe estar conformado por dos paquetes: Administrar Alojamiento, Gestión de Ubicación. A continuación en la figura 3.1 se representa el diagrama de paquetes y sus relaciones.

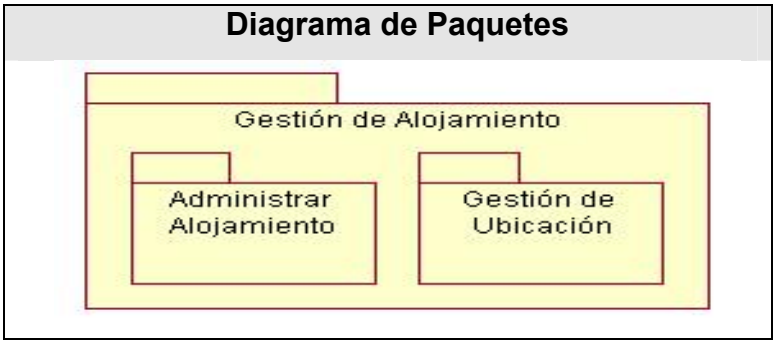


Figura 3.1 Diagrama de paquetes y sus relaciones.

3.6 Paquete “Administrar Alojamiento”

En este paquete se recogen las funcionalidades administrativas que permiten definir la estructuración del alojamiento en tiempo de misión.

3.6.1 Diagrama de casos de uso.

Los Casos de Uso son “fragmentos” de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores. De manera más precisa, un Caso de Uso

especifica una secuencia de acciones que el sistema puede llevar a cabo interactuando con sus actores, incluyendo alternativas dentro de la secuencia. [12]

A continuación se muestra el diagrama de casos de uso del sistema correspondiente al paquete.

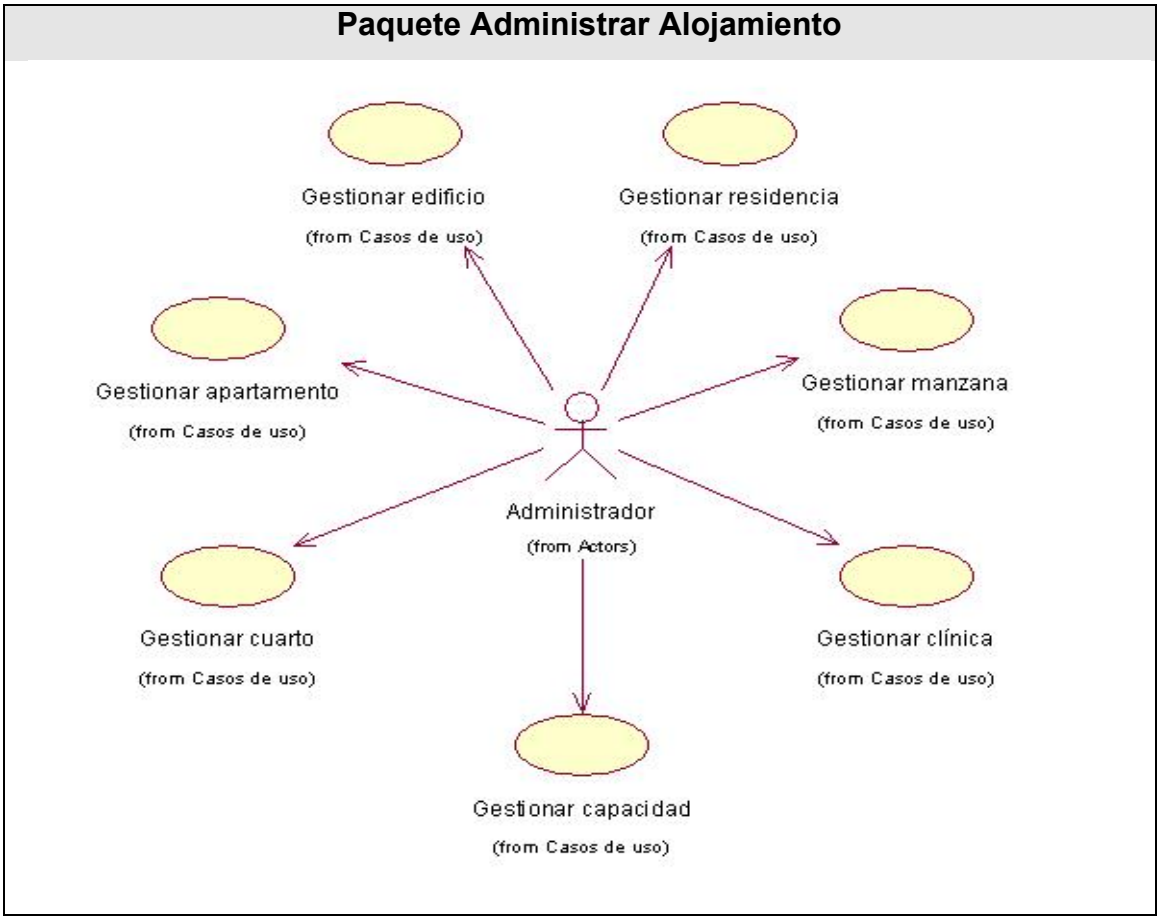


Figura 3.2 Diagrama de casos de uso del sistema del paquete Administrar Alojamiento.

3.6.2 Descripción de los casos de uso.

3.6.2.1 Caso de uso "Gestionar Apartamento".

Nombre del Caso de Uso	**Gestionar Apartamento**
Actores	Administrador
Propósito	Garantizar que los datos de cada apartamento se registren, se modifiquen y se eliminen en el sistema.

Resumen	El caso de uso se inicia cuando el administrador accede al sistema y solicita Gestionar Apartamento, inmediatamente se visualiza una pantalla donde se muestran los apartamentos existentes en el sistema, a partir de aquí el sistema brinda la posibilidad de efectuar las operaciones de agregar, modificar o eliminar uno de ellos. Finaliza una vez que el administrador realiza una de las operaciones.
Referencias	R1.5
Precondiciones	Si se desea agregar un apartamento debe existir la instancia del edificio al que pertenece.
Poscondiciones	Para los siguientes procesos: Insertar: se crea una instancia de la clase apartamento. Modificar: se modifican los atributos de la instancia seleccionada de la clase apartamento. Eliminar: se elimina la instancia seleccionada de la clase apartamento.
Requerimientos especiales	

Curso normal de los eventos	
Acciones del actor	**Respuestas del Sistema**
1. El administrador decide administrar apartamentos y selecciona la opción.	1.1. El sistema muestra una pantalla con el listado de los apartamentos registrados.
2. El administrador selecciona una de las operaciones a realizar.	2.1. El sistema ejecuta una de las siguientes acciones:

	• Si decide agregar un apartamento, ir a la sección “Agregar apartamento”. • Si decide modificar un apartamento, ir a la sección “Modificar apartamento”. • Si decide eliminar un apartamento, ir a la sección “Eliminar apartamento”. • Si decide buscar un apartamento, ir a la sección “Buscar apartamento”.
Sección “Agregar Apartamento”	
3. El administrador introduce los datos del apartamento.	3.1. El sistema verifica que los datos introducidos sean válidos y que los requeridos no estén vacíos. 3.2. El sistema verifica que el apartamento no exista. 3.3. El apartamento se registra en el sistema. 3.4. El sistema muestra un mensaje comunicando que el apartamento se agregó satisfactoriamente, dando la posibilidad de agregar otro apartamento y finaliza así el caso de uso.

Curso alternativo	
	3.2. Se emite un mensaje de error para que se llenen los campos obligatorios o se introduzcan correctamente los datos.
	3.3. Se emite un mensaje informando de la existencia del apartamento.
Sección "Modificar Apartamento"	
3. El administrador selecciona dentro del listado de apartamentos el que desea modificar.	3.1. El sistema busca el apartamento y muestra los datos en una nueva pantalla
4. El administrador realiza las actualizaciones deseadas.	4.1. El sistema verifica que los datos introducidos sean válidos y que los requeridos no estén vacíos. 4.2. El sistema actualiza la información. 4.3. Muestra el listado de los apartamentos y finaliza el caso de uso.
Curso alternativo	
	4.1. Se emite un mensaje de error para que se llenen los campos obligatorios o se introduzcan correctamente los datos.

Sección “Eliminar Apartamento”	
3. El administrador selecciona dentro del listado de apartamentos el que desea eliminar.	3.1. El sistema muestra un mensaje de confirmación. 3.2. El sistema elimina el apartamento y finaliza el caso de uso.
Sección “Buscar Apartamento”	
3. El administrador desea ver los datos de un apartamento específico y selecciona su número.	3.1. El sistema busca el apartamento y muestra sus datos en el listado.
Prototipo	Ver Anexo 8.

Tabla 3.2 Descripción del caso de uso “Gestionar Apartamento”.

3.6.2.2 Caso de uso “Gestionar Residencia”.

Nombre del Caso de Uso	**Gestionar Residencia**
Actores	Administrador
Propósito	Garantizar que los datos de cada residencia se registren, se modifiquen y se eliminen en el sistema.
Resumen	El caso de uso se inicia cuando el administrador accede al sistema y solicita Gestionar Residencia, inmediatamente se visualiza una pantalla donde se muestran las residencias existentes en el sistema, a partir de aquí el sistema brinda la posibilidad de efectuar las operaciones de agregar, modificar o eliminar una de ellas. Finaliza una vez que el administrador realiza una de las operaciones.

Referencias	R1.1
Precondiciones	
Poscondiciones	Para los siguientes procesos: Insertar: se crea una instancia de la clase residencia. Modificar: se modifican los atributos de la instancia seleccionada de la clase residencia. Eliminar: se elimina la instancia seleccionada de la clase residencia.

Curso normal de los eventos	
Acciones del actor	**Respuestas del Sistema**
1. El administrador decide administrar residencias y selecciona la opción.	1.1. El sistema muestra una pantalla con el listado de las residencias registradas.
2. El administrador selecciona una de las operaciones a realizar.	2.1. El sistema ejecuta una de las siguientes acciones: • Si decide agregar una residencia, ir a la sección “Agregar residencia”. • Si decide modificar una residencia, ir a la sección “Modificar residencia”. • Si decide eliminar una residencia, ir a la sección “Eliminar residencia”. • Si decide buscar una residencia, ir a la sección “Buscar

	residencia".
Sección "Agregar Residencia"	
3. El administrador introduce los datos de la residencia.	3.1. El sistema verifica que los datos introducidos sean válidos y que los requeridos no estén vacíos. 3.2. El sistema verifica que la residencia no exista. 3.3. La residencia se registra en el sistema. 3.4. El sistema muestra un mensaje comunicando que la residencia se agregó satisfactoriamente, dando la posibilidad de agregar otra residencia y finaliza así el caso de uso.
Curso alternativo	
	3.2. Se emite un mensaje de error para que se llenen los campos obligatorios o se introduzcan correctamente los datos.
	3.3. Se emite un mensaje informando de la existencia de la residencia.
Sección "Modificar Residencia"	
3. El administrador selecciona dentro	3.1. El sistema busca la residencia

del listado de residencias la que desea modificar.	y muestra los datos en una nueva pantalla.
4. El administrador realiza las actualizaciones deseadas.	4.1. El sistema verifica que los datos introducidos sean válidos y que los requeridos no estén vacíos. 4.2. El sistema actualiza la información. 4.3. Muestra el listado de las residencias y finaliza el caso de uso.
Curso alternativo	
	4.2. Se emite un mensaje de error para que se llenen los campos obligatorios o se introduzcan correctamente los datos.
Sección "Eliminar Residencia"	
3. El administrador selecciona dentro del listado de residencias la que desea eliminar.	3.1. El sistema muestra un mensaje de confirmación. 3.2. El sistema elimina la residencia y finaliza el caso de uso.
Sección "Buscar Residencia"	
3. El administrador desea ver los datos de una residencia específica y selecciona su número.	3.1. El sistema busca la residencia y muestra sus datos en el listado.

Prototipo	Ver Anexo 9.

Tabla 3.3 Descripción del caso de uso “Gestionar Residencia”.

3.5.2.3 Caso de uso “Gestionar Capacidad”.

Nombre del Caso de Uso	**Gestionar Capacidad**
Actores	Administrador
Propósito	Garantizar que los datos de cada capacidad se registren, se modifiquen y se eliminen en el sistema.
Resumen	El caso de uso se inicia cuando el administrador accede al sistema y solicita Gestionar Capacidad, inmediatamente se visualiza una pantalla donde se muestran las capacidades existentes en el sistema, a partir de aquí el sistema brinda la posibilidad de efectuar las operaciones de agregar, modificar o eliminar una de ellas. Finaliza una vez que el administrador realiza una de las operaciones.
Referencias	R1.7
Precondiciones	Si se desea agregar una capacidad debe existir la instancia del cuarto al que pertenece.
Poscondiciones	Para los siguientes procesos: Insertar: se crea una instancia de la clase capacidad. Modificar: se modifican los atributos de la instancia seleccionada de la clase capacidad. Eliminar: se elimina la instancia seleccionada de la clase capacidad.
Requerimientos especiales	

Curso normal de los eventos	
Acciones del actor	**Respuestas del Sistema**
1. El administrador decide administrar capacidades y selecciona la opción.	1.1. El sistema muestra una pantalla con el listado de las capacidades registradas.
2. El administrador selecciona una de las operaciones a realizar.	2.1. El sistema ejecuta una de las siguientes acciones: • Si decide agregar una capacidad, ir a la sección “Agregar capacidad”. • Si decide modificar una capacidad, ir a la sección “Modificar capacidad”. • Si decide eliminar una capacidad, ir a la sección “Eliminar capacidad”. • Si decide buscar una capacidad, ir a la sección “Buscar capacidad”.
Sección “Agregar Capacidad”	
3. El administrador introduce los datos de la capacidad.	3.1. El sistema verifica que los datos introducidos sean válidos y que los requeridos no estén vacíos. 3.2. El sistema verifica que la capacidad no exista.

	3.3. La capacidad se registra en el sistema. 3.4. El sistema muestra un mensaje comunicando que la capacidad se agregó satisfactoriamente, dando la posibilidad de agregar otra capacidad y finaliza así el caso de uso.
Curso alternativo	
	3.2. Se emite un mensaje de error para que se llenen los campos obligatorios o se introduzcan correctamente los datos.
	3.3. Se emite un mensaje informando de la existencia de la capacidad.
Sección "Modificar Capacidad"	
3. El administrador selecciona dentro del listado de capacidades la que desea modificar.	3.1. El sistema busca la capacidad y muestra los datos en una nueva pantalla.
4. El administrador realiza las actualizaciones deseadas.	4.1. El sistema verifica que los datos introducidos sean válidos y que los requeridos no estén vacíos. 4.2. El sistema actualiza la información. 4.3. Muestra el listado de las

	capacidades y finaliza el caso de uso.
Curso alternativo	
	4.2. Se emite un mensaje de error para que se llenen los campos obligatorios o se introduzcan correctamente los datos.
Sección "Eliminar Capacidad"	
3. El administrador selecciona dentro del listado de capacidades la que desea eliminar.	3.1. El sistema muestra un mensaje de confirmación. 3.2. El sistema elimina la capacidad y finaliza el caso de uso.
Sección "Buscar Capacidad"	
4. El administrador desea ver los datos de una capacidad específica y selecciona su número.	4.1. El sistema busca la capacidad y muestra sus datos en el listado.
Prototipo	Ver Anexo 10.

Tabla 3.4 Descripción del caso de uso "Gestionar Capacidad".

3.6.2.4 Caso de uso "Gestionar Clínica".

Nombre del Caso de Uso	**Gestionar Clínica**
Actores	Administrador
Propósito	Garantizar que los datos de cada clínica se registren, se modifiquen y se eliminen en el sistema.

Resumen	El caso de uso se inicia cuando el administrador accede al sistema y solicita Gestionar Clínica, inmediatamente se visualiza una pantalla donde se muestran las clínicas existentes en el sistema, a partir de aquí el sistema brinda la posibilidad de efectuar las operaciones de agregar, modificar o eliminar una de ellas. Finaliza una vez que el administrador realiza una de las operaciones.
Referencias	R1.2
Precondiciones	
Poscondiciones	Para los siguientes procesos: Insertar: se crea una instancia de la clase clínica. Modificar: se modifican los atributos de la instancia seleccionada de la clase clínica. Eliminar: se elimina la instancia seleccionada de la clase clínica.

Curso normal de los eventos	
Acciones del actor	**Respuestas del Sistema**
1. El administrador decide administrar clínicas y selecciona la opción.	1.1. El sistema muestra una pantalla con el listado de las clínicas registradas.
2. El administrador selecciona una de las operaciones a realizar.	2.1. El sistema ejecuta una de las siguientes acciones: • Si decide agregar una clínica, ir a la sección “Agregar clínica”. • Si decide modificar una clínica, ir a la sección “Modificar clínica”. • Si decide eliminar una clínica, ir a

	la sección “Eliminar clínica”. • Si decide buscar una clínica, ir a la sección “Buscar clínica”.
Sección “Agregar Clínica”	
3. El administrador introduce los datos de la clínica.	3.1. El sistema verifica que los datos introducidos sean válidos y que los requeridos no estén vacíos. 3.2. El sistema verifica que la clínica no exista. 3.3. La clínica se registra en el sistema. 3.4. El sistema muestra un mensaje comunicando que la clínica se agregó satisfactoriamente, dando la posibilidad de agregar otra clínica y finaliza así el caso de uso.
Curso alternativo	
	3.2. Se emite un mensaje de error para que se llenen los campos obligatorios o se introduzcan correctamente los datos.
	3.3. Se emite un mensaje informando de la existencia de la clínica.

Sección “Modificar Clínica”	
3. El administrador selecciona dentro del listado de clínicas la que desea modificar.	3.1. El sistema busca la clínica y muestra los datos en una nueva pantalla
4. El administrador realiza las actualizaciones deseadas.	4.1. El sistema verifica que los datos introducidos sean válidos y que los requeridos no estén vacíos. 4.2. El sistema actualiza la información. 4.3. Muestra el listado de las clínicas y finaliza el caso de uso.
Curso alternativo	
	4.2. Se emite un mensaje de error para que se llenen los campos obligatorios o se introduzcan correctamente los datos.
Sección “Eliminar Clínica”	
3. El administrador selecciona dentro del listado de clínicas la que desea eliminar.	3.1. El sistema muestra un mensaje de confirmación. 3.2. El sistema elimina la clínica y finaliza el caso de uso.
Sección “Buscar Clínica”	
3. El administrador desea ver los datos	3.1. El sistema busca la clínica y

de una clínica específica y selecciona su nombre.	muestra sus datos en el listado.
Prototipo	Ver Anexo 11.

Tabla 3.5 Descripción del caso de uso “Gestionar Clínica”.

3.6.2.5 Caso de uso “Gestionar Cuarto”.

Nombre del Caso de Uso	**Gestionar Cuarto**
Actores	Administrador
Propósito	Garantizar que los datos de cada cuarto se registren, se modifiquen y se eliminen en el sistema.
Resumen	El caso de uso se inicia cuando el administrador accede al sistema y solicita Gestionar Cuarto, inmediatamente se visualiza una pantalla donde se muestran los cuartos existentes en el sistema, a partir de aquí el sistema brinda la posibilidad de efectuar las operaciones de agregar, modificar o eliminar uno de ellos. Finaliza una vez que el administrador realiza una de las operaciones.
Referencias	R1.6
Precondiciones	Si se desea agregar un cuarto debe existir la instancia del apartamento al que pertenece.
Poscondiciones	Para los siguientes procesos: Insertar: se crea una instancia de la clase cuarto. Modificar: se modifican los atributos de la instancia seleccionada de la clase cuarto. Eliminar: se elimina la instancia seleccionada de la clase cuarto.
Curso normal de los eventos	

Acciones del actor	**Respuestas del Sistema**
1. El administrador decide administrar cuartos y selecciona la opción.	1.1. El sistema muestra una pantalla con el listado de los cuartos registrados.
2. El administrador selecciona una de las operaciones a realizar.	2.1. El sistema ejecuta una de las siguientes acciones: • Si decide agregar un cuarto, ir a la sección “Agregar cuarto”. • Si decide modificar un cuarto, ir a la sección “Modificar cuarto”. • Si decide eliminar un cuarto, ir a la sección “Eliminar cuarto”. • Si decide buscar un cuarto, ir a la sección “Buscar cuarto”.
Sección “Agregar Cuarto”	
3. El administrador introduce los datos del cuarto.	3.1. El sistema verifica que los datos introducidos sean válidos y que los requeridos no estén vacíos. 3.2. El sistema verifica que el cuarto no exista. 3.3. El cuarto se registra en el sistema. 3.4. El sistema muestra un mensaje comunicando que el cuarto se agregó satisfactoriamente,

	dando la posibilidad de agregar otro cuarto y finaliza así el caso de uso.
Curso alternativo	
	3.2. Se emite un mensaje de error para que se llenen los campos obligatorios o se introduzcan correctamente los datos.
	3.3. Se emite un mensaje informando de la existencia del cuarto.
Sección "Modificar Cuarto"	
3. El administrador selecciona dentro del listado de cuartos el que desea modificar.	3.1. El sistema busca el cuarto y muestra los datos en una nueva pantalla
4. El administrador realiza las actualizaciones deseadas.	4.1. El sistema verifica que los datos introducidos sean válidos y que los requeridos no estén vacíos. 4.2. El sistema actualiza la información. 4.3. Muestra el listado de los cuartos y finaliza el caso de uso.
Curso alternativo	

	4.2. Se emite un mensaje de error para que se llenen los campos obligatorios o se introduzcan correctamente los datos.
Sección "Eliminar Cuarto"	
3. El administrador selecciona dentro del listado de cuartos el que desea eliminar.	3.1. El sistema muestra un mensaje de confirmación. 3.2. El sistema elimina el cuarto y finaliza el caso de uso.
Sección "Buscar Cuarto"	
3. El administrador desea ver los datos de un cuarto específico y selecciona su número.	3.1. El sistema busca el cuarto y muestra sus datos en el listado.
Prototipo	Ver Anexo 12.

Tabla 3.6 Descripción del caso de uso "Gestionar Cuarto".

3.6.2.6 Caso de uso "Gestionar Edificio".

Nombre del Caso de Uso	**Gestionar Edificio**
Actores	Administrador
Propósito	Garantizar que los datos de cada edificio se registren, se modifiquen y se eliminen en el sistema.
Resumen	El caso de uso se inicia cuando el administrador accede al sistema y solicita Gestionar Edificio, inmediatamente se visualiza una pantalla donde se muestran los edificios existentes en el sistema, a partir de aquí el sistema

	brinda la posibilidad de efectuar las operaciones de agregar, modificar o eliminar uno de ellos. Finaliza una vez que el administrador realiza una de las operaciones.
Referencias	R1.4
Precondiciones	Si se desea agregar un edificio debe existir la instancia de la manzana a la que pertenece, así como la dirección de residencia a la que pertenece.
Poscondiciones	Para los siguientes procesos: Insertar: se crea una instancia de la clase edificio. Modificar: se modifican los atributos de la instancia seleccionada de la clase edificio. Eliminar: se elimina la instancia seleccionada de la clase edificio.

Curso normal de los eventos	
Acciones del actor	**Respuestas del Sistema**
1. El administrador decide administrar edificios y selecciona la opción.	1.1. El sistema muestra una pantalla con el listado de los edificios registrados.
2. El administrador selecciona una de las operaciones a realizar.	2.1. El sistema ejecuta una de las siguientes acciones: • Si decide agregar un edificio, ir a la sección “Agregar edificio”. • Si decide modificar un edificio, ir a la sección “Modificar edificio”. • Si decide eliminar un edificio, ir a la sección “Eliminar edificio”.

	• Si decide buscar un edificio, ir a la sección “Buscar edificio”.
Sección “Agregar Edificio”	
3. El administrador introduce los datos del edificio.	3.1. El sistema verifica que los datos introducidos sean válidos y que los requeridos no estén vacíos. 3.2. El sistema verifica que el edificio no exista. 3.3. El edificio se registra en el sistema. 3.4. El sistema muestra un mensaje comunicando que el edificio se agregó satisfactoriamente, dando la posibilidad de agregar otro edificio y finaliza así el caso de uso.
Curso alternativo	
	3.2. Se emite un mensaje de error para que se llenen los campos obligatorios o se introduzcan correctamente los datos.
	3.3. Se emite un mensaje informando de la existencia del edificio.
Sección “Modificar Edificio”	

3. El administrador selecciona dentro del listado de edificios el que desea modificar.	3.1. El sistema busca el edificio y muestra los datos en una nueva pantalla.
4. El administrador realiza las actualizaciones deseadas.	4.1. El sistema verifica que los datos introducidos sean válidos y que los requeridos no estén vacíos. 4.2. El sistema actualiza la información. 4.3. Muestra el listado de los edificios y finaliza el caso de uso.
Curso alternativo	
	4.2. Se emite un mensaje de error para que se llenen los campos obligatorios o se introduzcan correctamente los datos.
Sección "Eliminar Edificio"	
3. El administrador selecciona dentro del listado de edificios el que desea eliminar.	3.1. El sistema muestra un mensaje de confirmación. 3.2. El sistema elimina el edificio y finaliza el caso de uso.
Sección "Buscar Edificio"	
3. El administrador desea ver los datos de un edificio específico y selecciona	3.1. El sistema busca el edificio y muestra sus datos en el

su número.	listado.
Prototipo	Ver Anexo 13.

Tabla 3.7 Descripción del caso de uso “Gestionar Edificio”.

3.6.2.7 Caso de uso “Gestionar Manzana”.

Nombre del Caso de Uso	**Gestionar Manzana**
Actores	Administrador
Propósito	Garantizar que los datos de cada manzana se registren, se modifiquen y se eliminen en el sistema.
Resumen	El caso de uso se inicia cuando el administrador accede al sistema y solicita Gestionar Manzana, inmediatamente se visualiza una pantalla donde se muestran las manzanas existentes en el sistema, a partir de aquí el sistema brinda la posibilidad de efectuar las operaciones de agregar, modificar o eliminar una de ellas. Finaliza una vez que el administrador realiza una de las operaciones.
Referencias	R1.3
Precondiciones	
Poscondiciones	Para los siguientes procesos: Insertar: se crea una instancia de la clase manzana. Modificar: se modifican los atributos de la instancia seleccionada de la clase manzana. Eliminar: se elimina la instancia seleccionada de la clase manzana.
Curso normal de los eventos	

Acciones del actor	**Respuestas del Sistema**
1. El administrador decide administrar manzanas y selecciona la opción.	1.1. El sistema muestra una pantalla con el listado de las manzanas registradas.
2. El administrador selecciona una de las operaciones a realizar.	2.1. El sistema ejecuta una de las siguientes acciones: • Si decide agregar una manzana, ir a la sección “Agregar manzana”. • Si decide modificar una manzana, ir a la sección “Modificar manzana”. • Si decide eliminar una manzana, ir a la sección “Eliminar manzana”. • Si decide buscar una manzana, ir a la sección “Buscar manzana”.
Sección “Agregar Manzana”	
3. El administrador introduce los datos de la manzana.	3.1. El sistema verifica que los datos introducidos sean válidos y que los requeridos no estén vacíos. 3.2. El sistema verifica que la manzana no exista. 3.3. La manzana se registra en el sistema.

	3.4. El sistema muestra un mensaje comunicando que la manzana se agregó satisfactoriamente, dando la posibilidad de agregar otra manzana y finaliza así el caso de uso.
Curso alternativo	
	3.2. Se emite un mensaje de error para que se llenen los campos obligatorios o se introduzcan correctamente los datos.
	3.3. Se emite un mensaje informando de la existencia de la manzana.
Sección "Modificar Manzana"	
3. El administrador selecciona dentro del listado de manzanas la que desea modificar.	3.1. El sistema busca la manzana y muestra los datos en una nueva pantalla.
4. El administrador realiza las actualizaciones deseadas.	4.1. El sistema verifica que los datos introducidos sean válidos y que los requeridos no estén vacíos. 4.2. El sistema actualiza la información. 4.3. Muestra el listado de las manzanas y finaliza el caso de uso.

Curso alternativo	
	4.2. Se emite un mensaje de error para que se llenen los campos obligatorios o se introduzcan correctamente los datos.
Sección "Eliminar Manzana"	
3. El administrador selecciona dentro del listado de manzanas la que desea eliminar.	3.1. El sistema muestra un mensaje de confirmación. 3.2. El sistema elimina la manzana y finaliza el caso de uso.
Sección "Buscar Manzana"	
3. El administrador desea ver los datos de una manzana específica y selecciona su nombre.	3.1. El sistema busca la manzana y muestra sus datos en el listado.
Prototipo	Ver Anexo 14.

Tabla 3.8 Descripción del caso de uso "Gestionar Manzana".

3.6.3 Paquete Gestionar Ubicación.

En este paquete se recogen las funcionalidades propias del flujo de los procesos del alojamiento, orientado a los distintos tipos de personas que reciben dicho servicio. A continuación se muestra el diagrama de casos de uso del sistema correspondiente al paquete.

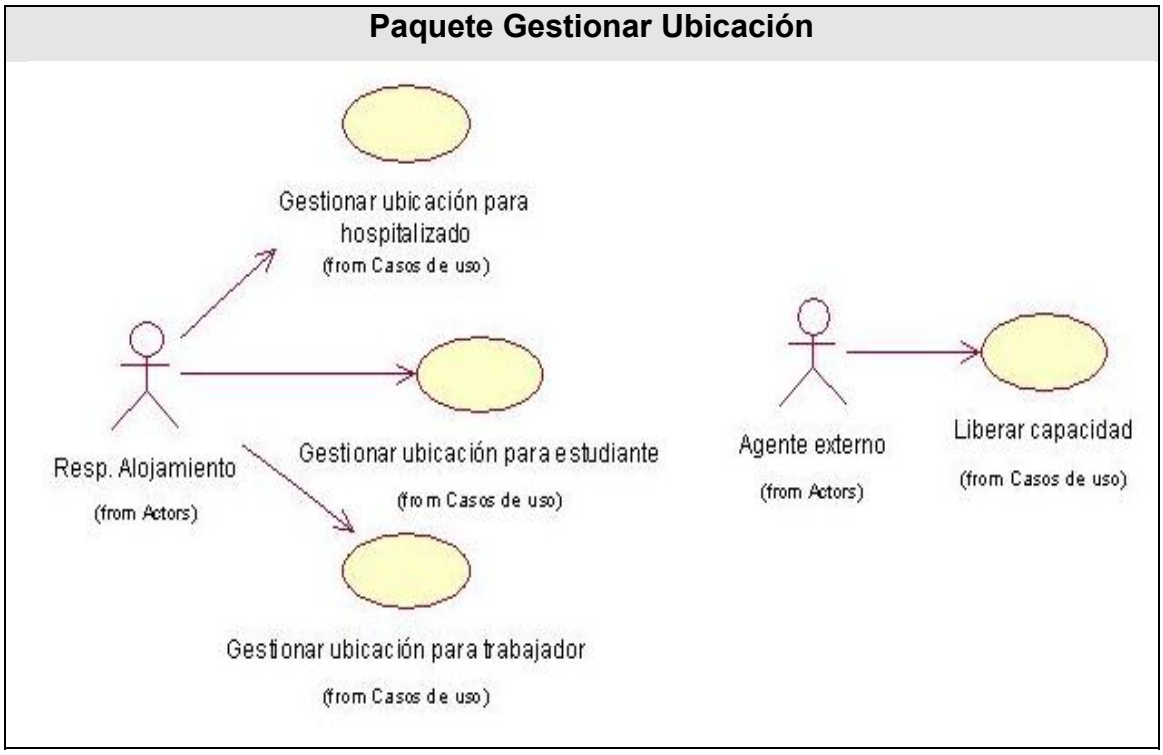


Figura 3.3 Diagrama de casos de uso del sistema del paquete Gestionar Ubicación.

3.6.3.1 Caso de uso "Gestionar Ubicación para estudiante".

Nombre del Caso de Uso	**Gestionar Ubicación para estudiante**
Actores	Responsable de alojamiento
Propósito	Poder ubicar de forma automática a los estudiantes planificados en cada etapa de la misión brindando la posibilidad de reubicación.
Resumen	El caso de uso se inicia cuando el responsable de alojamiento accede al sistema y solicita gestionar ubicaciones para estudiantes, a partir de aquí el sistema brinda la posibilidad de efectuar las operaciones de ubicar de forma automática o reubicar estudiantes de forma manual o asistida. Finaliza una vez que el administrador realiza una de las operaciones.

Referencias	R2.1
Precondiciones	El estudiante debe estar acreditado.
Poscondiciones	Se asignará una nueva ubicación al estudiante.
Requerimientos especiales	Para reubicar un estudiante debe habérsele asignado una ubicación previamente.

Curso normal de los eventos	
Acciones del actor	**Respuestas del Sistema**
1. El responsable de alojamiento decide gestionar ubicación para estudiantes y selecciona la operación.	1.1. El sistema ejecuta una de las siguientes acciones: • Si decide ubicar estudiantes, ir a la sección “Ubicar estudiantes”. • Si decide reubicar un estudiante, ir a la sección “Reubicar estudiantes”.
Sección “Ubicar Estudiantes”	
	1.2. El sistema muestra el listado de estudiantes acreditados pendientes de ubicación.
2. El responsable de alojamiento selecciona ubicar automáticamente.	2.1. El sistema asigna una ubicación a cada estudiante del listado. 2.2. Muestra el listado de los estudiantes con la ubicación asignada y finaliza el caso de

	uso.
Sección “Reubicar Estudiantes”	
2. El responsable de alojamiento selecciona el método a utilizar.	2.1. El sistema ejecuta una de las siguientes acciones: • Si decide reubicar estudiantes manualmente, ir a la sección “Reubicar estudiantes de forma manual”. • Si decide reubicar un estudiante de forma asistida, ir a la sección “Reubicar estudiantes de forma asistida”.
Sección “Reubicar Estudiantes de forma manual”	
	2.2. El sistema muestra la pantalla que permite seleccionar al estudiante.
3. Selecciona el estudiante.	3.1. El sistema muestra una pantalla con el estudiante seleccionado y el mapa de capacidades.
4. Selecciona la ubicación.	4.1. Asigna la ubicación al estudiante liberando la anterior. 4.2. Muestra un mensaje que indica que el estudiante ha sido reubicado y finaliza el caso de

	uso.
Sección "Reubicar Estudiantes de forma asistida"	
	2.2. El sistema muestra la pantalla que permite seleccionar al estudiante.
3. Selecciona el estudiante a reubicar.	3.1. Muestra el estudiante seleccionado así como el mapa de capacidades, con filtros para los distintos tipos de capacidades.
4. Selecciona la ubicación a partir de un criterio de búsqueda de capacidades.	4.1. Asigna la ubicación al estudiante liberando la anterior. 4.2. Muestra un mensaje que indica que el estudiante ha sido reubicado y finaliza el caso de uso.
Curso alternativo	
Prototipo	Ver Anexo 15.

Tabla 3.9 Descripción del caso de uso "Gestionar Ubicación para estudiante".

3.6.3.2 Caso de uso "Gestionar Ubicación para trabajador".

Nombre del Caso de Uso	**Gestionar Ubicación para trabajador**
Actores	Responsable de alojamiento
Propósito	Poder ubicar al trabajador de manera asistida o manual

	brindando la posibilidad de reubicación en caso necesario.
Resumen	El caso de uso se inicia cuando el responsable de alojamiento accede al sistema y solicita gestionar ubicaciones para trabajadores, a partir de aquí el sistema brinda la posibilidad de efectuar las operaciones de ubicar o reubicar trabajadores de forma manual o asistida. Finaliza una vez que el administrador realiza una de las operaciones.
Referencias	R2.2
Precondiciones	El trabajador debe estar acreditado.
Poscondiciones	Se asignará una nueva capacidad al trabajador.
Requerimientos especiales	Para reubicar un trabajador debe habérsele asignado una ubicación previamente.

Curso normal de los eventos	
Acciones del actor	**Respuestas del Sistema**
1. El responsable de alojamiento decide gestionar ubicación para trabajador y selecciona la operación.	1.1. El sistema ejecuta una de las siguientes acciones: • Si decide ubicar trabajador, ir a la sección “Ubicar trabajadores”. • Si decide reubicar un trabajador, ir a la sección “Reubicar trabajadores”.
Sección “Ubicar Trabajadores”	
2. El responsable de alojamiento	2.1. El sistema ejecuta una de las

selecciona el método a utilizar.	siguientes acciones: • Si decide ubicar trabajadores manualmente, ir a la sección "Ubicar trabajadores de forma manual". • Si decide ubicar un trabajador de forma asistida, ir a la sección "Ubicar trabajadores de forma asistida".
Sección "Ubicar Trabajadores de forma manual"	
	2.2. El sistema muestra una pantalla con el listado de los trabajadores dando la posibilidad de seleccionar uno.
3. Selecciona el trabajador a ubicar.	3.1. El sistema muestra una pantalla con los datos del trabajador seleccionado y el mapa de capacidades.
4. El responsable de alojamiento selecciona la ubicación.	4.1. Asigna la ubicación al trabajador. 4.2. Muestra un mensaje que indica que el trabajador ha sido ubicado y finaliza el caso de uso.
Sección "Ubicar Trabajadores de forma asistida"	
	2.2. El sistema muestra el listado de trabajadores pendientes de

	ubicación.
3. Selecciona el trabajador a ubicar.	3.1. El sistema muestra una pantalla con los datos del trabajador seleccionado y el mapa de capacidades con filtros para los distintos tipos de capacidades.
4. Selecciona una ubicación a partir de un criterio de búsqueda de capacidades.	4.1. Asigna la ubicación al trabajador. 4.2. Muestra un mensaje que indica que el trabajador ha sido ubicado y finaliza el caso de uso.
Sección "Reubicar Trabajadores"	
2. El responsable de alojamiento selecciona el método a utilizar.	2.1. El sistema ejecuta una de las siguientes acciones: • Si decide reubicar trabajadores manualmente, ir a la sección "Reubicar trabajadores de forma manual". • Si decide reubicar un trabajador de forma asistida, ir a la sección "Reubicar trabajadores de forma asistida".
Sección "Reubicar Trabajadores de forma manual"	
	2.2. El sistema muestra la pantalla que permite seleccionar el

	trabajador que se desea reubicar.
3. Selecciona el trabajador a reubicar.	3.1. El sistema muestra una pantalla con los datos del trabajador y el mapa de capacidades.
4. El responsable de alojamiento selecciona la ubicación para el trabajador.	4.1. Asigna la ubicación al trabajador liberando la anterior. 4.2. Muestra un mensaje que indica que el trabajador ha sido reubicado y finaliza el caso de uso.
Sección "Reubicar Trabajadores de forma asistida"	
	2.2. El sistema muestra la pantalla que permite seleccionar el trabajador que se desea reubicar.
3. Selecciona el trabajador a reubicar.	3.1. El sistema muestra una pantalla con los datos del trabajador y el mapa de capacidades con filtros para los distintos tipos de capacidades.
4. Selecciona una ubicación a partir de un criterio de búsqueda de capacidades.	4.1. Asigna la ubicación al trabajador liberando la anterior. 4.2. Muestra un mensaje que indica que el trabajador ha sido

	reubicado y finaliza el caso de uso.
Prototipo	Ver Anexo 16.

Tabla 3.10 Descripción del caso de uso “Gestionar Ubicación para trabajador”.

3.6.3.3 Caso de uso “Gestionar Ubicación para hospitalizado”.

Nombre del Caso de Uso	**Gestionar Ubicación para hospitalizado**
Actores	Responsable de alojamiento
Propósito	Poder ubicar al hospitalizado de manera asistida o manual brindando la posibilidad de reubicación en caso necesario.
Resumen	El caso de uso se inicia cuando el responsable de alojamiento accede al sistema y solicita gestionar ubicaciones para hospitalizados, a partir de aquí el sistema brinda la posibilidad de efectuar las operaciones de ubicar o reubicar hospitalizados de forma manual o asistida. Finaliza una vez que el administrador realiza una de las operaciones.
Referencias	R2.3
Precondiciones	El hospitalizado debe estar acreditado.
Poscondiciones	Se asignará una nueva capacidad al trabajador.
Requerimientos especiales	Para reubicar un hospitalizado debe habérsele asignado una ubicación previamente.
Curso normal de los eventos	
Acciones del actor	**Respuestas del Sistema**

1. El responsable de alojamiento decide gestionar ubicación para hospitalizados y selecciona la operación.	1.1. El sistema ejecuta una de las siguientes acciones: • Si decide ubicar hospitalizado, ir a la sección "Ubicar hospitalizados". • Si decide reubicar un hospitalizado, ir a la sección "Reubicar hospitalizados".
Sección "Ubicar Hospitalizados"	
2. El responsable de alojamiento selecciona el método a utilizar.	2.1. El sistema ejecuta una de las siguientes acciones: • Si decide ubicar hospitalizado manualmente, ir a la sección "Ubicar hospitalizados de forma manual". • Si decide ubicar un hospitalizado de forma asistida, ir a la sección "Ubicar hospitalizados de forma asistida".
Sección "Ubicar Hospitalizados de forma manual"	
	2.2. El sistema muestra la pantalla que permite realizar dicha operación.
3. Selecciona el hospitalizado a ubicar.	3.1. El sistema muestra el mapa de capacidades.
4. El responsable de alojamiento	4.1. Asigna la ubicación al

selecciona la ubicación.	hospitalizado. 4.2. Muestra un mensaje que indica que el hospitalizado ha sido ubicado y finaliza el caso de uso.
Sección "Ubicar Hospitalizados de forma asistida"	
	2.2. El sistema muestra el listado de los grupos pendientes de alojamiento.
3. Selecciona el grupo a ubicar.	3.1. El sistema muestra el listado de los miembros del grupo y el mapa de capacidades con filtros para los distintos tipos de capacidades.
4. El responsable de alojamiento selecciona ubicación para cada uno de los miembros del grupo y presiona el botón "Ubicar".	4.1. Asigna la ubicación a los hospitalizados pertenecientes al grupo. 4.2. Muestra un mensaje que indica que los hospitalizados han sido ubicados y finaliza el caso de uso.
Sección "Reubicar Hospitalizados"	
2. El responsable de alojamiento selecciona el método a utilizar.	2.1. El sistema ejecuta una de las siguientes acciones: • Si decide reubicar hospitalizados manualmente, ir a la sección "Reubicar hospitalizados de

	forma manual". • Si decide reubicar un hospitalizado de forma asistida, ir a la sección "Reubicar hospitalizados de forma asistida".
Sección "Reubicar Hospitalizados de forma manual"	
	2.2. El sistema muestra la pantalla que permite seleccionar el trabajador que se desea reubicar.
3. Selecciona el hospitalizado a reubicar.	3.1. El sistema muestra los datos del hospitalizado seleccionado y el mapa de capacidades.
4. Selecciona la ubicación para el hospitalizado.	4.1. Asigna la ubicación al hospitalizado liberando la anterior. 4.2. Muestra un mensaje que indica que el hospitalizado ha sido reubicado y finaliza el caso de uso.
Sección "Reubicar Hospitalizados de forma asistida"	
	2.2. El sistema muestra la pantalla que permite seleccionar el trabajador que se desea reubicar.

3. Selecciona el hospitalizado a reubicar.	3.1. El sistema muestra los datos del hospitalizado y el mapa de capacidades con filtros para los distintos tipos de capacidades.
4. Selecciona una ubicación a partir de un criterio de selección de capacidades.	4.1. Asigna la ubicación al hospitalizado liberando la anterior. 4.2. Muestra un mensaje que indica que el hospitalizado ha sido reubicado y finaliza el caso de uso.
Curso alternativo	
Prototipo	Ver Anexo 17.

Tabla 3.11 Descripción del caso de uso “Gestionar Ubicación para hospitalizado”.

3.6.3.4 Caso de uso “Liberar capacidades”.

Nombre del Caso de Uso	**Liberar capacidades**
Actores	Agente externo
Propósito	Liberar la capacidad que ocupa una persona una vez que se detecte su salida del sistema.
Resumen	El caso de uso se inicia cuando el agente externo accede al sistema e indica que una persona determinada ha causado baja del mismo, ya sea por alta médica o traslado en el caso de los hospitalizados o por la terminación del período de trabajo en el caso de los trabajadores y estudiantes. El caso de uso finaliza

	cuando el sistema libera la capacidad.
Referencias	R2.4
Precondiciones	La persona debe tener una ubicación asignada.
Poscondiciones	La capacidad será liberada.
Curso normal de los eventos	
Acciones del actor	**Respuestas del Sistema**
1. El agente externo indica la salida de la persona del sistema.	1.1. El sistema verifica si tiene una capacidad asignada. 1.2. Libera la capacidad y finaliza el caso de uso.

Tabla 3.12 Descripción del caso de uso “Liberar capacidades”.

3.7 Conclusiones.

En este capítulo se comenzó a desarrollar la propuesta de solución, obteniéndose a partir del análisis de los procesos del negocio, un listado con las principales funcionalidades que debe tener el sistema y los requisitos adicionales, se representaron los diagramas de casos de uso del sistema, y finalmente se describieron las acciones de los actores del sistema con los casos de uso con los que interactúan. Gracias a esto ahora se puede empezar a construir el sistema tratando de que se cumplan todos los requerimientos y las funciones que han sido consideradas necesarias en este capítulo.

CAPÍTULO 4

Descripción de la solución propuesta

4.1 Introducción.

Tras la definición y descripción, en el anterior capítulo, de las funcionalidades deseadas y necesarias del sistema propuesto; se hace necesario definir cómo se desarrollará.

Este capítulo tiene el objetivo de plantear la concepción general del diseño del sistema propuesto y cómo se implementa éste. Así, se presentan los diagramas de clases Web que detallan la interacción de las distintas páginas; se estructura la información que se desea persista a través del diseño de la base de datos; se describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. Son también descritos los estándares de diseño y programación seguidos.

4.2 Clases Base.

Los casos de uso del sistema definidos en el capítulo anterior, deben encajar en la arquitectura cuando se llevan a cabo, mientras que la arquitectura debe permitir el desarrollo de los casos de uso requeridos ahora y en el futuro. De esa manera, la arquitectura debe diseñarse para permitir que el sistema evolucione, que los desarrolladores puedan progresar hasta obtener una visión común, que se organice el desarrollo del software y que se fomente la reutilización. A partir de esto se definen explícitamente interfaces, haciendo posible una buena comunicación entre los desarrolladores. También se consideran las posibilidades de reutilización de las partes del sistema parecidas o de productos softwares generales. Los subsistemas, interfaces u otros elementos del diseño se añaden posteriormente.

Teniendo en cuenta las posibilidades que ofrece el patrón Modelo Vista Controlador, descrito en el Capítulo 1 se decide hacer uso del mismo para definir el sistema de clases base de la aplicación, pues se persigue facilidad de mantenimiento, escalabilidad, rapidez de desarrollo e independencia entre las distintas capas del sistema para facilitar su futura evolución.

4.2.1 Diagrama de Clases.

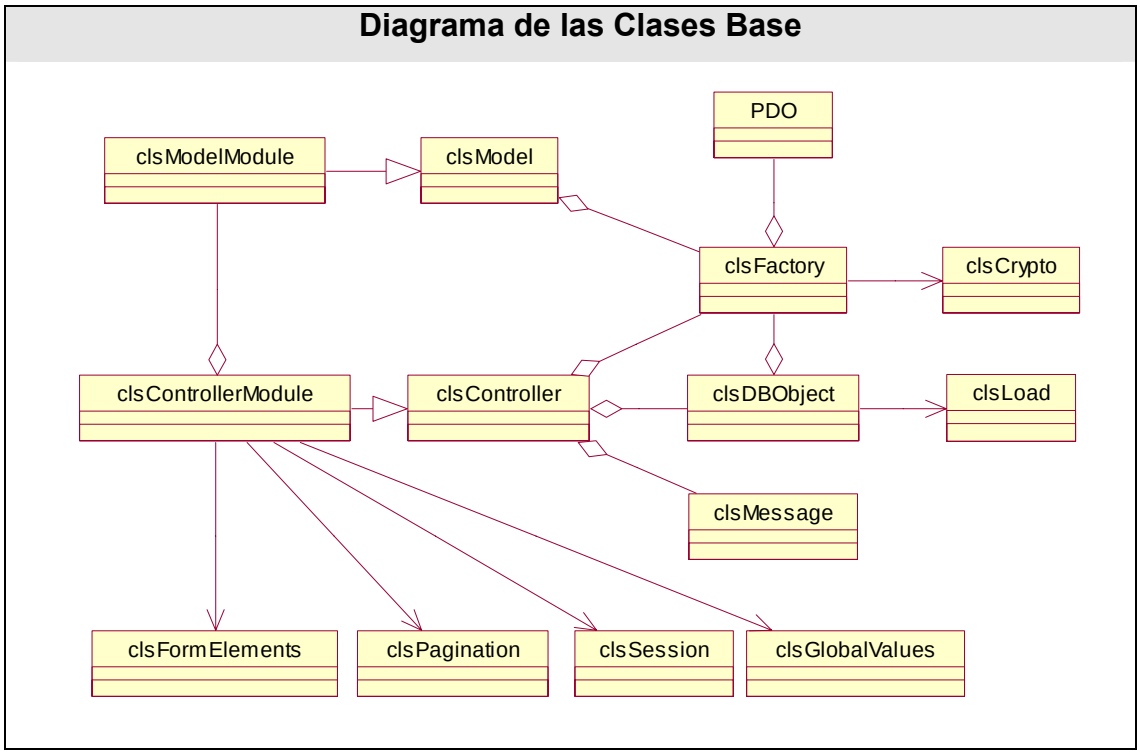


Figura 4.1 Diagrama de las Clases Base.

4.2.2 Descripción de las clases.

clsControllerLodging	Clase que gestiona la interacción entre el modelo y las vistas del módulo.
Métodos	
AdministerBeds	Carga la vista para la administración de camas y establece la relación entre la vista y el modelo
AddBeds	Carga la vista para la adición o modificación de camas y establece la relación con el modelo.
AdministerRooms	Carga la vista para la administración de cuartos y establece la relación entre la vista y el modelo.
AddRooms	Carga la vista para la adición o modificación de cuartos y establece la relación con el modelo.
AdministerApartments	Carga la vista para la administración de apartamentos y establece la relación entre la vista

	y el modelo.
AddApartments	Carga la vista para la adición o modificación de apartamentos y establece la relación con el modelo.
AdministerBuildings	Carga la vista para la administración de edificios y establece la relación entre la vista y el modelo.
AddBuildings	Carga la vista para la adición o modificación de edificios y establece la relación con el modelo.
AdministerBlocks	Carga la vista para la administración de manzanas y establece la relación entre la vista y el modelo.
AddBlock()	Carga la vista para la adición o modificación de manzanas y establece la relación con el modelo.
AdministerResidences	Carga la vista para la administración de residencias y establece la relación entre la vista y el modelo.
AddResidences	Carga la vista para la adición o modificación de residencias y establece la relación con el modelo.
AdministerClinic	Carga la vista para la administración de clínicas y establece la relación entre la vista y el modelo.
AddClinic	Carga la vista para la adición o modificación de clínicas y establece la relación con el modelo.
Locate	Carga la vista para la gestión del proceso de ubicación
Relocate	Carga la vista para la gestión del proceso de reubicación

Tabla 4.1 Descripción de la clase clsControllerLodging.

clsModelLodging	Clase que contiene la lógica del negocio y se encarga del acceso a datos.
Métodos	
GetDataBed()	Devuelve el listado de camas atendiendo a un criterio de filtro.

GetDataRoom()	Devuelve el listado de los cuartos atendiendo a un criterio de filtro.
GetDataApartment()	Devuelve el listado de los apartamentos atendiendo a un criterio de filtro.
GetDataBuilding()	Devuelve el listado de los edificios atendiendo a un criterio de filtro.
GetDataBlock ()	Devuelve el listado de las manzanas atendiendo a un criterio de filtro.
GetDataResidence()	Devuelve el listado de las residencias atendiendo a un criterio de filtro.
GetDataClinic()	Devuelve el listado de las clínicas atendiendo a un criterio de filtro.
Locate()	Asigna una ubicación a una persona determinada en la base de datos.
Relocate()	Indica la reubicación de una persona determinada en la base de datos.

Tabla 4.2 Descripción de la clase clsModelLodging.

clsController	Clase que funciona como espina dorsal del sistema, la que permite gestionar el proceso y la validación de datos, y la interacción entre el modelo y las vistas del sistema.
Métodos	
ProcessData	Se encarga de procesar toda la información llegada desde un formulario, teniendo en cuenta el contenido de los datos asociados.
ExtractVarsTo	Extrae de un arreglo solamente los índices con los valores asociados que se desee, aplicándole además la validación.
Validate	Valida el contenido de una variable.
Clean	Elimina los espacios en una cadena y la limpia de

	caracteres no legibles.
IsPostBack	Determina si existe la llegada de variables por el método POST.
CreatePKForm	Crea un objeto HTML input de tipo “hidden” con la clave primaria del objeto DBObject asociado.
ConvertToArray	Devuelve en un arreglo los atributos de un objeto y sus valores.
GetService	Crea una instancia de la clase controladora de otro módulo.
LoadModel	Crea una instancia de la clase modelo del módulo.
LoadView	Carga una vista del módulo.

Tabla 4.3 Descripción de la clase clsController.

clsModel	Clase a través de la cual se gestiona el acceso a datos del sistema.
Métodos	
__construct	Constructor de la clase modelo del sistema, cargando la instancia de la capa gestora de acceso a datos.
GetEntityPagination	Realiza la paginación de una tabla de la base de datos.

Tabla 4.4 Descripción de la clase clsModel.

clsFactory	Clase que contiene los métodos de acceso a datos del sistema
Métodos.	
Instance	Crea una instancia de la clase actual en forma de singleton.
CreateStoreProcedure	Método para crear los procedimientos almacenados teniendo en cuenta una serie de parámetros.
ExecStoreProcedure	Ejecuta un procedimiento almacenado determinado

	teniendo en cuenta los parámetros pasados.
ExecQuery	Este método ejecuta una consulta a la base de datos.
ConstructFilters	Teniendo en cuenta el arreglo pasado y el tipo de filtro, este método construye una cadena formateada que estará lista para usarse en una consulta a la base de datos.
Count	Cuenta la cantidad de elementos devueltos en una consulta determinada.
Select	Construye una consulta SELECT teniendo en cuenta los parámetros pasados.
Insert	Construye una consulta INSERT teniendo en cuenta los parámetros pasados.
Delete	Construye una consulta DELETE teniendo en cuenta los parámetros pasados.
LastInsertId	Devuelve el último ID generado por una consulta Insert.
IsUsingSP	Devuelve si se están usando procedimientos almacenados.
errorInfo	En caso de existir algún error en el acceso a datos este devuelve el error que se generó.

Tabla 4.5 Descripción de la clase clsFactory.

clsDBObject	Esta clase está diseñada para crear una representación de una entidad de la base de datos, dando la posibilidad de poder alterar elementos de esta entidad, adicionando, actualizando o eliminando elementos.
Métodos	
GetEntity	Devuelve nombre de la entidad asociada a la clase.

Set	Asignar valores a atributos de clase, en caso de no existir el atributo se crea.
Get	Tomar valores de atributos de la clase.
GetFieldsList	Obtener lista de atributos de la clase, asociado a la base de datos.
IssetPK	Comprueba si existe la clave primaria, lo hace comprobando el nombre y que no esté vacía.
GetPK	Obtiene el valor de la clave primaria en caso de existir.
GetPKName	Devuelve el nombre la clave primaria en caso de existir.
Save	Llama a Update o Insert dependiendo del contenido de los atributos.
Update	Actualiza el contenido en la entidad asociada a la clase.
Insert	Inserta el contenido en la entidad asociada a la clase.
Delete	Eliminar los datos de la entidad que coincidan con en contenido actual de los atributos de la clase.
DeleteByPK	Eliminar los datos de la entidad que coincidan con la clave primaria.
LoadFieldsFromArray	Cargar datos desde un arreglo hacia los atributos de la clase y en caso de no existir los crea.
LoadFieldsFromDB	Cargar los nombres de los atributos desde la entidad existente en la base de datos.
LoadDataFromDB	Carga el contenido de los atributos desde la entidad existente en la base de datos.
Cleaning	Limpia todo el contenido de los atributos de la clase.
VarsDump	Devuelve un listado de todos los atributos de la clase.

GetLastResult	Obtiene el último resultado o acción devuelta por la clase.

Tabla 4.6 Descripción de la clase clsDBObject.

clsMessage	Controla todo los mensajes devueltos por el sistema y que se le muestran al usuario
Métodos	
Add	Agrega un mensaje a la clase teniendo en cuenta el tipo (Normal, de Error).
GetList	Devuelve un arreglo con los errores que ha almacenado la clase.
MessageCount	Devuelve la cantidad de mensajes de un tipo que contenga la clase.
ShowHTMLMessages	Muestra en un HTML formateado el listado de mensajes almacenado en el sitio.
Clear	Vacía todos los mensajes que contiene la clase.

Tabla 4.7 Descripción de la clase clsMessage.

clsCrypto	Cifrado y descifrado de datos
Métodos	
Encrypt	Cifra una cadena pasada por parámetro o el atributo cadena de la clase.
Decrypt	Descifra cadena pasada por parámetro o el atributo cadena de la clase.

Tabla 4.8 Descripción de la clase clsCrypto.

clsLoad	Carga datos desde cualquier fuente de datos
Métodos	
XmlFromString	Método para cargar un XML desde una cadena XML.

XmlFromFile	Método para cargar un XML desde un fichero.
DatabaseConfig	Cargar las configuraciones de acceso a la base de datos.
DatabaseStructure	Método para extraer la estructura de una tabla en la base de datos y pasarla a un arreglo.
ModuleConfig	Carga la configuración de un módulo desde un archivo de configuración de módulo.
ModulesXMLs	Carga los archivos XML de configuración de los módulos y los procesa.
MenuFromArray	Construye el menú desde un arreglo.
MessageXML	Carga todo el contenido del XML de mensajes hacia variables.
File	Incluye un fichero o listado de ficheros pasados en un arreglo.

Tabla 4.9 Descripción de la clase clsLoad.

clsFormElements	
Métodos	
ListBox	Construye un objeto List box teniendo en cuenta como parámetro un arreglo y el nombre del objeto.
ListBoxFromTable	Extrae un conjunto de datos desde una tabla de la base de datos para construir un List box.
Checkbox	Construye un listado de objetos de tipo Checkbox.
CheckboxFromTable	Extrae un conjunto de datos desde una tabla de la base de datos para construir un listado de Checkbox.

Tabla 4.10 Descripción de la clase clsFormElements.

clsPagination	Genera la paginación teniendo en cuenta un conjunto y un subconjunto de datos

Métodos	
SetCantidadPorPagina	Asigna la cantidad de elementos que se mostrará por páginas.
SetCantidadTotal	Asigna la cantidad total de elementos.
GetTotalFromDB	Extrae la cantidad total de elementos desde una tabla en base de datos.
getURL	Extrae de la URL la variable de control de paginación.
putTemplate	Muestra el HTML generado por la paginación.

Tabla 4.11 Descripción de la clase clsPagination.

clsSession	Controla las sesiones del sistema
Métodos	
Begin	Inicializa las sesiones.
Set	Crea una variable de sesión.
Get	Devuelve el valor de una variable de sesión.
Delete	Elimina una variable de sesión.
Clear	Elimina todas las variables de sesiones.
End	Destruye todos los datos guardados en una sesión.

Tabla 4.12 Descripción de la clase clsSession.

clsGlobalValues	Controlar los valores globales del sistema (POST, GET, SESSION)
Métodos	
Reset	Resetea los valores de una variables o de todas las variables.
Set	Asigna un valor a una variable o la crea en caso de no existir.
Get	Devuelve el valor de una variable.
Delete	Elimina una variable.

Exists	Determina si una variable existe teniendo en cuenta los parámetros pasados.
ImportFromVar	Importa el contenido de arreglo hacia las variables de la clase.

Tabla 4.13 Descripción de la clase clsGlobalValues.

4.2.3 Diagramas de secuencia.

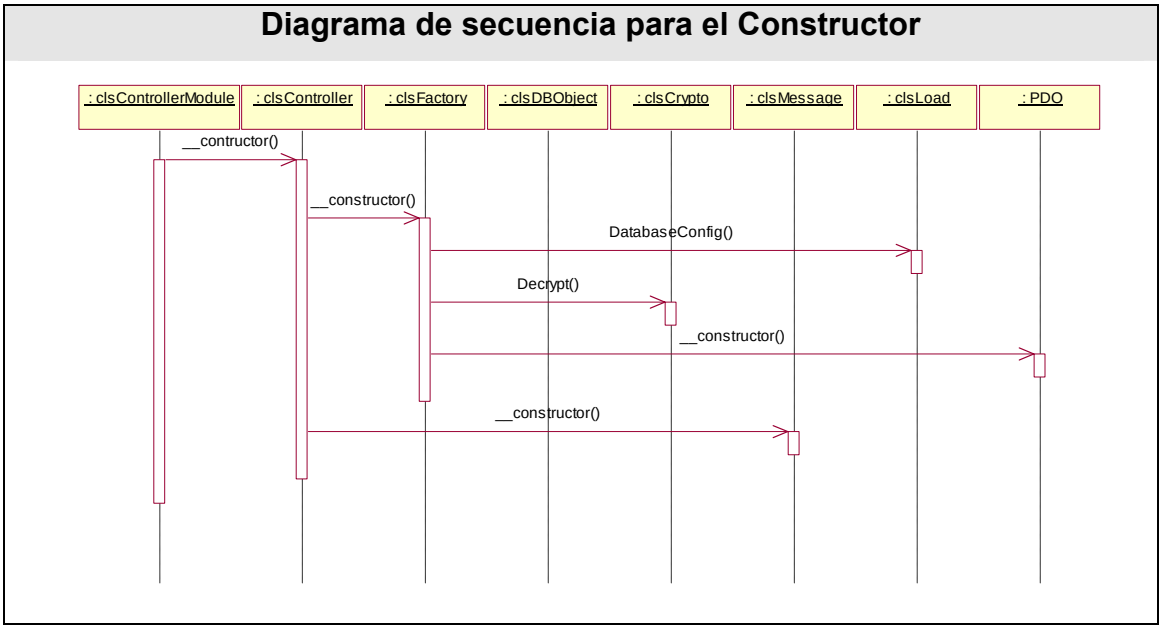


Figura 4.2 Diagrama de secuencia para el Constructor.

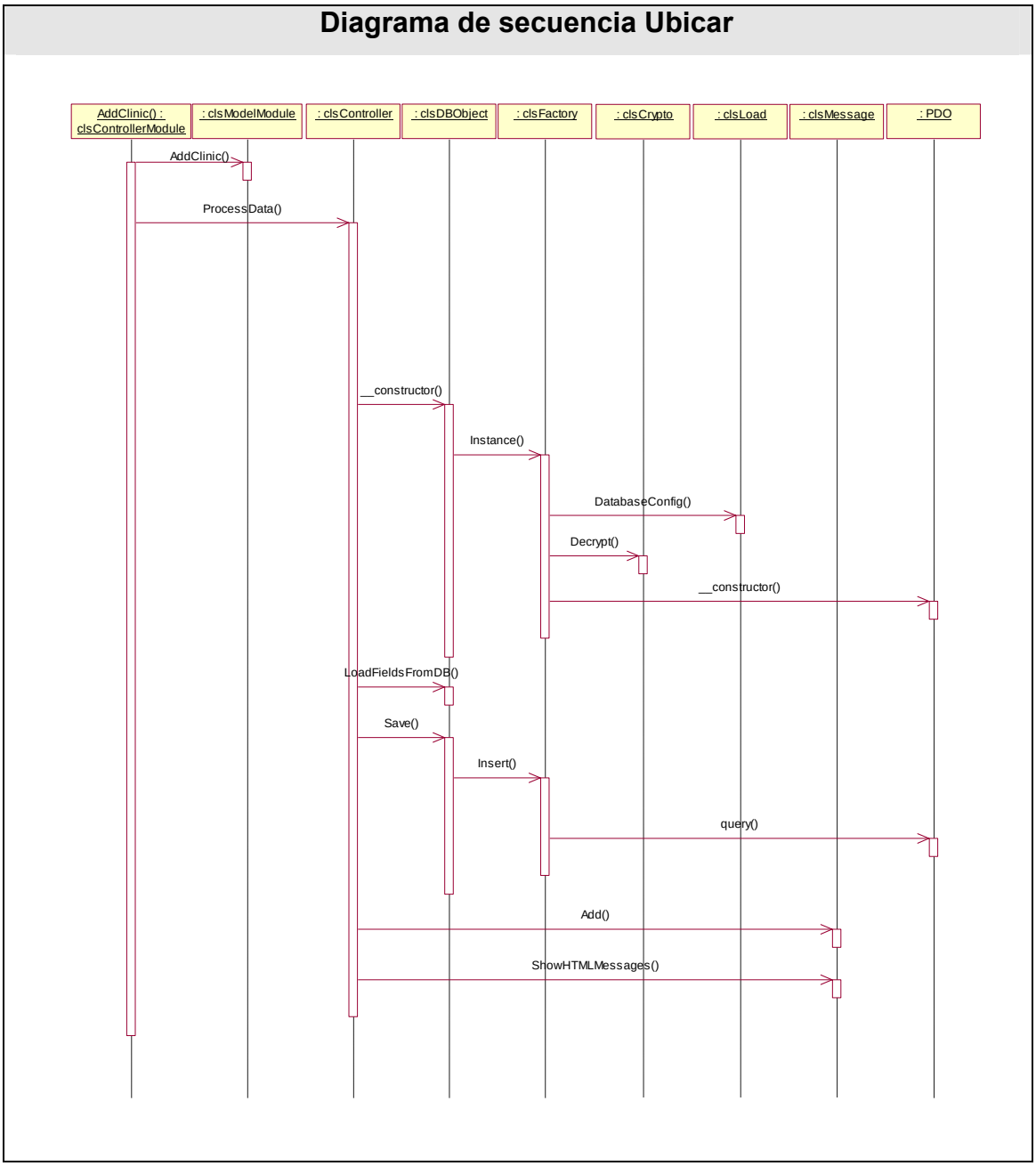


Figura 4.3 Diagrama de secuencia para el método AddClinic.

4.3 Diagrama de clases del diseño.

Un diagrama de clases presenta las clases del sistema con sus relaciones estructurales y de herencia, todo el código que irá creando las páginas, así como el contenido dinámico de estas una vez que estén en el navegador del cliente. En el caso de las aplicaciones Web, el diagrama de clases representa las colaboraciones que ocurren entre las páginas, donde cada página lógica puede ser representada como una clase,

es muy importante pues estos son los artefactos que se necesitan modelar para que el desarrollador los implemente y obtener así el producto final con la calidad requerida. Al tratar de utilizar el diagrama de clases tradicional para modelar aplicaciones Web surgen varios problemas, por lo cual los especialistas del Rational plantearon la creación de una extensión al modelo de análisis y diseño que permitiera representar el nivel de abstracción adecuado y la relación con los restantes artefactos de UML.[7]

El diagrama de clases Web, fue definido, a partir de los diferentes casos de uso del sistema y empleando las extensiones de UML para Web, a continuación se muestran los diagramas de clases para los distintos paquetes.

4.3.1 Paquete “Administrar Alojamiento”

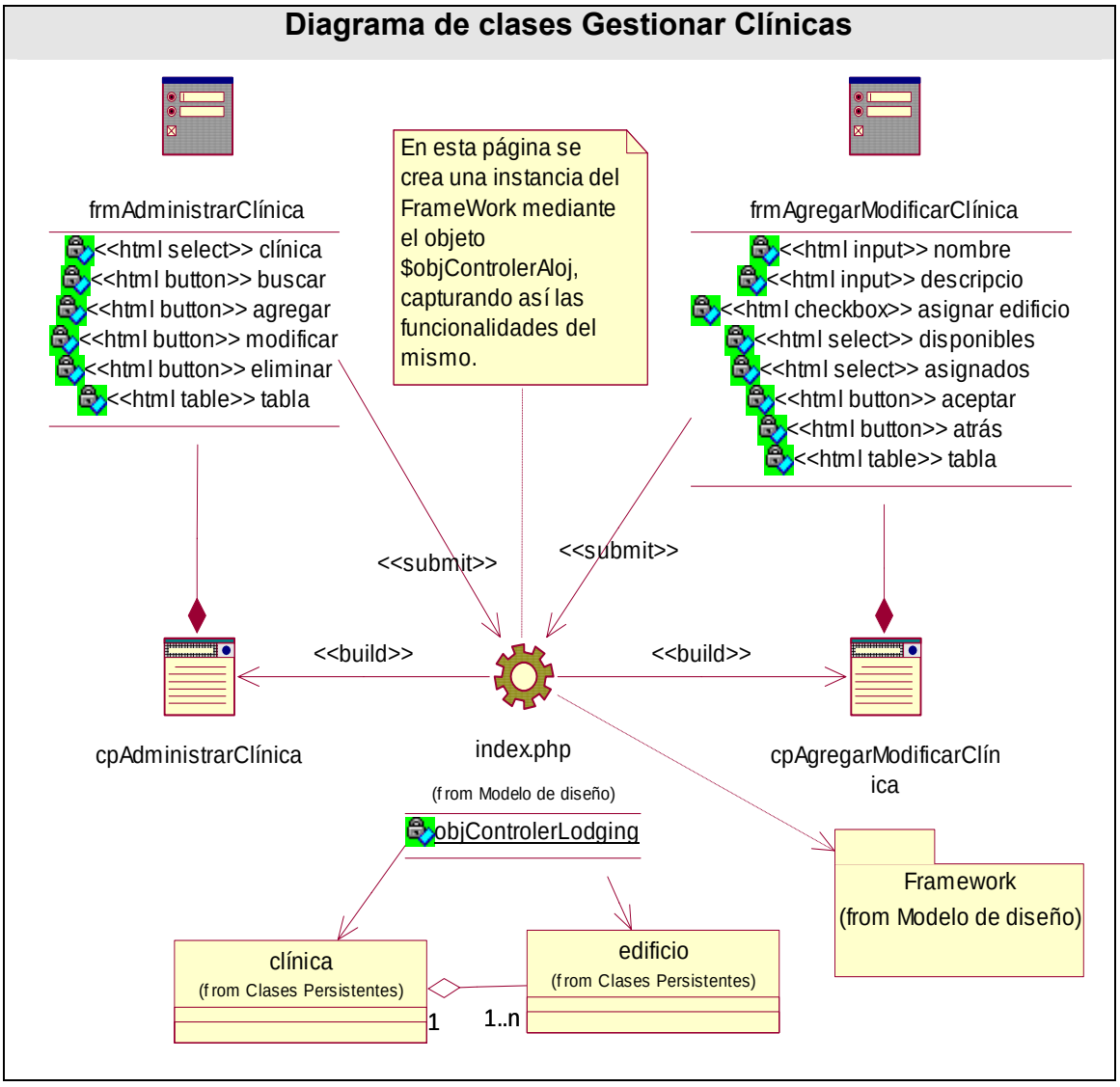


Figura 4.4 Diagrama de clases Gestionar Clínicas.

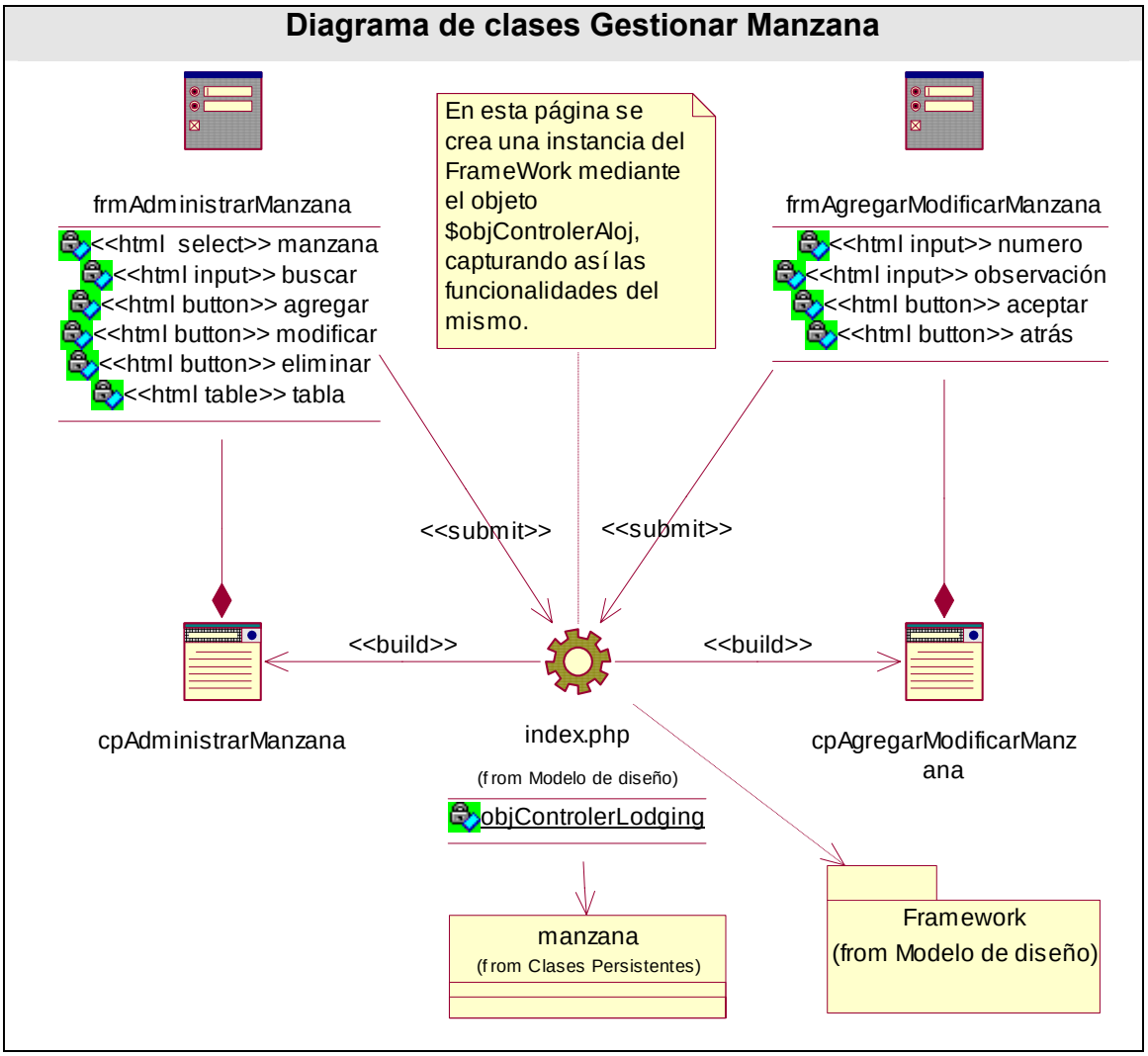


Figura 4.5 Diagrama de clases Gestionar Manzana.

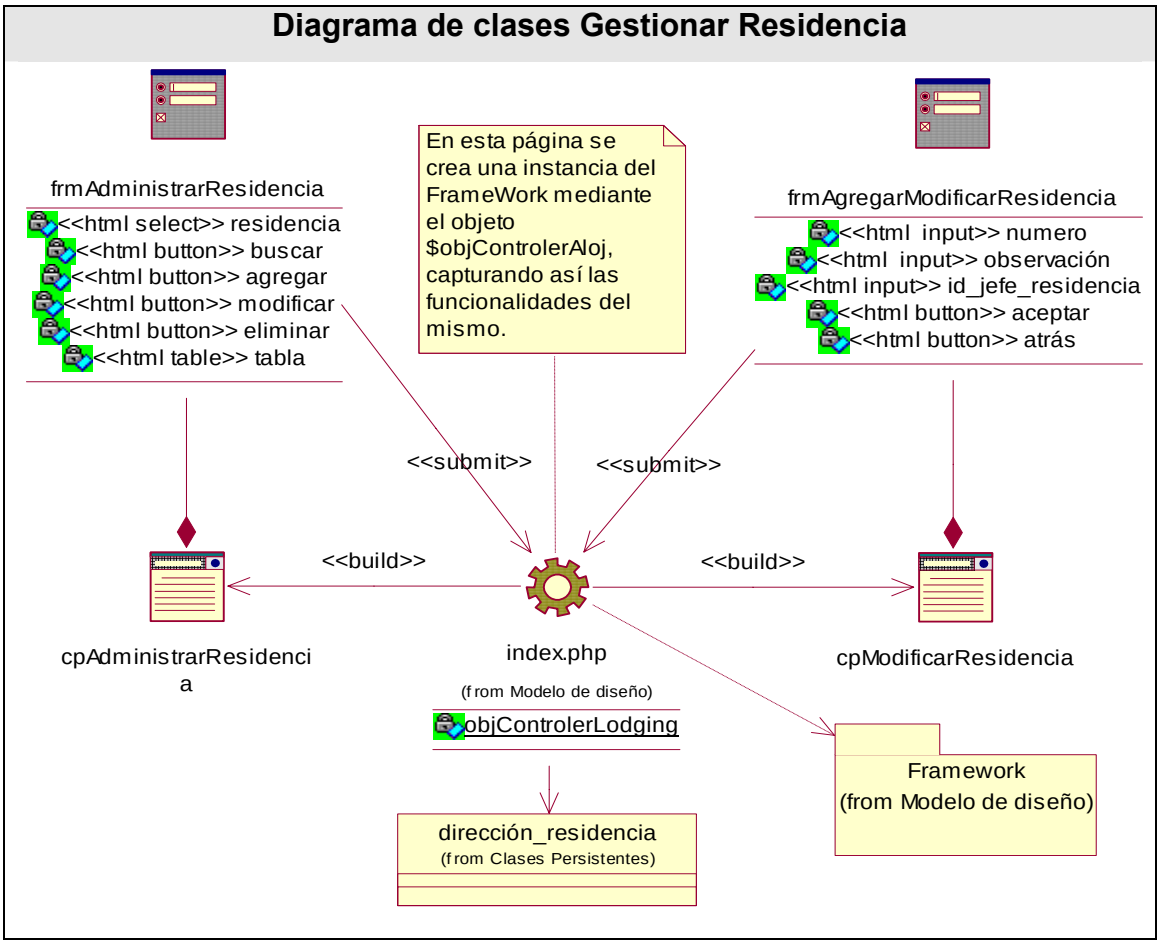


Figura 4.6 Diagrama de clases Gestionar Residencia.

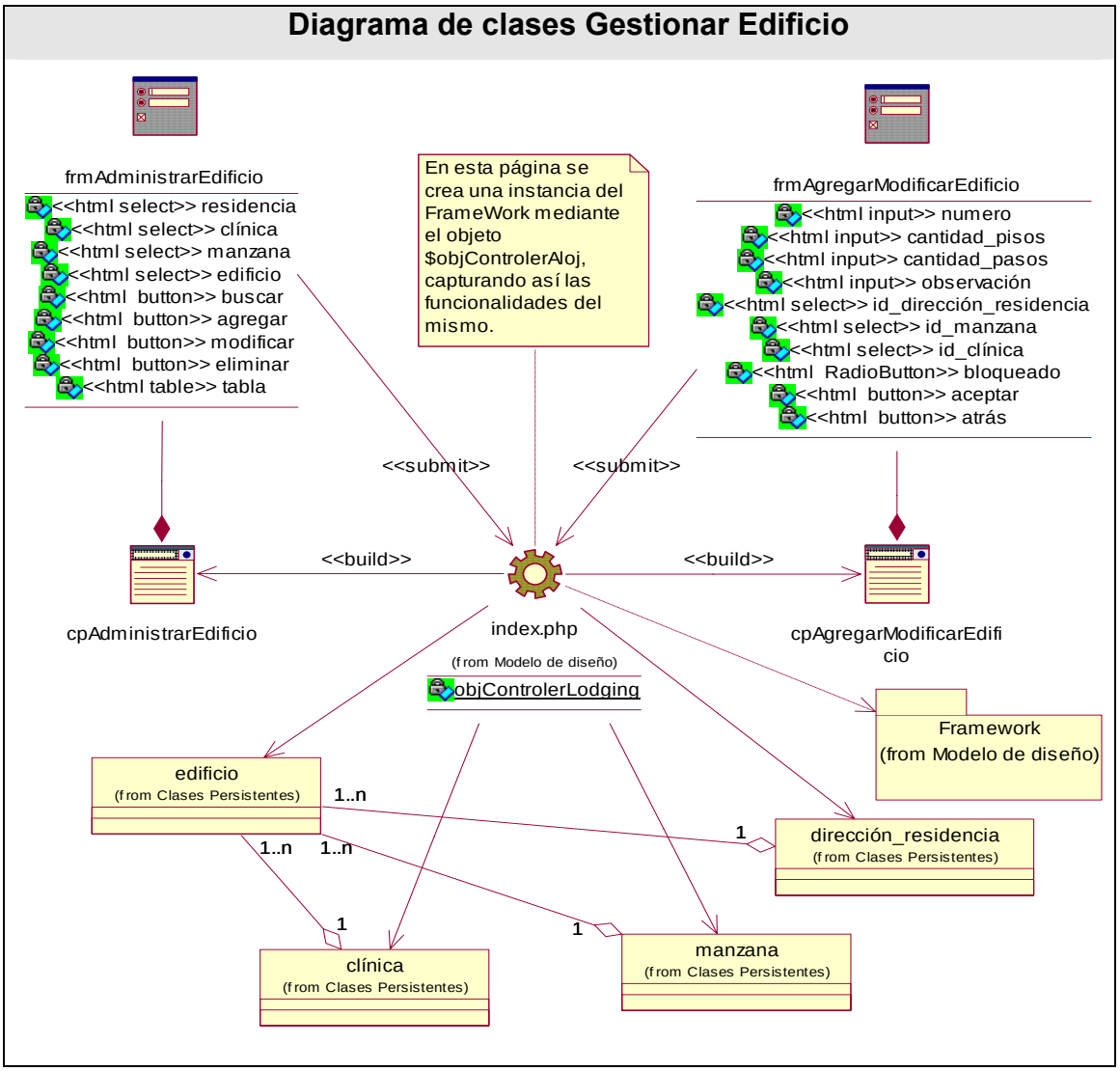


Figura 4.7 Diagrama de clases Gestionar Edificio.

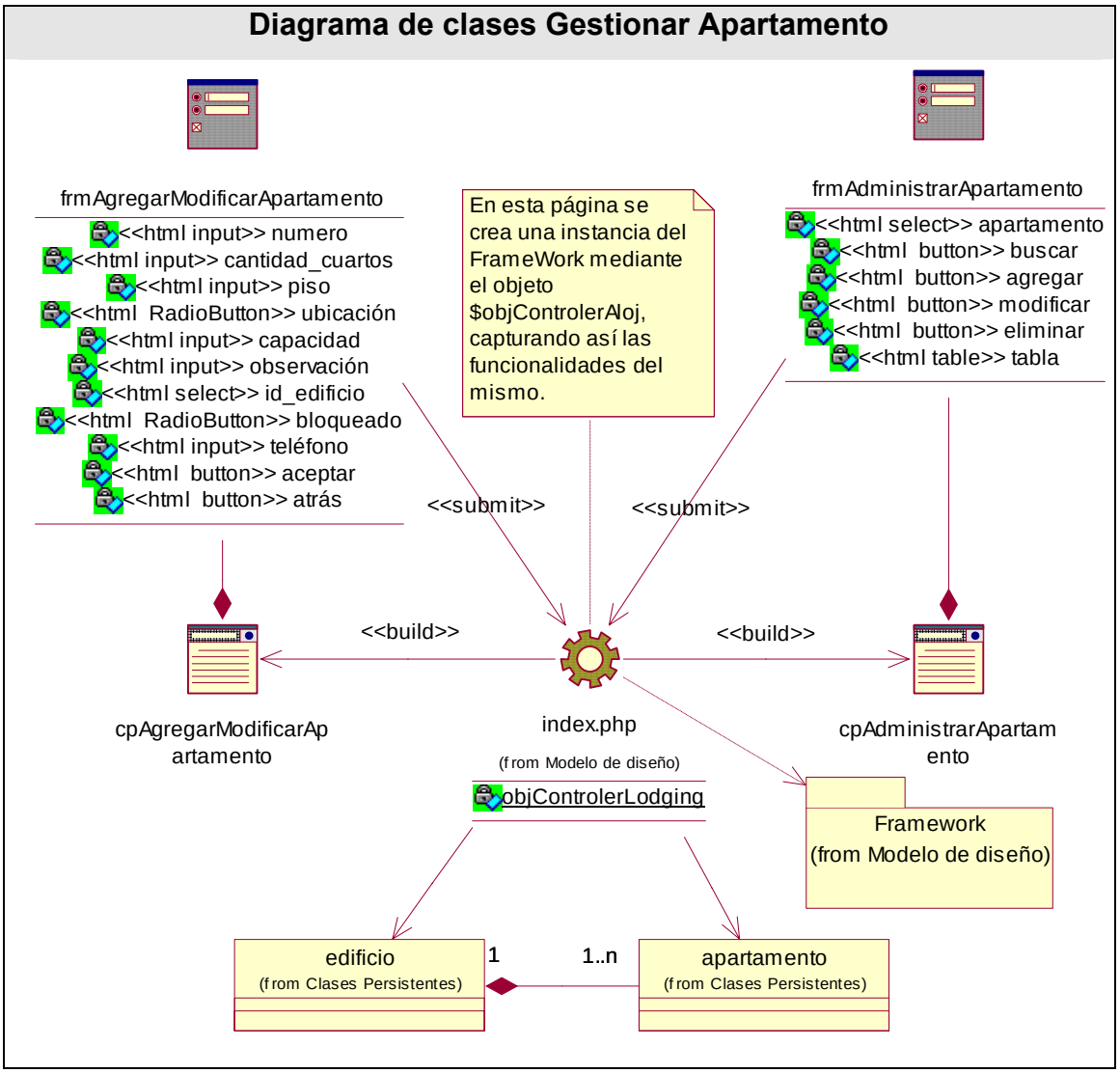


Figura 4.8 Diagrama de clases Gestionar Apartamento.

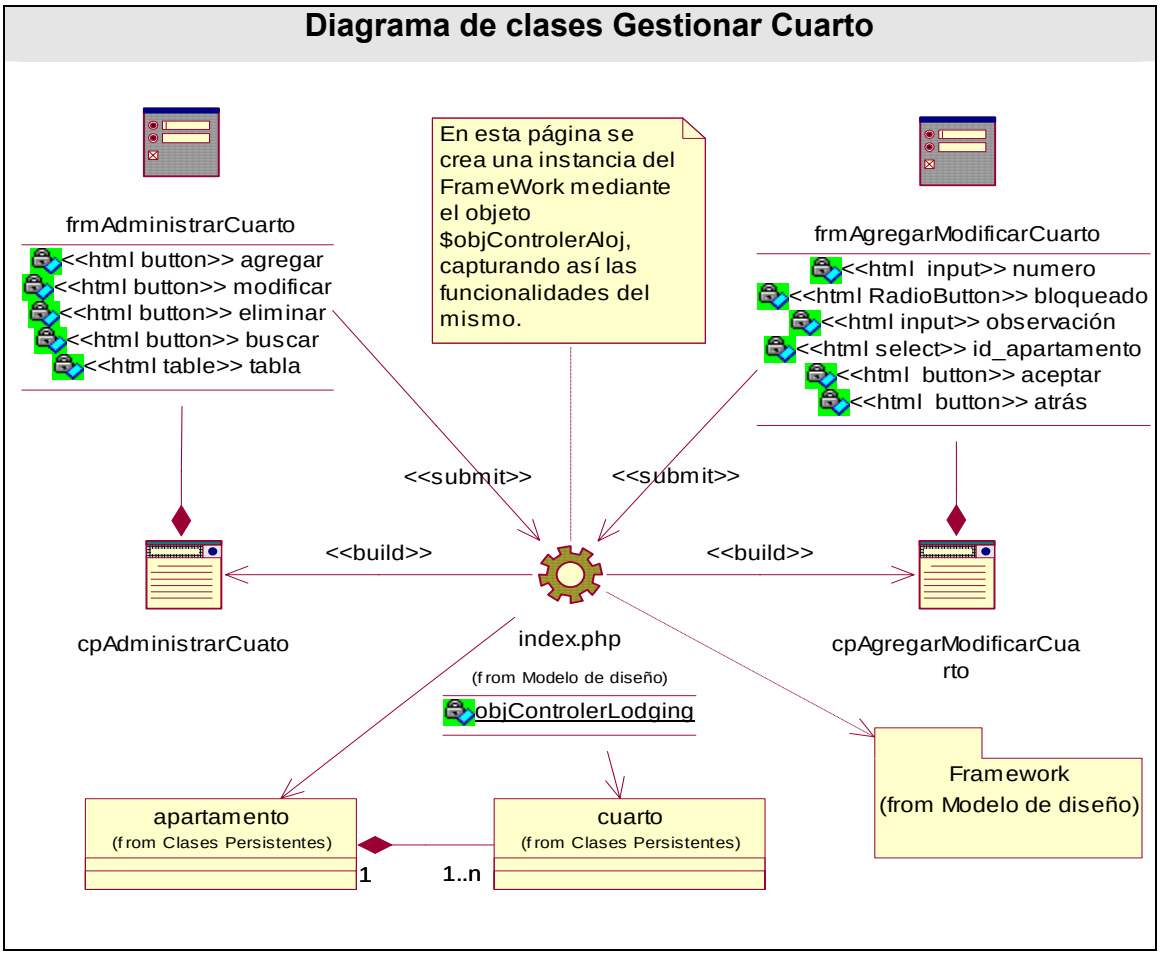


Figura 4.9 Diagrama de clases Gestionar Cuarto.

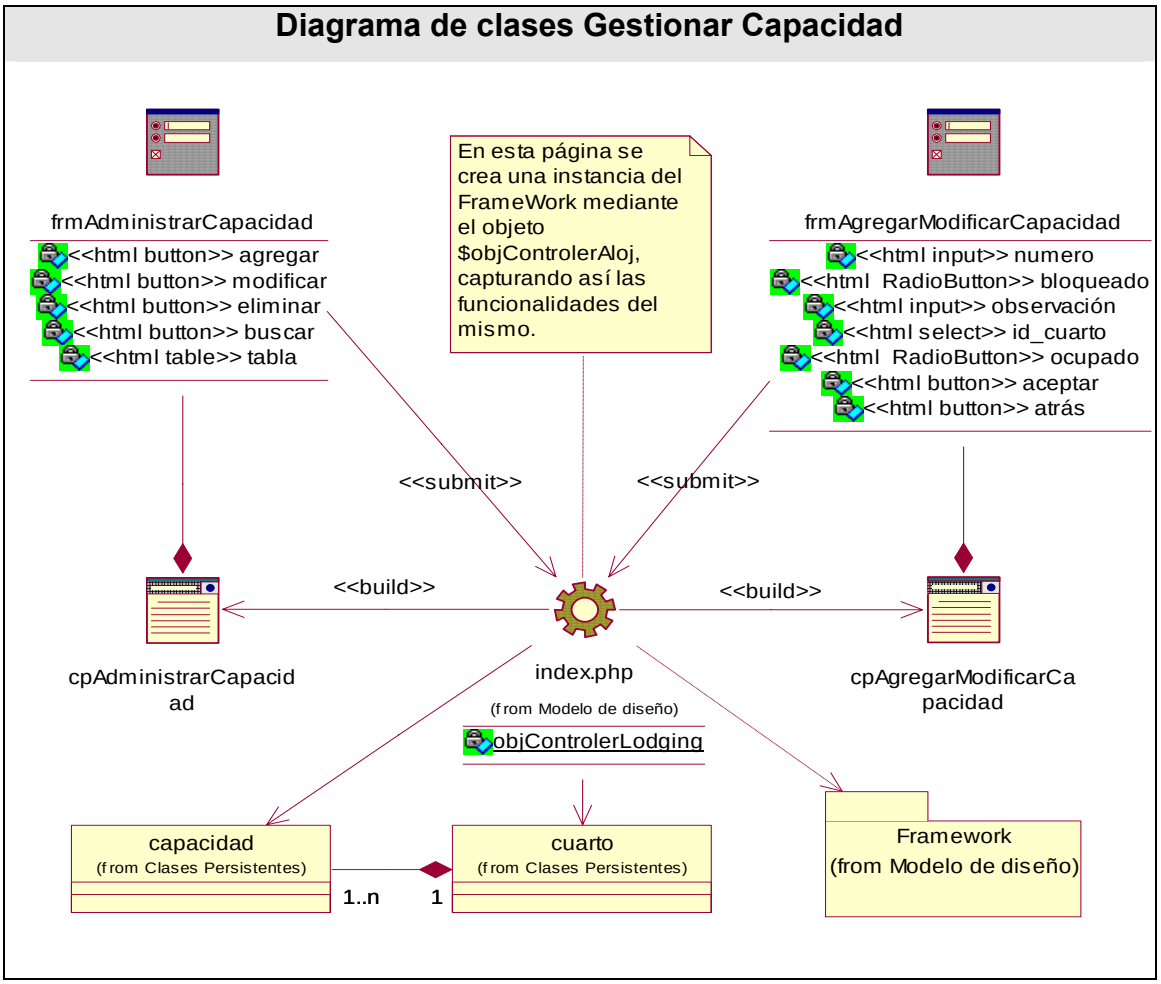


Figura 4.10 Diagrama de clases Gestionar Capacidad.

4.3.2 Paquete "Gestionar Ubicación"

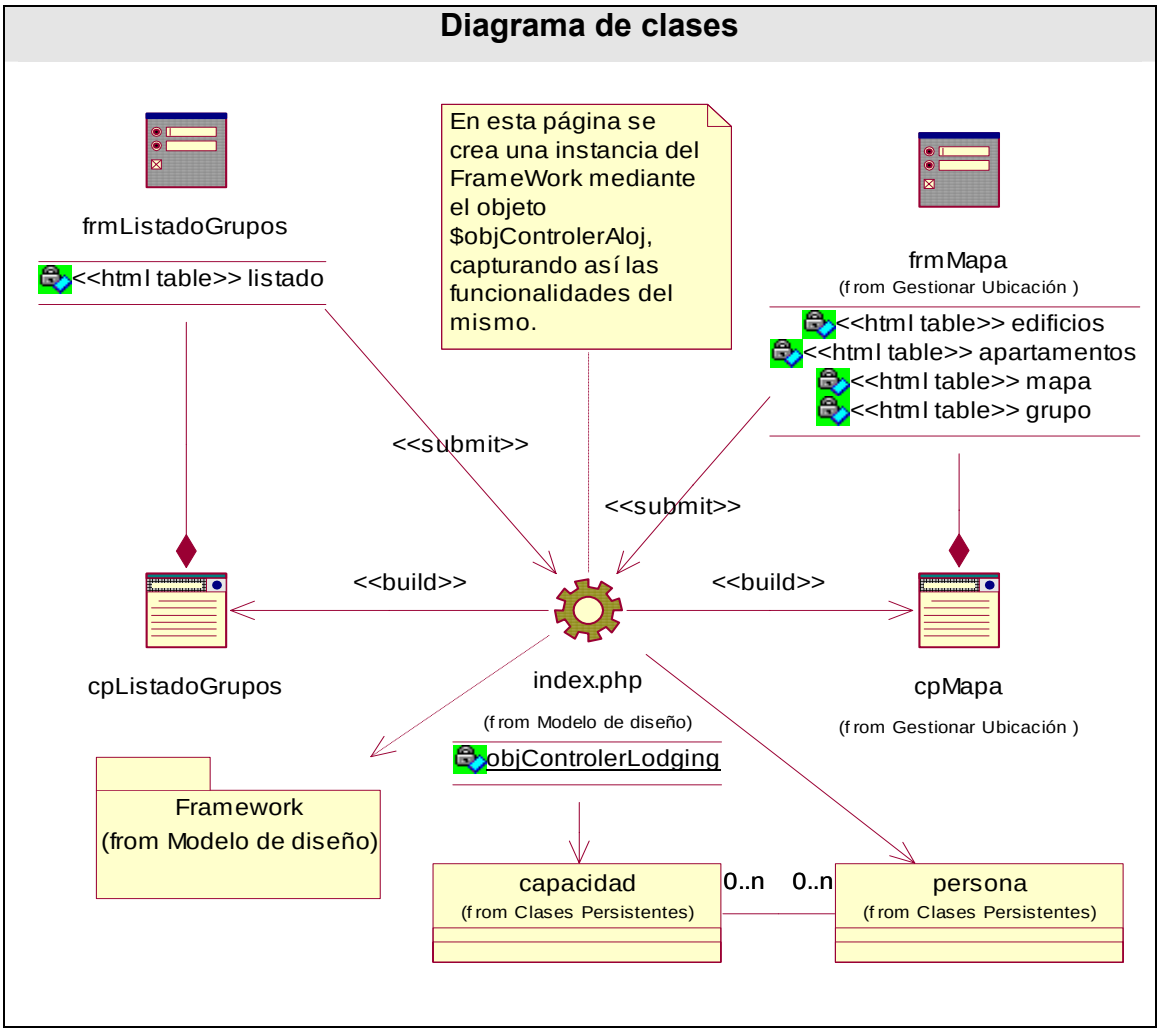


Figura 4.11 Diagrama de clases Gestionar Ubicación para hospitalizados.

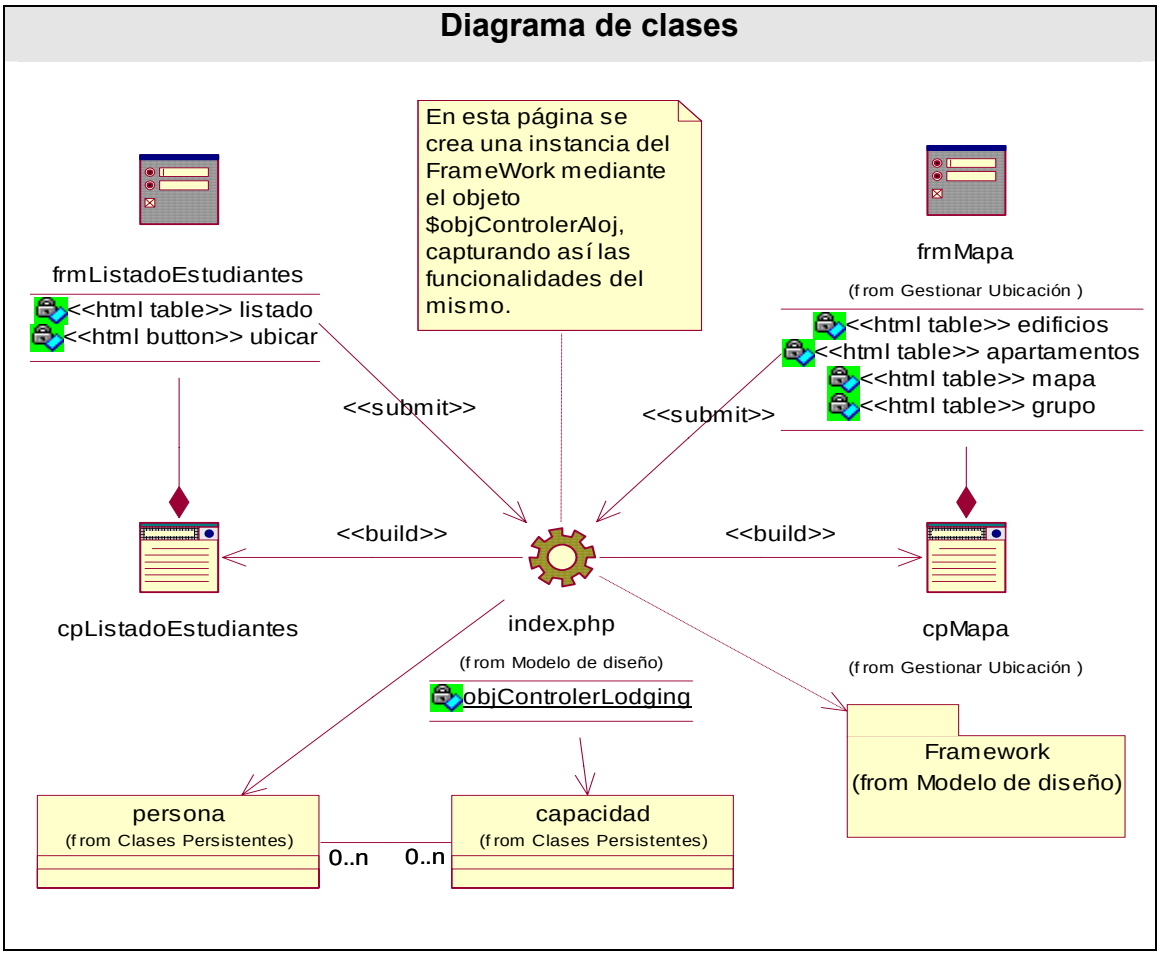


Figura 4.12 Diagrama de clases Gestionar Ubicación para estudiantes.

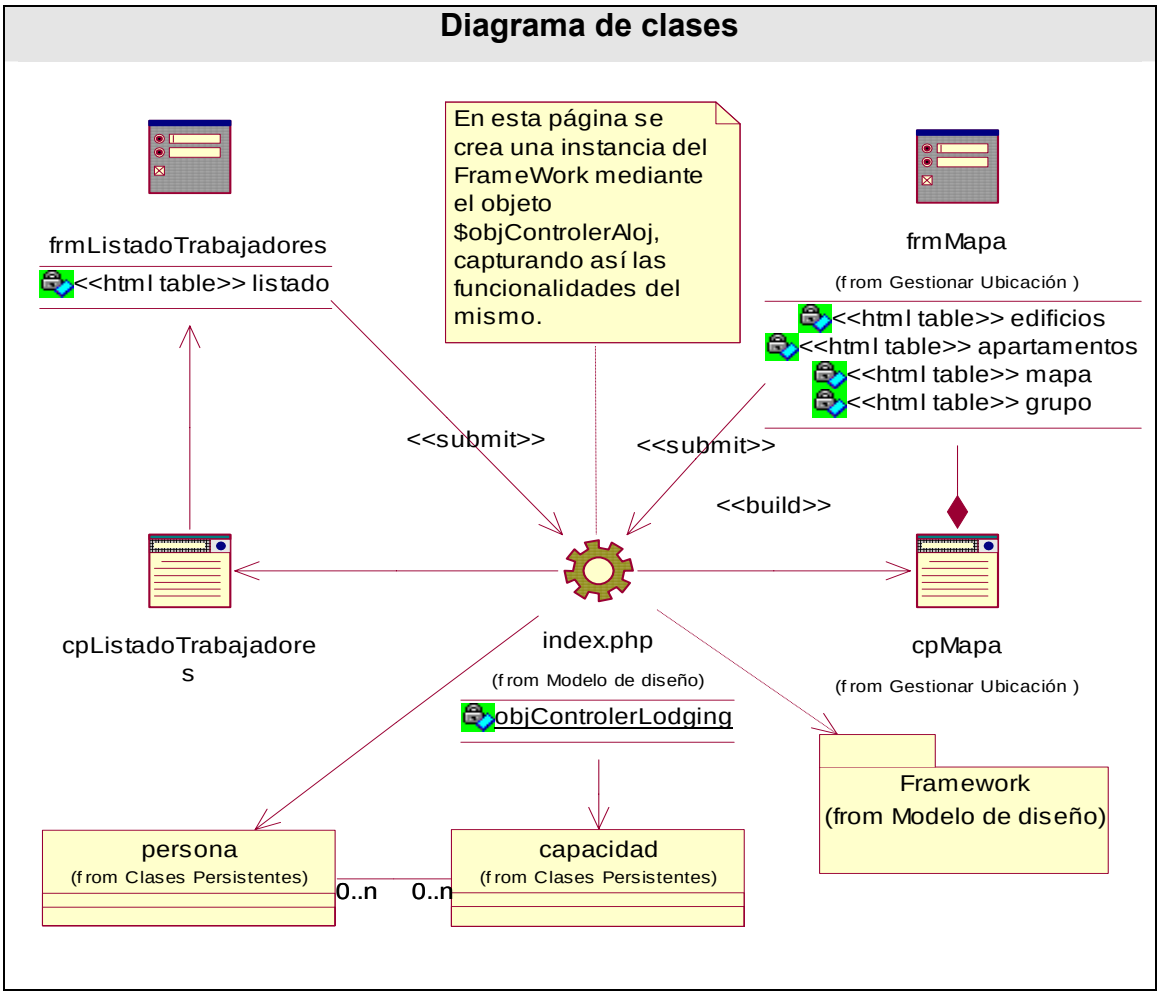


Figura 4.13 Diagrama de clases Gestionar Ubicación para trabajadores.

Para el caso de uso “Liberar Capacidad” se propone como solución la implementación de un trigger en el servidor de datos que permita actualizar el estado de las capacidades una vez que se detecte la salida del centro de una persona, en caso de que haya sido alojada.

4.4 Principios de diseño.

El diseño de la interfaz de una aplicación, el formato de los reportes, la concepción de la ayuda y el tratamiento de excepciones tiene gran influencia en el éxito o fracaso de una aplicación. A continuación se describen los principios de diseño seguidos para el desarrollo del sistema en cuestión.

4.4.1 Interfaz de usuario.

El diseño de interfaces de usuario es una tarea que ha adquirido relevancia en el desarrollo de un sistema, se puede definir como: “el conjunto de trabajos y pasos que seguirá el usuario, durante todo el tiempo que se relacione con el programa, detallando lo que verá y escuchará en cada momento, y las acciones que realizará, así como las respuestas que el sistema dará”. [11]

La calidad de la interfaz de usuario puede ser uno de los motivos que conduzca a un sistema al éxito o al fracaso, es por eso que uno de los aspectos más relevantes de la usabilidad de un sistema es la consistencia de su interfaz de usuario.

Para el desarrollo de la interfaz se tuvo en cuenta los siguientes aspectos:

1). Reducir la carga a la memoria.
2). Atajos a Usuarios expertos.
3). Obtener información de retroalimentación.
4). Diseño de diálogos que conducen a una conclusión.
5). Previsión de errores y manejo de errores simples.
6). Lograr deshacer acciones fácilmente.
7). Que el usuario sintiera la sensación de control.

Una de las premisas fundamentales de la aplicación es la ventaja que proporcionan las interfaces Web sobre las interfaces de comando. Ya que las interfaces Web:

- Proporcionan un ambiente amigable.
- Conducen a un aprendizaje más natural.
- Establecen un “sentimiento” (sobre todo en la uniformidad del ambiente) al usuario que enriquece su experiencia en el uso de la aplicación.
- Además de estos principios, se tuvieron en cuenta las siguientes características:
- Utilizar una misma tipografía, forma y estilo en todas las páginas.
- La facilidad del usuario de poder navegar desde cualquier punto a otro dentro de la aplicación.
- Se tuvo presente siempre el ancho de banda y por ello se utilizaron formato de imágenes de compresión favorables.

- La simplicidad y consistencia, favoreciendo la usabilidad de la aplicación.
- Navegación simple en todas las páginas de la aplicación, de forma tal que siempre sea accesible por el usuario.
- Estabilidad y uniformidad del diseño, para así poder ubicar al usuario dentro del mismo y hacerlo sentir parte de él.

Se utilizó una hoja de estilos para guardar la configuración del diseño para todas las páginas, para los botones y las líneas se utilizaron estos estilos, eliminando así el número de imágenes que demoren la presentación de la página.

Los formularios de entradas ocupan el centro superior y las entradas organizadas por importancia. Se incluye una breve explicación del objetivo del formulario, y alguna especificación con respecto a las entradas.

Se realizan múltiples operaciones en cada página, de forma que el usuario no tenga que moverse tanto dentro de la aplicación, para completar una operación.

4.4.2 Formato de salida de los reportes.

Generar reportes que permitan un control de la información que fluye en el Hospital UCI durante la Misión Milagro es una de las principales funcionalidades del sistema propuesto. Estos se obtienen, en dependencia de las necesidades del usuario.

Los informes, resumen de resultados de la misión, se han concebido en ventanas diferentes a la aplicación, utilizando letra legible y colores claros, de fondo, para no recargar la página y lograr calidad y nitidez en la impresión de la información. Cada reporte e informe tiene un encabezado que le identifica y describe brevemente, luego se muestra la información obtenida de manera legible y organizada.

4.4.3 Ayuda.

La ayuda está accesible como parte del menú en todas las páginas de la aplicación, y con el fin de que el usuario vea solo la información que necesita en ese momento, cada

página muestra como realizar solo aquellas operaciones que se estén realizando en el momento, además se aportan los conceptos que se manejan en la aplicación, para que el usuario se familiarice con algunas entradas.

La ayuda constará en gran parte de la explicación funcional del sistema aunque abarcará algunos temas teóricos para mayor comprensión. Esto tiene el objetivo de que el usuario no solo tenga la explicación funcional del sistema sino que también pueda entender en que consiste el mismo y tenga mayor información en caso de decidir posteriormente en su mantenimiento.

4.4.4 Tratamiento de errores.

En el sistema propuesto se evitan, minimizan y tratan los posibles errores, con el fin de garantizar la integridad y confiabilidad de la información que en este se registra y muestra. Los errores se tratan en una página especial que incluye el fichero de configuración general, y está preparada para recoger el número del error y presentar la pantalla con el error que le corresponde a ese código.

Los mensajes de error que emite el sistema se muestran en un lenguaje de fácil comprensión para los usuarios. Cuando se introduce información en un formulario y faltan datos, sale un cuadro de alerta indicando el campo o dato que falta. Similar ocurre cuando se introduce información errónea en un campo numérico, e-mail o moneda. *Ver Anexos 19 y 20*.

4.5 Diseño de la base de datos.

La base de datos es el sistema utilizado para el almacenamiento de datos y acceso controlado a los datos almacenados. En este epígrafe se muestra el diseño de la base de datos del sistema propuesto a través del diagrama de clases persistentes y el esquema de la base de datos generados a partir de este, el modelo de datos.

4.5.1 Diagrama de clases persistentes.

Las clases persistentes son las clases que necesitan ser capaz de guardar su estado en un medio permanente, la necesidad de guardar su estado esta dado por al almacenamiento físico permanente de la información de la clase, para la copia de

seguridad en caso del fracaso del sistema, o para el intercambio de información. A continuación se muestra el diagrama de clases persistentes.

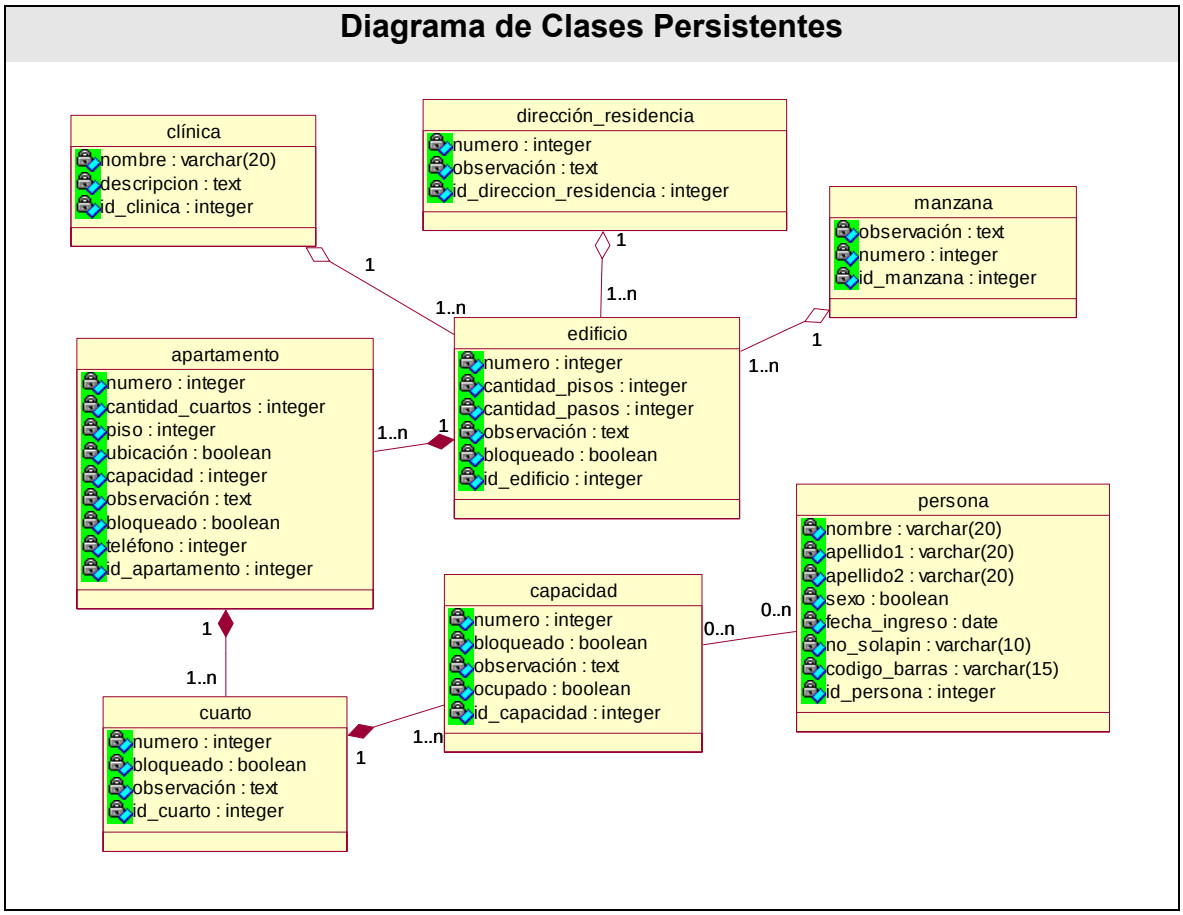


Figura 4.14 Diagrama de Clases Persistentes

4.5.2 Modelo de datos.

El modelo de los datos describe la representación lógica y física de datos persistentes en el sistema. A continuación se muestra el modelo de datos.

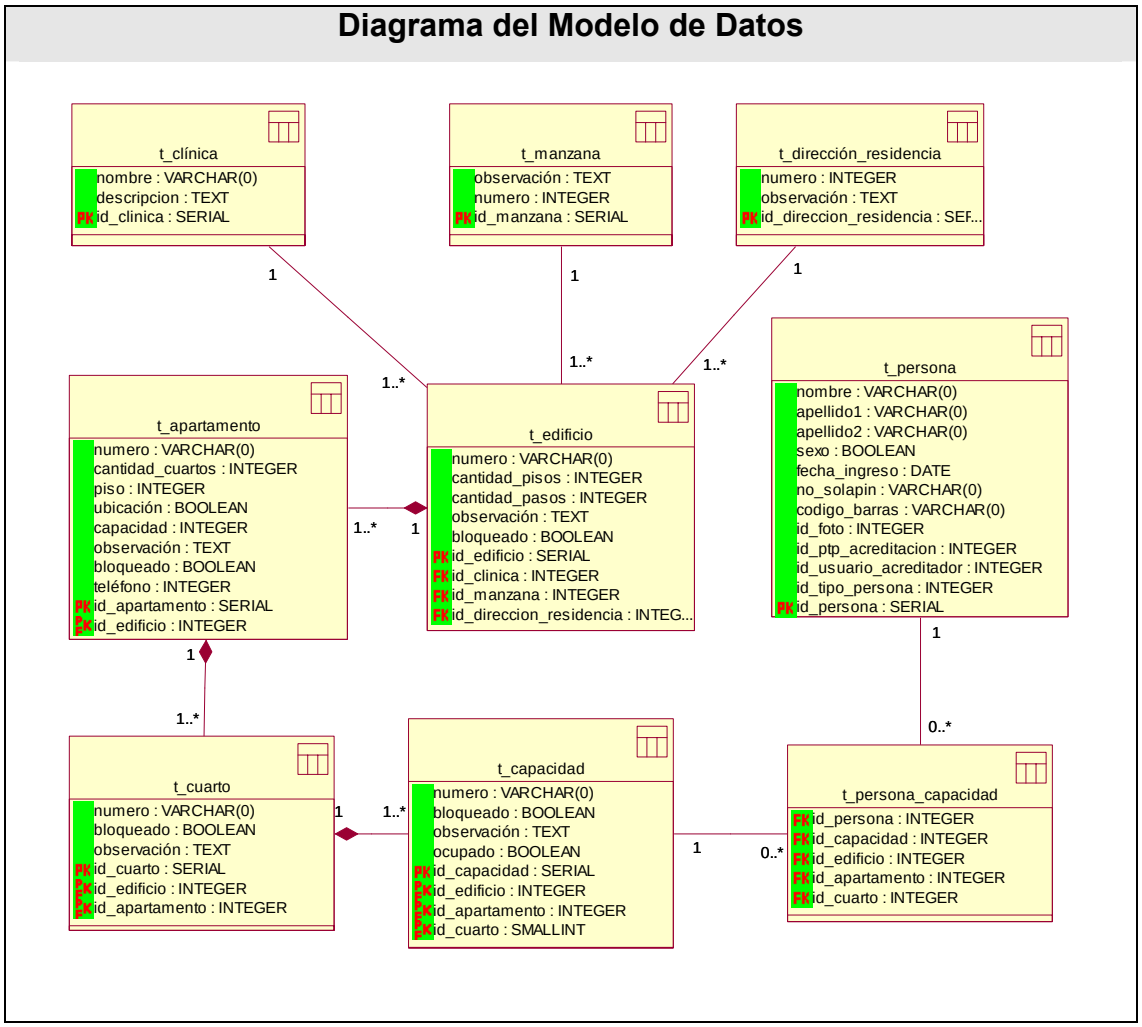


Figura 4.15 Diagrama del Modelo de Datos.

4.6 Diagrama de despliegue.

El diagrama de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. Es una colección de nodos y arcos; donde cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo de hardware similar. [14]

Muestra la configuración de los componentes hardware, los procesos, los elementos de procesamiento en tiempo de ejecución y los objetos que existen en tiempo de ejecución. En este tipo de diagramas intervienen nodos, asociaciones de comunicación, componentes dentro de los nodos y objetos que se encuentran a su vez dentro de los componentes. Un nodo es un objeto físico en tiempo de ejecución, es decir una

máquina que se compone habitualmente de, por lo menos, memoria y capacidad de procesamiento, a su vez puede estar formada por otros componentes. [6]

El diagrama de despliegue muestra la topología del hardware sobre el que se ejecuta el sistema. Ver Tabla 4.2 Diagrama de despliegue.

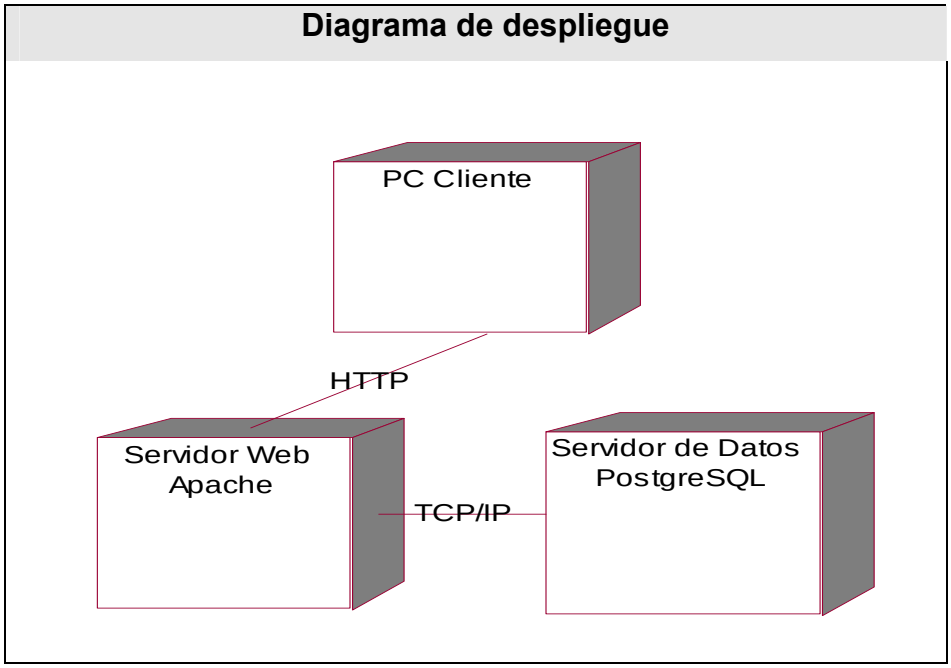


Figura 4.16 Diagrama de despliegue.

4.7 Conclusiones.

En el presente capítulo se desarrollaron los diagramas de clases de la aplicación y el diseño de la base de datos del sistema. Se describieron, además, los principios de diseño seguidos, específicamente, los temas de estándares de la interfaz, concepción del tratamiento de errores, sistema de ayuda y principios de codificación.

CAPÍTULO

Estudio de Factibilidad 5

5.1 Introducción.

Para la realización de un proyecto es de suma importancia el análisis del costo y los beneficios que reportará. Como resultado de este análisis se obtiene el tiempo de desarrollo en meses, costo y la cantidad de personas que se necesitan para desarrollar el proyecto.

En este capítulo se describe la estimación de costos del sistema propuesto y sus beneficios.

5.2 Planificación basada en casos de uso.

Paso 1. Cálculo de los Puntos de casos de uso Desajustados.

$$UUCP = UAW + UUCW$$

Donde:

UUCP: Puntos de casos de uso sin ajustar.

UAW: Factor de peso de los actores sin ajustar.

UUCW: Factor de peso de los casos de uso sin ajustar.

Tipo de actor	Descripción	Factor de peso	Actores	Total
Simple	Sistema con sistema a través de interfaz de programación.	1	0	0
Medio	Sistema con sistema mediante protocolo de interfaz basada en texto.	2	0	0
Complejo	Persona que interactúa con el sistema mediante interfaz gráfica.	3	3	9
	Total		3	9

Tabla 5.1 Factor de peso de los actores sin ajustar.

$$UAW = \sum cant\ \ actores * peso$$

$$UAW = 9$$

Tipo de CU	Descripción	Peso	Cantidad de CU	Total
Simple	El caso de uso tiene de 1 a 3 transacciones.	5	1	5
Medio	El caso de uso tiene de 4 a 7 transacciones.	10	8	80
Complejo	El caso de uso tiene más de 8 transacciones.	15	2	30
Total			11	115

Tabla 5.2 Factor de peso de los casos de uso sin ajustar.

$$UUCW = \sum cant\,CU * Peso$$

$$UUCW = 115$$

$$UUCP = 9 + 115$$

$$UUCP = 124$$

Paso 2. Cálculo de los Puntos de casos de uso ajustados.

$$UCP = UUCP * TCF * EF$$

Donde:

UCP: Puntos de casos de uso ajustados.

UUCP: Puntos de casos de uso sin ajustar.

TCF: Factor de complejidad técnica.

EF: Factor de ambiente.

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

Factor	Descripción	Peso	Valor asignado	Total
T1	Sistema distribuido	2	0	0
T2	Tiempo de respuesta	1	4	4
T3	Eficiencia del usuario final	1	4	4
T4	Funcionamiento Interno complejo	1	2	2
T5	El código debe ser reutilizable	1	4	4
T6	Facilidad de instalación	0,5	3	1.5
T7	Facilidad de uso	0,5	5	2,5
T8	Portabilidad	2	4	8
T9	Facilidad de cambio	1	4	4
T10	Concurrencia	1	5	5
T11	Incluye objetivos especiales de seguridad	1	3	3
T12	Provee acceso directo a terceras partes	1	3	3
T13	Se requieren facilidades especiales de entrenamiento de usuarios	1	2	2
	Total			43

Tabla 5.3 Factor de complejidad técnica.

$$TCF = 0.6 + 0.01 * \sum (peso * valor\ asignado)$$

TCF = 0.6 + 0.01 * 43

TCF = 0.6 + 0.43

$TCF = 1.03$

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

Factor	Descripción	Peso	Valor asignado	Total

E1	Familiaridad con el modelo de proyecto utilizado	1,5	4	6
E2	Experiencia en la aplicación	0,5	4	2
E3	Experiencia en la orientación a objetivos.	1	4	4
E4	Capacidad del analista líder.	0,5	4	2
E5	Motivación.	1	5	5
E6	Estabilidad de requerimientos	2	4	8
E7	Personal Part–Time	-1	2	-2
E8	Dificultad del lenguaje de programación	-1	2	-2
	Total			23

Tabla 5.4 Factor de ambiente.

$$EF = 1.4 - 0.03 * \sum (peso * valor\ asignado)$$

EF = 1.4 – 0,03 * 23

EF = 1.4 – 0.69

$EF = 0.71$

UCP = UUCP * TCF * EF

UCP = 124 *1.03 * 0.71

$UCP = 90.6812$

Paso 3. Estimación de esfuerzo a través de los puntos de casos de uso.

$$E = UCP * CF$$

Donde:

E: Esfuerzo estimado en horas hombres.

UCP: Punto de casos de usos ajustados.

CF: Factor de conversión.

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

En este caso se puede decir que:

CF = 20 Horas-Hombre / Punto de Casos de uso.

$E = 90.6812 * 20$

$E = 1813.624$ Horas-Hombre

Paso 4. Calcular esfuerzo de todo el proyecto.

Actividad	Porcentaje %	Horas-Hombres
Análisis	10	453,406
Diseño	20	906,812
Implementación	40	1813,624
Pruebas	15	680,109
Sobrecarga (otras actividades)	15	680,109
Total	100	4534,06

Tabla 5.5 Esfuerzo del proyecto.

Si **E_T = 5183.2 horas-hombre** y se estima que cada mes tiene como promedio 192 horas laborables, eso daría un **E_T = 23,61489583 mes-hombre.**

Esto quiere decir que 1 persona puede realizar el problema analizado en 23 meses y medio aproximadamente.

-Costo del Proyecto.

Se asume como salario promedio mensual $50.00

CH: Cantidad de hombres.

Tiempo: Tiempo total del proyecto.

CH = 5 hombres

CHM = 5 * Salario Promedio

CHM = 250.00 $/mes

Costo = CHM * E_T / CH

Costo = 250.00 * 23,61489583 / 5

Costo = $1180.744792 ≈ $1180.74

Tiempo = E_T / CH

Tiempo = 23,61489583 / 5

Tiempo = 4,72297917 ≈ 4.72

De los resultados obtenidos se interpreta que con 5 hombres trabajando en el proyecto el mismo se desarrolla en 4,72 meses y su costo total se estima que sea $1180.74.

5.3 Beneficios tangibles e intangibles.

El Sistema Automatizado para la Gestión de Información de la Misión Milagro no es un producto con fines comerciales, su principal objetivo es resolver los problemas que existen durante el desarrollo de esta tarea en el hospital UCI.

El beneficio fundamental del sistema es contar con una aplicación Web flexible, dinámica y de interfaz agradable que le permita registrar, actualizar y conocer de una forma más precisa y en el menor tiempo posible datos de interés de los participantes en esta actividad.

Por tanto, los beneficios inmediatos son generalmente intangibles:

- Disminución del tiempo y esfuerzo que se invierte en esta tarea que se realiza, hasta ahora, de forma manual.
- Disminución de la acumulación de materiales impresos relacionados con los procesos de alojamiento.
- Disminución de los gastos pues resulta menos costoso crear y procesar información digital que copias duras.
- Fácil detección de problemas.
- Fácil y rápido acceso y publicación de la información actualizada.
- Fácil procesamiento de la información y obtención, dinámica, de reportes de la situación de la misión en cualquier momento.

5.4 Análisis de costos y beneficios.

Desarrollar un producto informático cuesta. Justificar entonces su desarrollo depende de los beneficios que reportarían su implantación y utilización. Los beneficios pueden ser económicos y de orden social, estos últimos son de tanta importancia como los

primeros. El sistema que se propone está dirigido fundamentalmente a la salud, por tanto su mayor beneficio es de orden social.

Una vez implantado el sistema éste contribuirá a aumentar la eficiencia de los servicios y recursos que se brindan en el Hospital UCI durante la Misión Milagro, al disminuir el tiempo necesario a emplear en el registro, consulta y actualización de la compleja y diversa información; y generar informes de resultados de los procesos que se desarrollan con mayor rapidez y certeza.

La tecnología utilizada para el desarrollo del sistema es totalmente libre, por tanto no hay que incurrir en gastos en el pago de licencias de uso. El sistema es portable por lo que un cambio de plataforma para la implantación del mismo es viable y factible, y no hay que incurrir en muchos cambios; debido a la estructuración en capas de los procesos del negocio que se diseñaron.

Analizando el costo del proyecto, los numerosos beneficios que reporta, detallados con anterioridad, se puede concluir que su implementación es realmente factible.

5.5 Conclusiones.

En este capítulo se describió el estudio de factibilidad realizado correspondiente al sistema propuesto, teniendo en cuenta el costo estimado y los beneficios que reportará al ser implantado.

La herramienta propuesta reportará beneficios significativos e importantes para el desarrollo de la Misión Milagro en la UCI, al contribuir a mejorar todos los servicios y procesos que se realizan aquí, lo que indica que es factible implementar la herramienta propuesta.

CONCLUSIONES

Llegado este punto se espera que el documento haya servido para la comprensión teórica de la situación problémica existente y su solución, así como el desarrollo de las diferentes etapas de la aplicación usando la metodología RUP.

Se alcanzó, satisfactoriamente, el objetivo propuesto: desarrollar una solución robusta, flexible y única de software que dé soporte a los procesos de alojamiento durante la Misión Milagro; reafirmando así la utilidad y validez de emplear las tecnologías informáticas para apoyar la labores que se desarrollan en cualquier tipo de esfera. Se obtuvieron además los siguientes resultados:

- Se ha demostrado la eficacia de los lenguajes y tecnologías utilizadas para el desarrollo del sistema.

- Se realizó una base de datos, donde se almacena toda la información necesaria que se genera de los procesos de alojamiento, para de esta forma garantizar la veracidad y centralización de la misma.

- Se realizó el análisis, diseño e implementación del sistema.

- La solución propuesta ha sido acertada, los requerimientos soportan al sistema y los casos de uso satisfacen las necesidades funcionales.

- Se han seguido los principios básicos de diseño descritos para el desarrollo del sistema.

- Se logra una seguridad y protección de los datos consecuente con el nivel de seguridad requerido.

RECOMENDACIONES

Se recomienda:

- Poner a prueba el sistema durante un período de tiempo significativo, para comprobar su desempeño y que las funcionalidades del sistema se correspondan con la actividad que se está gestionando.
- Continuar el estudio con el objetivo de añadir nuevas funcionalidades.
- Proponer, tras corroborar un desempeño exitoso, la utilización y generalización de este sistema en los diferentes lugares que se lleva a cabo la Misión Milagro en nuestro país.

REFERENCIAS BIBLIOGRÁFICAS

[1] Izarra, Andrés. *Misión Milagro: Convenio Solidario*, Octubre 2004. Disponible en: <http://www.gobiernoenlinea.ve/docMgr/sharedfiles/Folleto_Mision_Milagro.pdf>. [Fecha de consulta 15 marzo 2006].

[2] Serrano, Pascual. *Infiltrado en un avión de la Misión Milagro con destino a La Habana*, febrero 2006. Disponible en: <http://www.rebelion.org/noticia.php?id=26464>. [Fecha de consulta 15 marzo 2006].

[3] *Misión Milagro: Solidaridad de Cuba y Venezuela con los desposeídos de América Latina,* febrero 2006. Disponible en: <http://www.fmln.org.sv/portal/modules.php?op=modload&name=News&file=article&sid=179>. [Fecha de consulta 15 marzo 2006].

[4] Valerino, María. *Misión Milagro, Dioses de blanco*, febrero 2006. Disponible en: <http://www.lademajagua.co.cu/infgran3941.htm>. [Fecha de consulta 15 marzo 2006].

[5] *Programación Web*. [Disponible en: <http://www.arsys.es/soporte/programacion/windows.htm > [Fecha de consulta 20 marzo 2006].

[6] Valido, Y. y Moreira, Y. *SAIMM: Sistema de Apoyo Integral a la Misión Milagro*. Trabajo de Diploma para optar por el título de Ingeniero Informático, Instituto Superior Politécnico "José Antonio Echeverría", Ciudad de la Habana, junio 2005.

[7] Méndez, L. y Torres, A. *Sistema de Promoción y Gestión Comercial para la oficina de Transferencia Tecnológica de la Universidad de Cienfuegos*. Trabajo de Diploma para optar por el título de Ingeniero Informático, Instituto Superior Politécnico "José Antonio Echeverría", Ciudad de la Habana, junio 2005.

[8] Hernández, J. y Sáez, L. *SRM: Sistema del Registro Mercantil*. Trabajo de Diploma para optar por el título de Ingeniero Informático, Instituto Superior Politécnico "José Antonio Echeverría", Ciudad de la Habana, junio 2004.

[9] Mandrake, *Revisión Rápida de PHP5 integrado con Zend*, septiembre 2004. Disponible en: <http://www.venezolano.web.ve/archives/230-Revision-rapida-de-PHP5-integrado-con-Zend.html>. [Fecha de consulta 27 marzo 2006].

[10 Cantero, J. *Un vistazo a PHP5 [I],* julio 2004. Disponible en:<http://libertonia.escomposlinux.org/story/2004/7/15/115328/134>. [Fecha de consulta 29 marzo 2006].

[11] Parra, A. y Matos, M. *Sistema Automatizado para la Gestión de Información de la Unión de Jóvenes Comunistas*. Trabajo de Diploma para optar por el título de Ingeniero Informático, Instituto Superior Politécnico "José Antonio Echeverría", Ciudad de la Habana, junio 2005.

[12] [Disponible en: < http://milagro.prod.uci.cu:5901/sitios/PostgreSQL > [Fecha de consulta 30 marzo 2006].

[13] *Macromedia Dreamweaver MX 2004*. Getting Started. Ayuda. Macromedia, Inc. 2003. [Fecha de consulta 22 marzo 2006].

[14] Jacobson, I.; Booch, G. y Rumbaugh, J. *El Proceso Unificado de Desarrollo de software*. Addison-Wesley. 2000.

[15] *OMG Unified Modeling Language Specification*. OMG, INc. 2003.

[16] Schmuller, J. *Aprendiendo UML en 24 horas*. Prentice Hall.

[17] Pérez, Y. y Sánchez, Y. *Registro de Partos y Nacimientos para el Sistema Integral de Salud*. Trabajo de Diploma para optar por el título de Ingeniero Informático, Instituto Superior Politécnico "José Antonio Echeverría", Ciudad de la Habana, junio 2005.

[18] Castellanos, Y. *Portal de las Misiones Sociales de la República Bolivariana de Venezuela*. Trabajo de Diploma para optar por el título de Ingeniero Informático, Instituto Superior Politécnico "José Antonio Echeverría", Ciudad de la Habana, junio 2005.

[19] *Patrones de Diseño*. Conferencia 7 de *Ingienería del Software I*, curso 2005-2006, UCI.

[20] [Disponible en: <http://milagro.prod.uci.cu:5901/documentacion/Otros/MVC/> [Fecha de consulta 24 mayo 2006].

[21] *AJAX*. [Disponible en: <http://es.wikipedia.org/wiki/AJAX> [Fecha de consulta 31 mayo 2006].

[22] Welicki, L. *Patrones y Antipatrones: una Introducción –Parte II.* Disponible en:<http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/MTJ_3317.asp>. [Fecha de consulta 2 junio 2006].

BIBLIOGRAFÍA

- *AJAX un nuevo acercamiento a aplicaciones Web*, mayo 28 del 2005. Disponible en: <http://www.uberbin.net/archivos/internet/ajax-un-nuevo-acercamiento-a-aplicaciones-web.php> [Fecha de consulta 31 mayo 2006].
- *Clases de Ingienería del Software I*, curso 2005-2006, UCI.
- Hernández, Rolando A. y Coello, Sayda. *El Paradigma Cuantitativo de la Investigación Científica*. Noviembre 2002, UCI.
- *Introducción a php*. Disponible en: <www.ciberteca.net/webmaster/php> [Fecha de consulta 24 marzo 2006].
- Larman, Craig. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* Primera Edición por Prentice Hall, Hispanoamericana S.A. 1999.
- Larman, Craig. *UML y Patrones. Introducción al análisis y diseño orientado a objetos y al proceso unificado.* Segunda Edición por Prentice Hall.
- MARRERO, D. *Modelado de aplicaciones Web con UML.* En: Conferencia de Ingeniería de Software, Diciembre 2002, ISPJAE (CEIS).
- Matos, Rosa María. *Introducción al trabajo con Base de Datos*. Asignatura de Sistemas de Gestión de Base de Datos.
- *PostgreSQL 8.1.x*. Disponible en: <http://www.postgresql.cl/> [Fecha de consulta 26 marzo 2006].
- Peralta, Mario. *Estimación del esfuerzo basada en casos de uso*. Centro de Ingienería del Software e Ingienería del Conocimiento, Buenos Aires, Argentina.
- Quatrani, Terry. *Visual Modeling with Rational Rose 2000 and UML,* Publisher Addison Wesley, Second Edition October 19, 1999
- *Tutorial de PostgreSQL*. Disponible en: <http://es.tldp.org/Postgresql-es/web/navegable/tutorial/tutorial.html> [Fecha de consulta 26 marzo 2006].

GLOSARIO DE TÉRMINOS Y SIGLAS

- ALBA: *Alternativa Bolivariana para las Américas*.
- Administrador: es la persona que tiene privilegios para determinadas funcionalidades del sistema.
- APACHE: es un servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etcétera), Windows y otras, que implementa el protocolo HTTP/1.1.
- AJAX: Asynchronous JavaScript And XML.
- 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). Con ASP se pueden combinar páginas HTML, *scripts* y objetos COM. Con el objetivo de crear aplicaciones potentes. Se caracterizan por su fácil desarrollo y mantenimiento.
- Arquitectura Cliente/Servidor: es un modelo para el desarrollo de sistemas de información, en el que las transacciones se dividen en elementos independientes que cooperan entre sí para intercambiar información, servicios o recursos.
- CASE: *Computer Aided Software Engineering.*
- CGI: *Common Gateway Interface.*
- CEIS: *Centro de Estudio de Ingeniería de Sistemas*.
- COCOMO: Modelo para la estimación de costos de productos informáticos.
- CUJAE: *Ciudad Universitaria José Antonio Echeverría*.
- CUN: *Caso de uso del negocio*.
- CUS: *Caso de uso del sistema*.
- DHTML: *Dynamic HTML.*
- HTML: *HyperText Markup Language.* Lenguaje usado para escribir documentos para servidores World Wide Web. Es una aplicación de la ISO Standard 8879:1986. Es un lenguaje de marcas. Los lenguajes de marcas no son equivalentes a los lenguajes de programación aunque se definan igualmente como "lenguajes". Son sistemas complejos de descripción de información, normalmente documentos, que se pueden controlar desde cualquier editor ASCII.

- HTTP: *HyperText Transfer Protocol.* Protocolo de Transferencia de Hipertextos. Modo de comunicación para solicitar páginas Web.
- Herramientas CASE: Herramientas utilizadas para el desarrollo de proyectos de Ingeniería de Software.
- Hardware: Componentes electrónicos, tarjetas, periféricos y equipo que conforman un sistema de computación; se distinguen de los programas (software) porque son tangibles.
- Internet: Sistema de redes de computación ligadas entre sí, con alcance mundial, que facilita servicios de comunicación de datos como registro remoto, transferencia de archivos, correo electrónico y grupos de noticias. Internet es una forma de conectar las redes de computación existentes que amplía en gran medida el alcance de cada sistema participante.
- JSP: *Java Server Pages*. Es la tecnología para generar páginas web de forma dinámica en el servidor, desarrollado por Sun Microsystems, basado en scripts que utilizan una variante del lenguaje java. La tecnología JSP, o de JavaServer Pages, es una tecnología Java que permite a los programadores generar dinámicamente HTML, XML o algún otro tipo de página web. Esta tecnología permite al código Java y a algunas acciones predefinidas ser embebidas en el contenido estático.
- Linux: Es el nombre de un núcleo, pero se suele denominar con este nombre a un sistema operativo de libre distribución software libre (y de código abierto), donde el código fuente está disponible públicamente y cualquier persona, con los conocimientos informáticos adecuados, puede libremente estudiarlo, usarlo, modificarlo y redistribuirlo.
- Macromedia Dreamweaver MX: Herramienta para el desarrollo de aplicaciones Web de Macromedia. Combina en un único entorno de desarrollo accesible y potente las reconocidas herramientas de presentación visual de Dreamweaver, las características de rápido desarrollo de aplicaciones Web de Dreamweaver UltraDev y ColdFusion Studio, y el extenso soporte de edición de código de HomeSite. Ofrece una completa solución abierta para las tecnologías Web y estándares de hoy, incluyendo la accesibilidad y servicios Web.

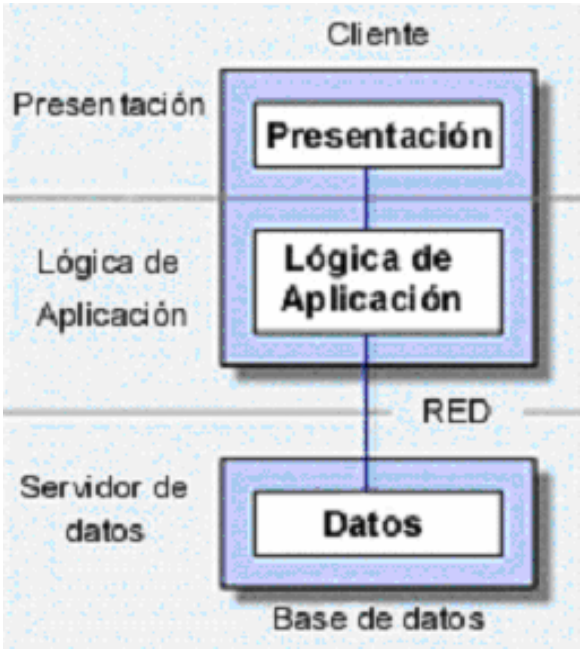
- Microsoft: Compañía que manufactura los sistemas de operación DOS y Windows.
- MySQL: Es un sistema de gestión de bases de datos relacional que cuentan con todas las características de un motor de BD comercial: transacciones atómicas, triggers, replicación, llaves foráneas entre otras. Su ingeniosa arquitectura lo hace extremadamente rápido y fácil de personalizar.
- MVC: *Modelo Vista Controlador.*
- PC : *Personal Computer.*
- PHP: *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.
- PostgreSQL: es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS) libre.
- Perl: *Practical Extraction and Report Language.* Es un lenguaje de programación desarrollado por Larry Wall inspirado en otras herramientas de UNIX como son: sed, grep, awk, c-shell.
- Personal que trabaja en la misión: incluye al personal UCI, personal externo y el personal médico.
- Personal UCI: son los profesores, estudiantes y trabajadores de la UCI.
- Personal externo: son personas que no trabajan en la UCI y vienen a brindar cualquier tipo de servicio.
- Personal médico: son los médicos, enfermeras y técnicos de salud que participan en la misión.
- PDO: *PHP Data Objects.*
- RUP: *Rational Unified Process* (Proceso Unificado de desarrollo). Metodología para el desarrollo de Software.
- Software: Programas de sistema, utilerías o aplicaciones expresados en un lenguaje de máquina.
- 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.

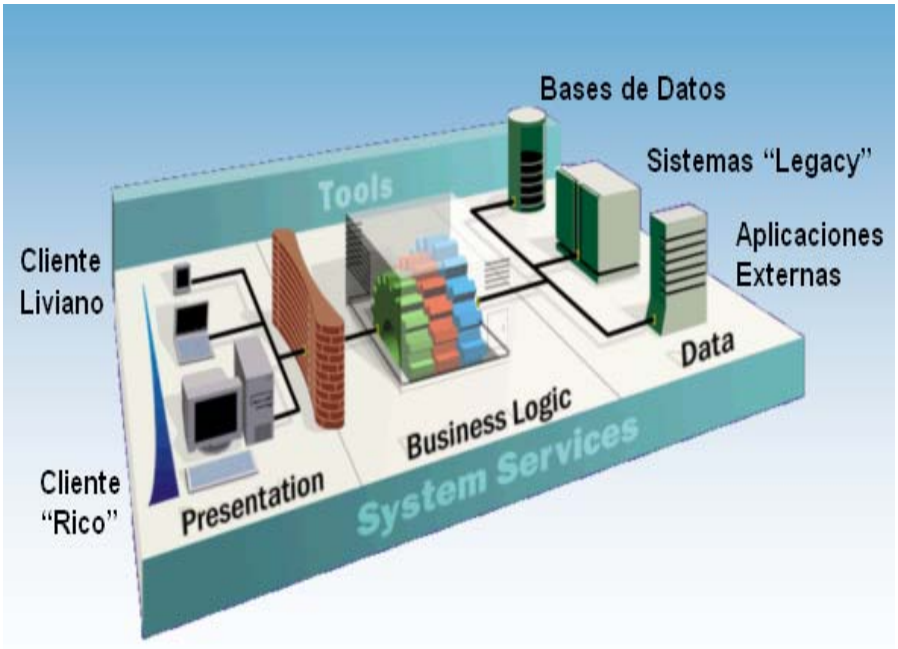
- Sitio Web: Es un conjunto de páginas web, típicamente comunes a un dominio de Internet o subdominio en la World Wide Web en Internet.
- SGBD: *Sistema de Gestión de Bases de Datos*. Es el software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez.
- SAIMM: *Sistema de Apoyo Integral a la Misión Milagro.*
- SAGIMM: *Sistema Automatizado para la Gestión de Información de la Misión Milagro*.
- UCI: *Universidad de las Ciencias Informáticas*.
- 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.
- WEB (WWW): Red de documentos HTML intercomunicados y distribuidos entre servidores del mundo entero.
- WML: *Website Meta Language*.
- 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.

ANEXOS

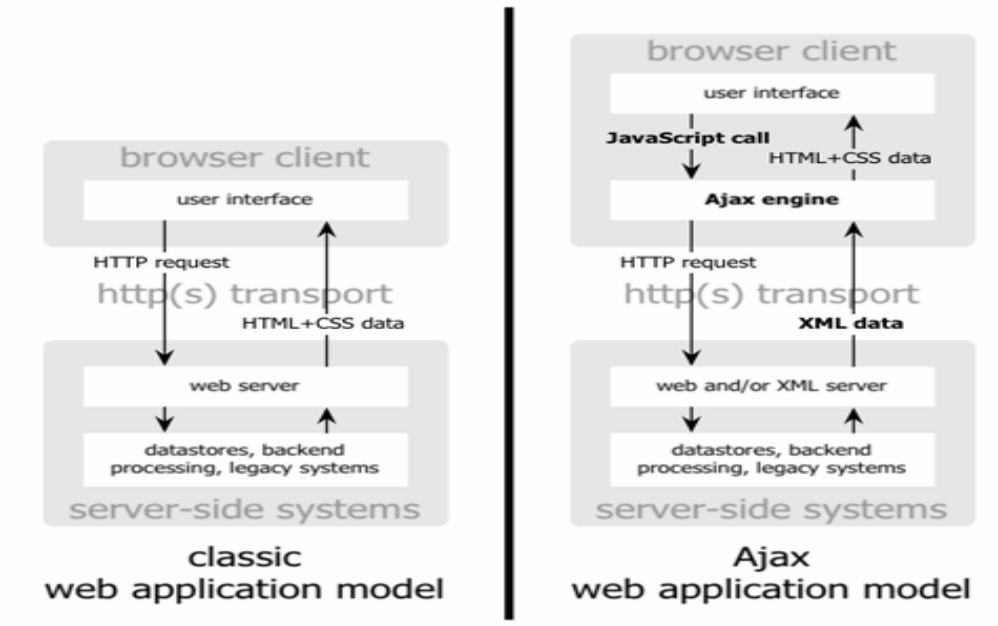
Anexo 1: Modelo Cliente – Servidor de dos capas.



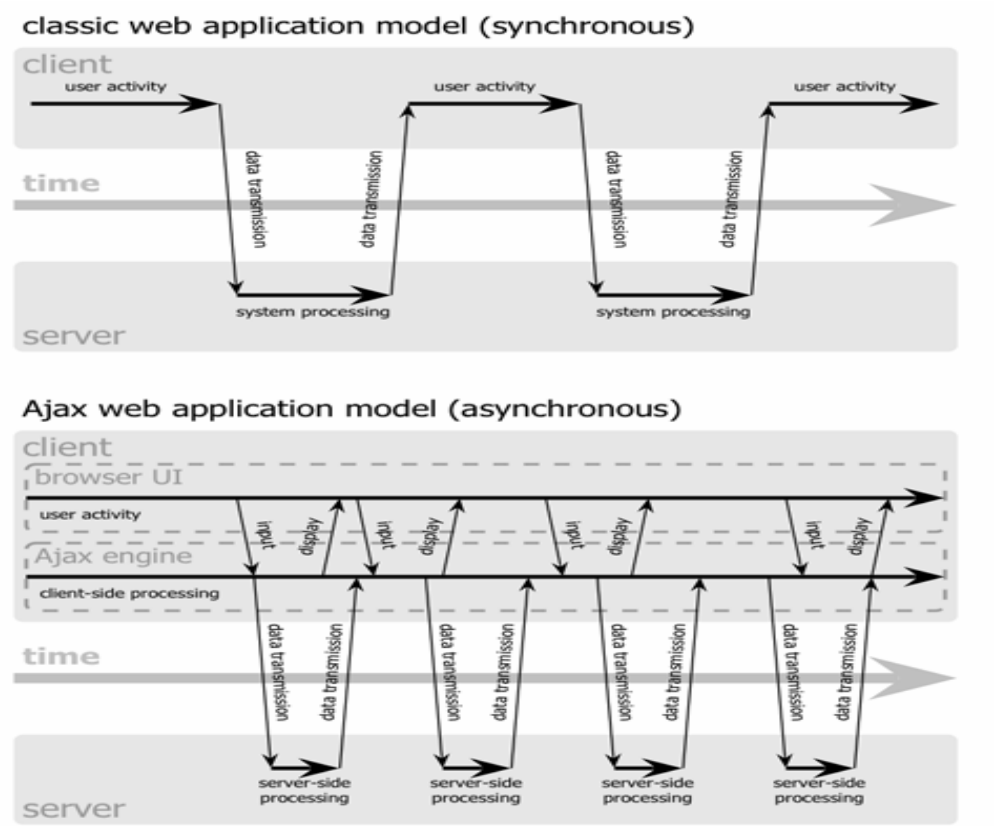
Anexo 2: Modelo Cliente – Servidor de tres capas.



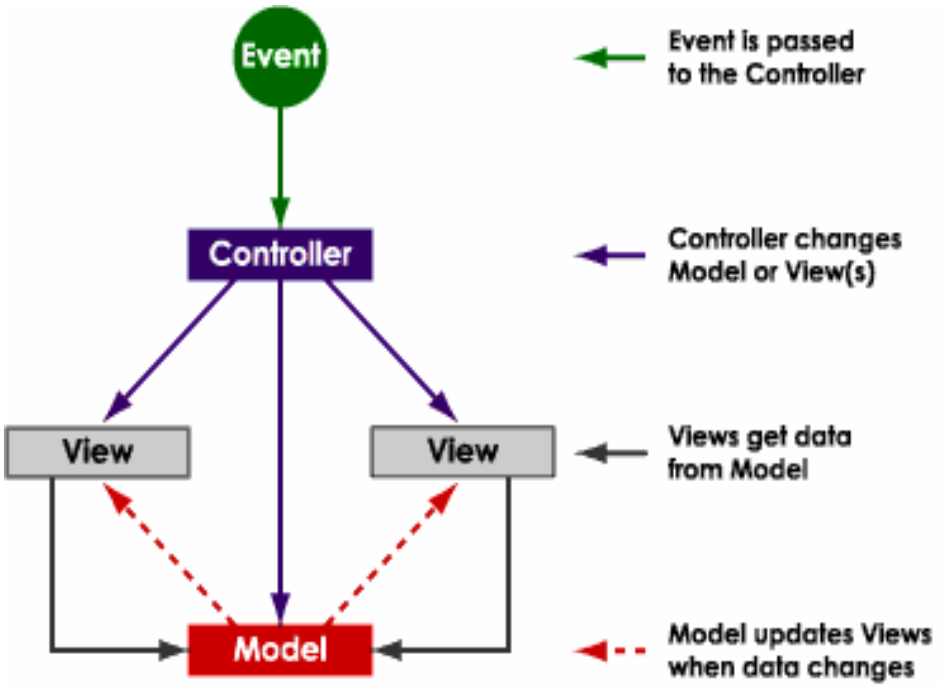
Anexo3: Se muestra el modelo tradicional para las aplicaciones Web (*a la izquierda*), comparado con el modelo de AJAX (*a la derecha*).



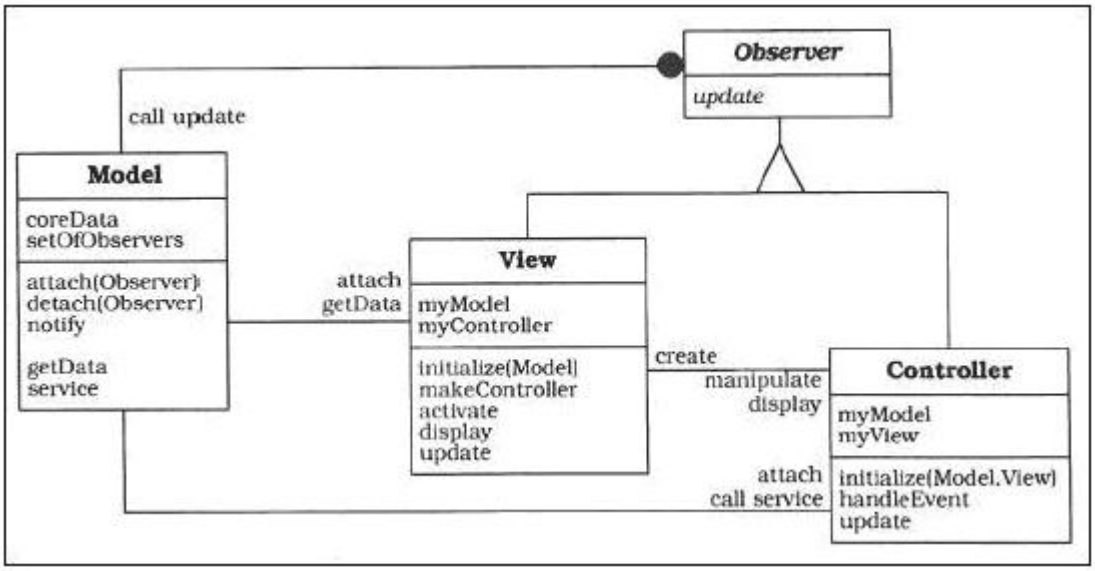
Anexo 4: Muestra el patrón de interacción sincrónica de una aplicación Web tradicional (*arriba*) comparada con el patrón asincrónico de una aplicación AJAX (*abajo*).



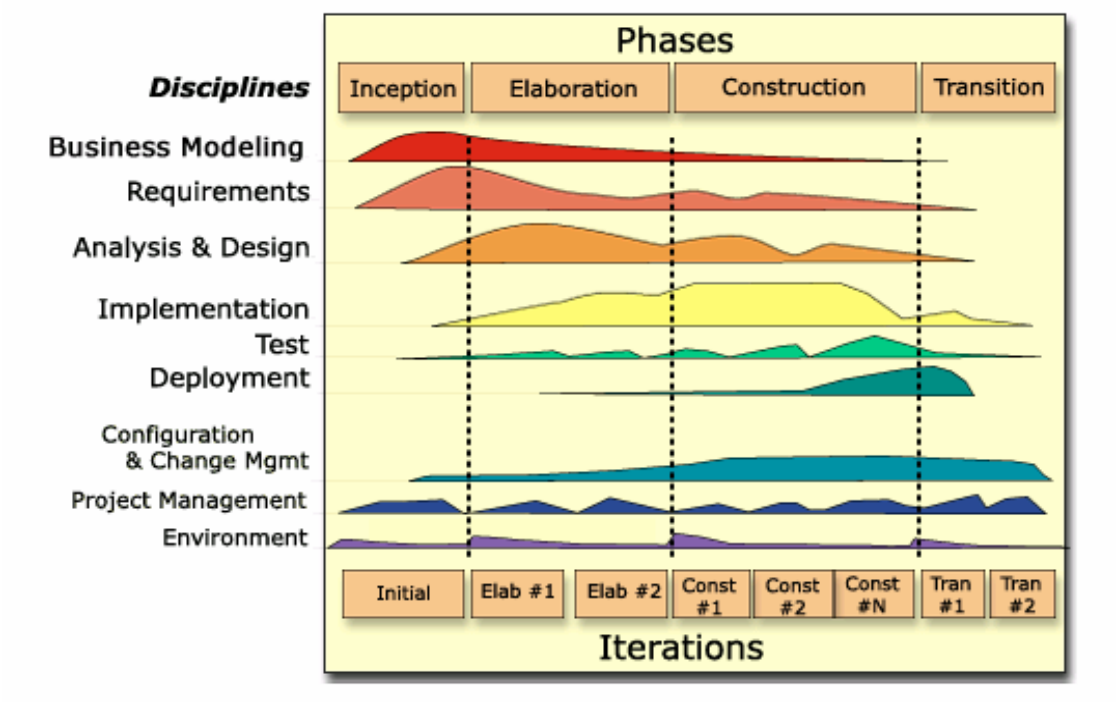
Anexo 5: Funcionamiento del patrón MVC.



Anexo 6: Estructura del patrón MVC.



Anexo 7: Flujos de trabajo de RUP.



Flujos de trabajo:

- **Modelamiento del negocio.**
- **Requerimientos.**
- **Análisis y diseño.**
- **Implementación.**
- **Prueba (Testeo).**
- **Instalación.**
- **Administración del proyecto.**
- **Administración de configuración y cambios.**
- **Ambiente.**

Anexo 8: Gestionar Apartamentos.

LISTADO DE APARTAMENTOS

Apartamento [-Seleccione-] Buscar

Agregar Modificar Eliminar

	Número	Cantidad de cuartos	Piso	Ubicación	Capacidad	Observación	Número_edificio	Bloqueado	Teléfono
○	9105	3	3	derecha	6		9	No	8974
○	9106	2	3	Izquierda	4		9	No	8975
○	9107	3	4	derecha	6		9	No	8976
○	9108	2	4	Izquierda	4		9	No	8977
○	9109	3	5	derecha	6		9	No	8978
○	9110	2	5	Izquierda	4		9	No	8979
○	9201	3	1	derecha	6		9	No	8980

AGREGAR APARTAMENTO

Número:

Cantidad de cuartos:

Pisos:

Ubicación: ○ Derecha ◉ Izquierda

Capacidad:

Observaciones:

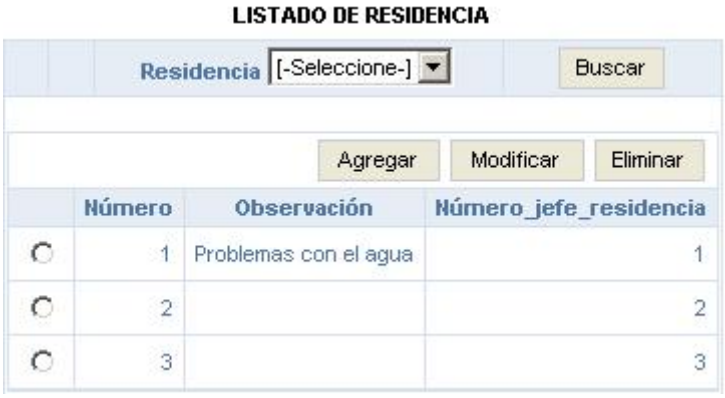
Edificio: 1

Bloqueado: ○ Sí ◉ No

Telefono:

Aceptar Cancelar

Anexo 9: Gestionar Residencias.



Anexo 10: Gestionar Capacidades.

LISTADO DE CAPACIDADES

Agregar | Modificar | Eliminar

	Número	Bloqueado	Observaciones	Número_cuarto	Ocupado
○	**101104-3B**	No		28647	No
○	**101106-1B**	No		28648	No
○	**101106-2A**	No		28649	No
○	**101106-2B**	No		28649	No
○	**101106-3A**	No		28650	No
○	**101106-3B**	No		28650	No
○	**101108-1A**	No		28651	No
○	**101108-1B**	No		28651	No
○	**101108-2A**	No		28652	No
○	**101108-2B**	No		28652	No

AGREGAR CAPACIDAD

Número:

Bloqueado: ○ Sí ◉ No

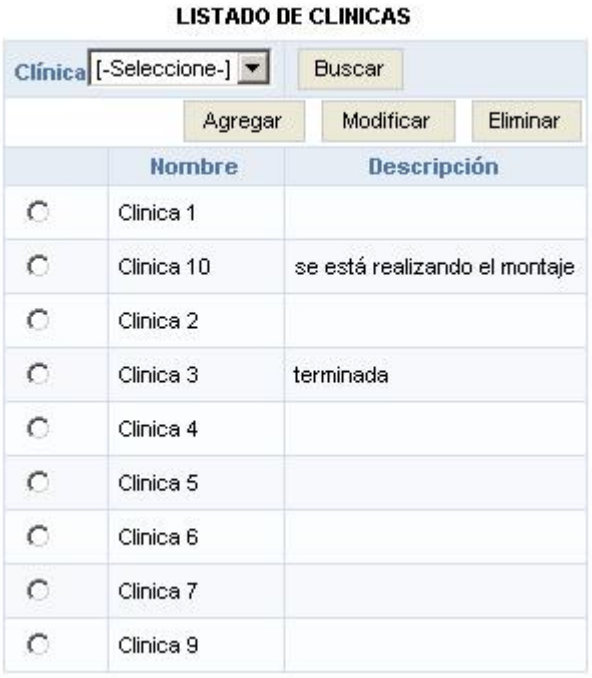
Observaciones:

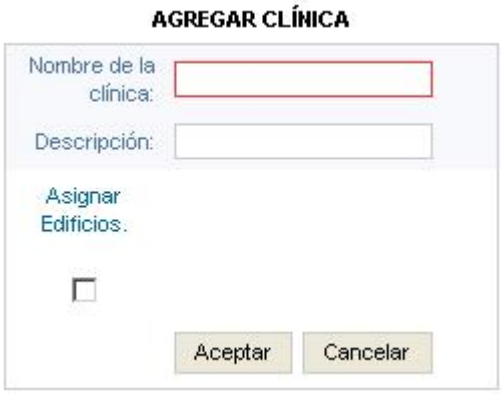
Cuarto: 101106-2

Ocupado: ○ Si ◉ No

Aceptar | Cancelar

Anexo 11: Gestionar Clínicas.





Anexo 12: Gestionar Cuartos.

LISTADO DE CUARTOS

Agregar | Modificar | Eliminar

	Número	Bloqueado	observaciones	Número_apartamento
○	**101106-2**	No		4
○	**101106-3**	No		4
○	**101108-2**	No		5
○	**101108-3**	No		5
○	**101204-1**	No		7
○	**101204-2**	No		7
○	**101204-3**	No		7
○	**101204-4**	No		7
○	**101206-1**	No		8

AGREGAR CUARTO

Cuarto:

Bloqueado: ○ Sí ◉ No

Observaciones:

Apartamento: 102104

Aceptar | Cancelar

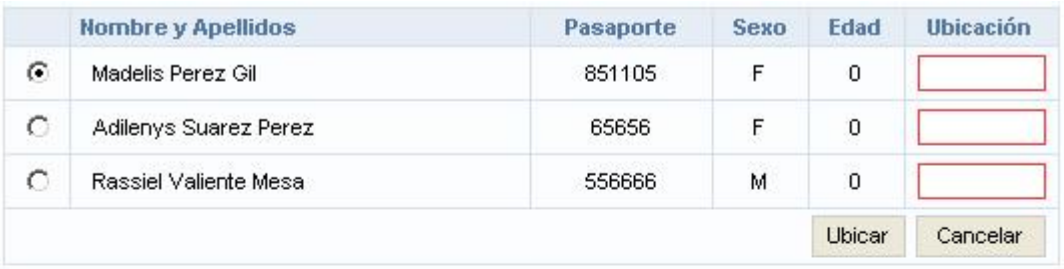
Anexo 13: Gestionar Edificios.

Anexo 14: Gestionar Manzanas.

Anexo 15: Gestionar Ubicación para Estudiantes.

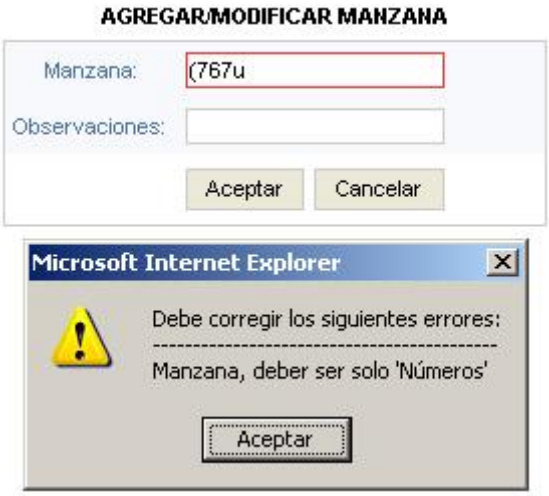
Anexo 16: Gestionar Ubicación para Trabajadores.

Anexo 17: Gestionar Ubicación para hospitalizados.



Anexo 18: Ejemplo del diseño de la interfaz.

Anexo 19: Ejemplo del tratamiento de errores de lado del cliente.



Anexo 20: Ejemplo del tratamiento de errores de lado del servidor.