Universidad de las Ciencias Informáticas

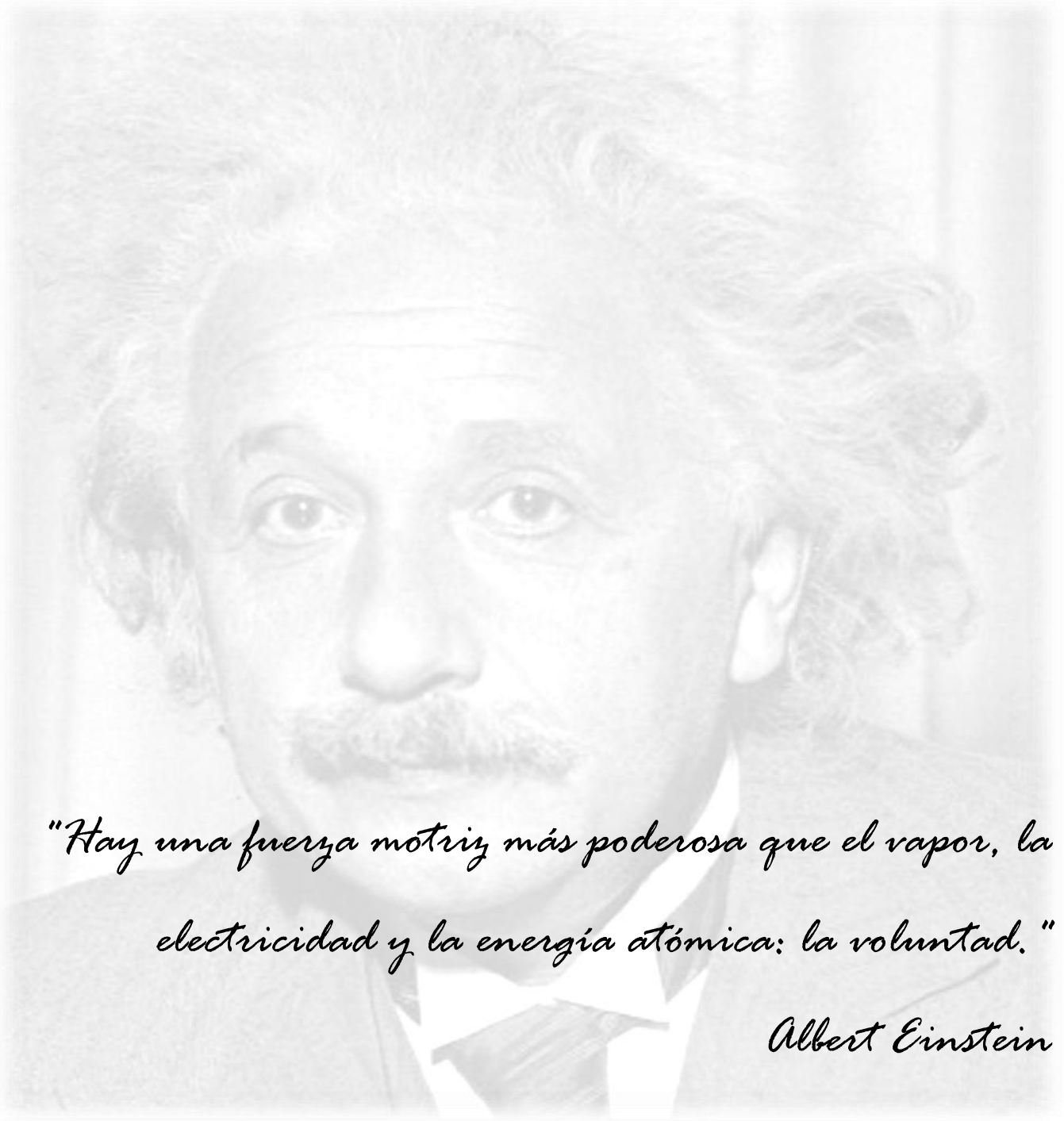
Facultad 6

Título: Sistema de Gestión de Tecnología de la Información de la Universidad de las Ciencias Informáticas: Diseño e implementación del módulo Solicitud de Servicios.

Trabajo de Diploma para optar por el título de Ingeniero Informático.

Autores: Jorge Luis Oliva Matos.
Raydel Vila Martínez.
Tutores: Ing. Lianny de Diego Puentes.
Ing. Alfonso Claro Arceo.

Ciudad de La Habana, junio del 2008



"Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad."

Albert Einstein

Declaración de autoría

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

Para que así conste firmamos la presente a los 24 días del mes de junio del año 2008.

Jorge Luis Oliva Matos

_____

Firma del autor

Raydel Vila Martínez

_____

Firma del autor

Ing. Lianny De Diego Puentes

_____

Firma de la tutora

Ing. Alfonso Claro Arceo

_____

Firma del tutor

Dedicatoria

A esas dos personas que han sabido guiar mis pasos en todo momento, que han estado siempre en buenos y malos momentos, a ellos debo todo lo que soy, a los mejores del mundo, a mis padres. A esa persona que es muy importante en mi vida, que vio como siendo prácticamente un niño entré a la UCI, pero hoy no me puede ver graduado, donde quiera que estés, gracias abuela. A Lien, por su apoyo incondicional, por soportar mis obstinaciones cuando las cosas no salían bien, por estar siempre en los buenos y malos momentos a mi lado, por ser quien es en mi vida. A mi familia, mi hermano, mis primos, mis tíos y tías, a todos ellos que siempre han estado a mi lado, gracias por todo. A mis amigos por permitirme compartir con ellos estos bellos años. A todos ustedes por haber influido decisivamente en mi formación profesional, en general a todas esas personas a quienes debo todo lo que soy y lo que tengo, a ellos va dedicado este trabajo.

Jorge Luis Oliva Matos

A mi querida madre, por haberme traído al mundo...
A mi querida abuelita, que tanto se preocupa por mí...
A mi hermano, para que siga mi ejemplo...
A mi querida novia, por estar siempre a mi lado...
A mis queridos tíos, por el apoyo que me han dado...
A todos mis amigos, por los buenos momentos...
A todos los que de una forma u otra han formado parte de mi vida...

Raydel Vila Martínez

Agradecimientos

A la Revolución y a la Universidad de las Ciencias Informáticas por habernos dado la posibilidad de realizar nuestros sueños.

A nuestros padres, por ser los mejores padres del mundo.

A todos nuestros familiares, que siempre han estado a nuestro lado apoyándonos.

A nuestros tutores, por su ayuda incondicional y dedicación.

A todos nuestros profesores que nos han permitido obtener los conocimientos necesarios para realizar esta investigación.

A nuestros compañeros de grupo, por estar con nosotros todo este tiempo.

A todos nuestros amigos que a lo largo de la vida han estado a nuestro lado en los buenos y en los malos momentos.

A Landy por estar siempre dispuesto a brindarnos su apoyo y sus conocimientos y por echarnos una descarguita siempre que hizo falta.

A todas aquellas personas que día a día nos preguntaban en todos los lugares: ¿Cómo va la tesis?

A los que de una forma u otra pusieron un granito de arena para que esto fuera posible.

A todos, muchas gracias.

Resumen

La obtención de un Sistema de Gestión de Tecnología de la Información en la Universidad de las Ciencias Informáticas que logre satisfacer los requerimientos actuales con que se cuenta, emerge como un problema necesitado de solución.

El diseño e implementación de una aplicación informática que permita gestionar los reportes asociados a la telefonía, la televisión, las redes o la asistencia técnica, teniendo como premisa que este sea unívoco (no exista ambigüedad a la hora de realizarlo) y que se pueda contar con una retroalimentación por parte del cliente asociada al servicio recibido, usando como entorno para su despliegue la Web, son los puntos y elementos que se analizan en esta investigación.

El objetivo perseguido con esta investigación es obtener una aplicación informática que constituya un pilar fundamental en la gestión de los reportes por roturas o desperfectos del equipamiento tecnológico de la Universidad de las Ciencias Informáticas.

Se espera con esta investigación proporcionar al ámbito universitario una herramienta capaz de satisfacer las más exigentes demandas, contribuyendo de esta forma a garantizar una alta disponibilidad tecnológica y la excelencia en la prestación de servicios de asistencia técnica.

Palabras claves: Reportes, Roturas, Desperfectos, Equipamiento Tecnológico.

Índice

Introducción

A raíz del surgimiento de la Batalla de Ideas en el país comienzan a desarrollarse varios programas destinados a fortalecer material y espiritualmente las conquistas de nuestro pueblo. Un ejemplo visible de esto lo constituye el Proyecto Futuro, que surge como resultado de la larga y clara visión del Comandante en Jefe. El producto inmediato de este proyecto fue la creación de la Universidad de las Ciencias Informáticas (UCI) como parte del programa de informatización de la sociedad cubana.

La UCI, moderna universidad de excelencia, donde se aplican novedosos sistemas y concepciones de enseñanza, fue creada para la formación y superación de profesionales de la informática y la producción de software y servicios asociados para la industria nacional y la exportación. En la misma se crea personal calificado para ejercer como ingenieros informáticos y lograr a largo plazo la informatización de los principales sectores de la economía cubana.

Debido al dinamismo y el uso exhaustivo de las tecnologías de la informática y las comunicaciones en la UCI se desarrollan proyectos con el objetivo de automatizar los procesos que tienen lugar en la misma. “La Dirección de Informatización de la UCI, perteneciente a la Infraestructura Productiva (IP), tiene como misión: dirigir, organizar, coordinar, chequear, diseñar y definir la informatización de todos los procesos internos en cada una de las áreas que rigen la vida de la universidad, desde la perspectiva de una Ciudad Digital, logrando una total integración de todas las entidades, flujos y procesos, basado en el funcionamiento armónico de la tecnología y los servicios informáticos.” [1]

En la actualidad existe insuficiencia en la automatización de los procesos asociados a ciertas áreas de la universidad, provocando retrasos en las prestaciones de los servicios solicitados. La Dirección de Gestión Tecnológica es una de las áreas mayormente afectadas por este fenómeno. La Dirección de Gestión Tecnológica tiene la misión de garantizar una alta disponibilidad tecnológica que proporcione a los procesos fundamentales de la universidad un escenario ideal para el alcance de sus objetivos.

Como resultado del constante uso al que se encuentra sometido el equipamiento tecnológico en la universidad, se producen a diario una gran cantidad de roturas y por consiguiente, reparaciones. Por tal motivo la Dirección de Informatización crea el Grupo de Asistencia Técnica (GAT): pequeño grupo compuesto por aproximadamente de 20 a 22 técnicos que tienen entre sus funciones la recepción de las solicitudes de los usuarios y la solución de las mismas; este grupo a medida que crecía la UCI, incrementaba también.

De septiembre del 2002 hasta diciembre del 2005 el proceso de solicitud de servicios se realizaba de la siguiente manera: el usuario o responsable del equipo reportaba por vía telefónica (6060) la rotura o desperfecto al Grupo de Asistencia Técnica. Esta llamada o solicitud de servicio se recogía

diariamente en un libro de incidencias que se volcaba en un reporte individual que era impreso y entregado al técnico que atendía el área para que diera solución a este reporte, por este concepto el Grupo de Asistencia Técnica consumía un paquete de hojas promedio mensual y un tonel para impresora HP Láser Jet 1000 cada dos o tres meses aproximadamente.

"Este procedimiento traía como consecuencias:

1. Duplicidad de solicitudes.
2. Pérdida o extravío de las boletas emitidas con las solicitudes realizadas por los usuarios.
3. Ineficiencia en el trabajo de los técnicos.
4. Imposibilidad de controlar el desempeño de cada técnico, por lo que no se podía controlar el plan de trabajo.
5. Gasto innecesario de papel y tonel para impresora.
6. Insatisfacción de los usuarios." [2]

En la actualidad, existe una aplicación que da solución a la situación que existía con los reportes de los equipos rotos por parte de los usuarios y al consumo innecesario de insumos para los trámites a fin de prestar el servicio. Este sistema permite a la operadora registrar los reportes por rotura o desperfecto del equipamiento tecnológico luego de haber sido recepcionados por vía telefónica (6060).

Los reportes de asistencia técnica están automatizados, pero las áreas involucradas en la emisión de reportes presentan problemas en la identificación única de los equipos a reportar ya que no existe una definición del flujo de procesos para mantener actualizada la base de datos de inventarios de medios por local de la universidad. A este problema se suma que el acceso a la aplicación está limitado a la dirección de asistencia técnica y que los reportes solo se pueden realizar mediante llamadas telefónicas. También se presenta la problemática de que la aplicación existente no se rige por las políticas actuales de desarrollo de sistemas en la UCI, las cuales plantean como uno de sus principios el uso del Software Libre en los productos desarrollados.

Problema científico: ¿Cómo obtener un producto funcional a partir de los requerimientos identificados para el módulo Solicitud de Servicios del Sistema de Gestión de Tecnología de la Información de la Universidad de las Ciencias Informáticas?

Objeto de estudio: Proceso de desarrollo de software para la gestión de servicios técnicos.

Campo de acción: Proceso de desarrollo de software para la gestión de servicios técnicos en la Universidad de las Ciencias Informáticas.

Objetivo general: Desarrollar el diseño y la implementación del módulo Solicitud de Servicios del Sistema de Gestión de Tecnología de la Información de la Universidad de las Ciencias Informáticas.

Como **objetivos específicos** se plantean:

- Diseñar el sistema.
- Implementar el sistema.

Tareas desarrolladas para cumplir los objetivos:

- Análisis de la arquitectura propuesta por la Dirección de Informatización de la Universidad de las Ciencias Informáticas para el desarrollo de la aplicación.
- Refinamiento de los requerimientos funcionales y no funcionales identificados.
- Obtención del diagrama de casos de uso del sistema.
- Obtención del diagrama de clases del diseño.
- Obtención del diagrama de clases persistentes.
- Obtención del modelo de datos.
- Obtención del diagrama de despliegue.
- Obtención del diagrama de componentes.
- Implementación del sistema.

Para lograr una mejor comprensión de la presente investigación se decide estructurar el contenido en tres capítulos donde se recoge todo el contenido de la misma.

Capítulo 1 Fundamentación Teórica: Este capítulo contiene un resumen de los principales artefactos que se obtuvieron en la investigación precedente. Además se describen los principales lineamientos en los cuales se basa la arquitectura de la UCI para el desarrollo de aplicaciones, así como las tendencias, técnicas, tecnologías y metodologías relacionadas con dicha arquitectura y las plataformas de desarrollo que la soportan. También se hace un estudio crítico y valorativo de la técnica de programación y plataformas en las que se basa este trabajo para darle solución al problema.

Capítulo 2 Diseño del Sistema: En este capítulo se plasman los requerimientos funcionales y no funcionales identificados. Además se documentan los artefactos que se obtienen a partir del diseño del sistema, tales como: diagrama de casos de uso del sistema, descripciones textuales de los casos de uso del sistema, diagrama de clases persistentes, diagrama de clases Web, modelo de datos y diagrama de despliegue. También se describen los estilos arquitectónicos y patrones de diseño utilizados para el desarrollo de la aplicación.

Capítulo 3 Implementación del Sistema: En este capítulo se plasman los artefactos que se obtienen como resultado de la implementación del sistema. Se especifican los componentes físicos que conforman el sistema y sus relaciones. Se incluye además el código fuente de las principales clases y pantallas de la aplicación y se brinda una descripción del mismo.

Capítulo I
FUNDAMENTACIÓN TEÓRICA

Este capítulo contiene un resumen de los principales artefactos que se obtuvieron en la investigación precedente. Además se describen los principales lineamientos en los cuales se basa la arquitectura de la UCI para el desarrollo de aplicaciones, así como las tendencias, técnicas, tecnologías y metodologías relacionadas con dicha arquitectura y las plataformas de desarrollo que la soportan. También se hace un estudio crítico y valorativo de la técnica de programación y plataformas en las que se basa este trabajo para darle solución al problema.

1.1 Artefactos de la investigación precedente.

Como antecedente a esta investigación se tiene el Trabajo de Diploma “Modelación del sistema de reportes para la dirección de Gestión Tecnológica de la Universidad de las Ciencias Informáticas”, en el mismo se encuentra documentada la modelación del sistema que se pretende implementar. Entre los principales artefactos que arrojó dicha investigación están:

- Modelo de negocio.
- Requerimientos funcionales (40).
- Requerimientos no funcionales.
- Casos de uso del sistema (22).
- Diagrama de clases del análisis.
- Diagrama de clases del diseño.
- Modelo de datos.

Debido a los cambios surgidos en los procesos que engloban el trabajo del Grupo de Asistencia Técnica, se hizo necesario un refinamiento de la solución propuesta por la investigación precedente. Los nuevos artefactos que se obtienen aparecen reflejados en el siguiente capítulo.

1.2 Arquitectura de Software.

“La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes y las relaciones entre los mismos, el ambiente y los principios que orientan su diseño y evolución.” [3] Es una vista estructural de alto nivel, ocurre muy tempranamente en el ciclo de vida de un producto software y define los estilos o grupos de estilos adecuados para cumplir con los

requerimientos no funcionales. Estas cuestiones estructurales se vinculan con el diseño, pues la arquitectura de software es después de todo una forma de diseño de software que se manifiesta tempranamente en el proceso de creación de un sistema, pero este diseño ocurre a un nivel más abstracto que el de los algoritmos y las estructuras de datos, dichas cuestiones estructurales incluyen organización a grandes rasgos y estructura global de control, protocolos para la comunicación, la sincronización y el acceso a datos, la asignación de funcionalidad a elementos del diseño, la distribución física, la composición de los elementos de diseño, escalabilidad y rendimiento, y selección entre alternativas de diseño.

Para el desarrollo de la presente investigación se utilizará la arquitectura SOA, pues cumple con los principales lineamientos de la arquitectura propuesta por la Dirección de Informatización.

1.2.1 Arquitectura Orientada a Servicios.

"La Arquitectura Orientada a Servicios (en inglés Service Oriented Architecture o SOA), es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requerimientos de software del usuario." [4] Proporciona una metodología y un marco de trabajo para documentar las capacidades de negocio y puede dar soporte a las actividades de integración y consolidación.

En un ambiente SOA, los nodos de la red hacen disponibles sus recursos a otros participantes en la red como servicios independientes a los que tienen acceso de un modo estandarizado. La mayoría de las definiciones de SOA identifican la utilización de Servicios Web empleando SOAP (Simple Object Access Protocol) y WSDL (Web Services Description Language) en su implementación, no obstante se puede implementar una SOA utilizando cualquier tecnología basada en servicios.

Al contrario de las arquitecturas orientadas a objetos, las SOAs están formadas por servicios de aplicación débilmente acoplados y altamente interoperables. Para comunicarse entre sí, estos servicios se basan en una definición formal independiente de la plataforma subyacente y del lenguaje de programación. La definición de la interfaz encapsula las particularidades de una implementación, lo que la hace independiente del fabricante, del lenguaje de programación o de la tecnología de desarrollo. Con esta arquitectura, se pretende que los componentes de software desarrollados sean muy reusables, ya que la interfaz se define siguiendo un estándar.

Servicio

"Un servicio es una función sin estado, autocontenida, que acepta una o varias llamadas y devuelve una o varias respuestas mediante una interfaz bien definida. Los servicios pueden también ejecutar unidades discretas de trabajo como serían editar y procesar una transacción. Los servicios no dependen del estado de otras funciones o procesos. La tecnología concreta utilizada para prestar el

servicio no es parte de esta definición, lo que implica que en una SOA pueden coexistir aplicaciones escritas en distintas tecnologías y lenguajes de programación.” [1]

SOAP

“SOAP (siglas de Simple Object Access Protocol) es un protocolo estándar creado por Microsoft, IBM y otros, está actualmente bajo el auspicio de la W3C que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. SOAP es uno de los protocolos utilizados en los servicios Web.” [1]

WSDL

“WSDL son las siglas de Web Services Description Language, un formato XML que se utiliza para describir servicios Web. WSDL describe la interfaz pública a los servicios Web. Está basado en XML y describe la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Las operaciones y mensajes que soporta se describen en abstracto y se ligan después al protocolo concreto de red y al formato del mensaje.” [1]

1.3 Las Aplicaciones Web.

“Una aplicación Web es un sistema informático que los usuarios utilizan accediendo a un servidor Web a través de Internet o de una intranet. Las aplicaciones Web son populares debido a la practicidad del navegador Web como cliente ligero. La facilidad para actualizar y mantener aplicaciones Web sin distribuir e instalar software en miles de potenciales clientes es otra razón de su popularidad.” [2]

Las aplicaciones Web generan dinámicamente una serie de páginas en un formato estándar, soportado por navegadores Web comunes como HTML o XHTML. Se utilizan lenguajes interpretados del lado del cliente, tales como JavaScript, para añadir elementos dinámicos a la interfaz de usuario. Generalmente cada página Web individual es enviada al cliente como un documento estático, pero la secuencia de páginas provee de una experiencia interactiva.

Ventajas de las aplicaciones Web:

- Multiplataforma: Con un solo programa, un único ejecutable, las 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, esta situación crecerá indefinidamente. Es por ello que se decide exponer las funcionalidades del sistema a implementar a través de una aplicación Web, por todas las ventajas que ofrece y por las características propias del entorno sobre el cual se va a realizar el despliegue de la misma.

1.4 Lenguajes de Programación.

La diferencia fundamental de Internet a otros medios de comunicación es la interacción y personalización de la información con el usuario. Esto se logra por medio de algunos de los diferentes lenguajes de programación para Web que existen hoy en día. Dichos lenguajes se clasifican en dos partes fundamentales: los lenguajes del lado del Servidor y los lenguajes del lado del Cliente.

Entre los lenguajes del lado del Servidor podemos encontrar entre los más sobresalientes por el auge que estos han tenido, algunos como PERL, ASP, PHP, Java y los módulos CGIs e ISAPIs. Estos se caracterizan por desarrollar la lógica de negocio dentro del Servidor, además de ser los encargados del acceso a bases de datos y tratamiento de la información. Del lado del Cliente se encuentra principalmente el JavaScript, encargado de aportar dinamismo a la aplicación en los navegadores.

Lenguajes de programación del lado del Cliente.

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 de los usuarios y mantienen 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 interactividad, velocidad y usabilidad en la misma.” [5]

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:

1. XHTML (o HTML) y hojas de estilos en cascada (CSS) para el diseño que acompaña a la información.
2. 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.
3. 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.
4. XML es el formato usado comúnmente para la transferencia de vuelta al servidor, aunque cualquier formato puede funcionar, incluyendo HTML pre formateado, texto plano, JSON y hasta EBML.

Por la interactividad que aporta esta tecnología a las aplicaciones Web, por hacer la experiencia del usuario más amigable ya que los resultados esperados de la aplicación son devueltos mucho más rápido, y por estar comprendida dentro de la arquitectura propuesta por la Dirección de Informatización, se decide utilizar Ajax en la implementación de la aplicación.

JavaScript

"JavaScript es un lenguaje de programación que no necesita compilación y se utiliza dentro del HTML. No es un lenguaje orientado a objetos pues no tiene herencia, se puede llamar como un lenguaje de prototipos, ya que mediante la clonación de las clases bases es que se obtienen las nuevas clases." [6] Tradicionalmente se ha utilizado en la realización de tareas y operadores en el marco de la aplicación únicamente del cliente.

El lenguaje es interpretado por el navegador y en la página Web, donde se encuentra insertado produce una acción determinada. JavaScript es basado en objetos, guiado por eventos y débilmente tipiado, de ahí deriva el dinamismo que pueden alcanzar las páginas que incluyan esta clase de código.

En la implementación de la aplicación se usa JavaScript para validar la entrada de datos de los usuarios y además porque el mismo forma parte de la técnica de desarrollo Web Ajax.

Lenguajes de programación del lado del Servidor.

PHP

"PHP (Hypertext Preprocessor) es un lenguaje de código abierto interpretado, de alto nivel, embebido en páginas HTML y ejecutado en el servidor. La meta de este lenguaje es permitir escribir a los creadores de páginas Web, páginas dinámicas de una manera rápida y fácil." [7]

Una de las características más potentes del lenguaje PHP es su soporte para gran cantidad de base de datos, entre las que se pueden mencionar Adabas D, dBase, Empress, Hyperwave, IBM DB2, Internase, FrontBase, mSQL, Direct MS-SQL, MySQL ODBC, Oracle (OCI7 and OCI8), PostgreSQL, Sybase, Unix dbm, siendo las más estándares y usadas MySQL y PostgreSQL. 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) y analizar código XML (eXtensible Markup Language) hasta la creación de programas incluyendo aplicaciones con interfaz gráfica usando la librería GTK+.

El lenguaje PHP puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, incluyendo Linux, muchas variantes Unix (incluyendo Solaris y OpenBSD), Microsoft Windows, Mac OS X y RISC OS. PHP soporta la mayoría de servidores Web de hoy en día, incluyendo Apache, Microsoft Internet Information Server, Personal Web Server, Netscape e iPlanet, Oreilly Website Pro server, Caudium, Xitami y OmniHTTPd.

PHP también permite la comunicación con otros servicios usando protocolos tales como LDAP, IMAP, SNMP, NNTP, POP3, HTTP y COM (en Windows). Con PHP4 se han logrado aumentos de entre 5 y 10 veces en la velocidad de ejecución de páginas PHP. Este nuevo rendimiento le ha puesto por delante de ASP, la tecnología de Microsoft.

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

Por las características y ventajas de PHP y por ser la propuesta de la Dirección de Informatización para el desarrollo de aplicaciones en la UCI, PHP es el lenguaje de programación Web que se usa para el desarrollo de la aplicación en su versión 5.1.6.

1.5 Frameworks

En el desarrollo de software, un framework es una estructura de soporte definida en la cual un proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, librerías y un lenguaje de scripting para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Frameworks de PHP.

Los frameworks de PHP ayudan en el desarrollo de un software, ya que proporcionan una estructura definida, la cual permite crear aplicaciones con mayor rapidez y darle mantenimiento a las mismas

gracias a la organización que brindan. Los frameworks son desarrollados con el objetivo de brindar a los programadores y diseñadores una mejor organización y estructura a sus proyectos. Utilizan la Programación Orientada a Objetos (POO), permitiendo la reutilización de código.

La Dirección de Informatización de la Universidad de las Ciencias Informáticas propone el uso de varios frameworks para la confección de sitios en PHP de manera más rápida, entre ellos están CakePHP, Symfony y CodeIgniter.

CakePHP

CakePHP es un framework Open Source escrito en PHP, modelado bajo los conceptos de Ruby on Rails, y distribuido bajo la licencia MIT. Su desarrollo, que comenzó en el 2005, ha ido creciendo con el tiempo, llegando incluso a contar con una propia forja de subproyectos basados en CakePHP. Como Ruby on Rails, CakePHP hace más fácil para el usuario interactuar con la base de datos mediante su ORM, Active Records. Entre sus características se destacan:

- Compatible con PHP4 y PHP5.
- CRUD integrado para la base de datos y consultas simplificadas.
- Request dispatcher con URLs personalizadas.
- Plantillas de sintaxis PHP con métodos de ayuda (helpers) integrados.
- Facilidades para las vistas con Ajax, JavaScript y formularios HTML.
- Validación built-in.
- Listas de control de acceso (ACL).
- Aplicación de scaffolding.
- Componentes para el manejo de peticiones, sesiones y seguridad.
- Caché para las vistas.

Symfony

Symfony es un framework diseñado para optimizar el desarrollo de aplicaciones Web a través de diversas características claves. Contiene una gran variedad de herramientas y clases para conseguir acortar el tiempo de desarrollo de aplicaciones Web complejas. Adicionalmente automatiza tareas comunes para que el programador pueda enfocarse por completo en las especificaciones. Entre las características generales del framework se pueden citar:

- Fácil de instalar y configurar, ha sido probado con éxito en plataformas Windows y derivadas de Unix.

- Independiente del gestor de base de datos, utiliza Propel, una capa de abstracción que le permite interactuar con diferentes gestores de bases de datos.
- Simple de usar y al mismo tiempo lo suficientemente flexible para adaptarse a escenarios complejos.
- Cumple con la mayoría de las mejores prácticas en el diseño Web y patrones de diseño.
- Utilizable en entornos empresariales, puede adaptarse a políticas y arquitecturas ya existentes en tecnologías de información, y es lo suficientemente estable para proyectos de largo plazo.
- Código legible, con comentarios en phpDocumentor para su fácil mantenimiento.
- Fácil de extender, permitiendo la integración con otras librerías.
- Incorpora herramientas que facilitan la prueba y depuración de aplicaciones, como unidades de generación de código, pruebas del funcionamiento del framework, panel de depuración, interfaz por línea de comandos y configuración en tiempo real.
- Incorpora una capa de internacionalización que posibilita la traducción de datos e interfaces, así como la localización de contenido en función de la ubicación geográfica del usuario.
- Uso de plantillas, las cuales pueden ser elaboradas por diseñadores de páginas Web que desconocen el resto de los detalles técnicos del framework.
- Asistentes de vistas, que reducen la cantidad de código de presentación al escribir grandes bloques de código con simples llamadas a funciones.
- Validación y regeneración automática de formularios, lo que asegura una buena calidad de los datos en la base de datos y una mejor experiencia de usuario.
- Verificación de la salida enviada por la aplicación, que ofrece una protección frente a ataques por datos corruptos.
- Manejo de memoria caché, lo cual reduce el uso del ancho de banda y la carga en el servidor.
- Mecanismos de autentificación y credenciales, que facilitan la creación de secciones restringidas y la gestión de seguridad de usuarios.
- URLs inteligentes que permiten que las direcciones de las páginas Web sean parte de la interfaz y resulten amigables a los motores de búsqueda.
- Una gestión de listas más amigables al usuario, gracias a la paginación automática, ordenamiento y filtrado de resultados.

CodeIgniter

CodeIgniter es un poderoso framework de PHP que permite a los desarrolladores construir aplicaciones mucho más rápido, ofreciendo un rico conjunto de librerías para tareas comunes, así

como una interfaz simple y estructura lógica para el acceso a dichas librerías. Su máxima cualidad es la cantidad de funciones que implementa por defecto. Entre sus características se pueden mencionar:

- Extremadamente ligero.
- Soporte para múltiples gestores de base de datos.
- Validación de formularios y datos.
- Seguridad y filtrado de XSS.
- Gestión de sesiones.
- Clases para correo electrónico, manipulación de imágenes, subida de ficheros, FTP, calendario, User Agent, ZIP, trackback y XML-RPC.
- Internacionalización.
- Paginación.
- Codificación de datos (encryption).
- Benchmarking.
- Cacheo de páginas completas.
- Logs de error.
- Perfiles de aplicación.
- Scaffolding.
- Template Engine.
- Unit Testing.
- Búsquedas optimizadas y URL's amigables.
- Soporte para hooks, extensión de clases y plugins.
- Gran librería de funciones (helpers).

Luego de haberse analizado las principales características de estos frameworks de PHP, se decide utilizar CodeIgniter para el desarrollo de la aplicación en su versión 1.6.1, ya que el tiempo requerido para su aprendizaje es mucho menor que el tiempo que se necesita para aprender cualquiera de los otros, además este requiere una mínima configuración para su uso.

1.6 Sistemas de Gestión de Bases de Datos.

Un Sistema Gestor de Base de Datos (SGBD) es un conjunto de programas 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 son:

- Definir una base de datos: especificar tipos, estructuras y restricciones de datos.
- Construir la base de datos: guardar los datos en algún medio controlado por el mismo SGBD.

- Manipular la base de datos: realizar consultas, actualizarla, generar informes.

Las principales características que debe contemplar un SGBD son:

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

En el mundo informático existen muchos Sistemas de Gestión de Base de Datos con soporte SQL, el que propone la Dirección de Informatización para el desarrollo de aplicaciones en la UCI es PostgreSQL.

PostgreSQL

PostgreSQL es un servidor de base de datos relacional orientado a objetos de software libre, liberado bajo la licencia BSD. Como muchos otros proyectos open source, el desarrollo de PostgreSQL no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group). Este robusto gestor de base de datos implementa el estándar SQL92/SQL99 y soporta distintos tipos de datos, al mismo tiempo que brinda soporte para los tipos de datos base soporta además datos de tipo fecha, monetarios, elementos gráficos y cadenas de bits, permite la creación de tipos propios e incorpora una estructura de datos array. Además añade funciones de diversa índole: manejo de fechas, geométricas y orientadas a operaciones con redes. Permite la declaración de funciones propias, así como la definición de disparadores. Soporta el uso de índices, vistas y procedimientos almacenados en múltiples lenguajes. Tiene interfaces de programación nativos para C/ C++, Java, Perl, Python, Ruby, Tcl, ODBC, entre otros. Es multiplataforma, funciona en todos los sistemas operativos importantes, incluyendo Linux, UNIX y Windows. A pesar de que la velocidad de respuesta que ofrece PostgreSQL para la gestión de bases de datos relativamente pequeñas puede parecer un poco deficiente, en el caso de que se gestionen bases de datos considerablemente grandes, la velocidad es la misma, lo cual resulta aceptable.

Por todas las ventajas del uso de este SGBD y por ser la propuesta de la Dirección de Informatización para el desarrollo de aplicaciones en la UCI, se decide usar PostgresSQL en el desarrollo de la aplicación en su versión 8.1.3.

1.7 Metodologías de Desarrollo de Software.

Todo proceso de desarrollo de software es riesgoso y difícil de controlar, si no se hace uso de una metodología entonces se obtienen clientes insatisfechos con el resultado y desarrolladores mucho más. Sin embargo muchas veces no se tiene en cuenta utilizar una metodología adecuada, sobre todo cuando se trata de proyectos pequeños.

Con relación a los proyectos de mayor envergadura sí toma sentido el basarse en una metodología de desarrollo y se empieza a buscar cual sería la más apropiada para el mismo. Al final en la mayoría de los casos no se encuentra la más adecuada y se hace un diseño propio de la metodología, por supuesto no está mal siempre y cuando sirva para alcanzar el objetivo.

Muchas veces se diseña el software de manera rígida, tal como el cliente lo solicitó, de esa manera cuando el cliente en la etapa de prueba solicita un cambio se hace muy difícil realizarlo, siendo este uno de los factores que atrasan el proyecto, crea incomodidad al desarrollador y en muchas ocasiones no se logra realizar el cambio solicitado, provocando insatisfacción en el cliente puesto que no ha sido tomado en cuenta su pedido; para evitar estos incidentes se debe llegar a un acuerdo formal con el cliente al inicio del proyecto de manera que no se perjudique el desarrollo del mismo.

Muchas veces los usuarios finales se dan cuenta que dejaron de mencionar algunas cosas y lo manifiestan en la etapa inicial del proyecto cuando se le muestra el prototipo del mismo.

Para el desarrollo de esta investigación se decide utilizar como metodología de desarrollo de software RUP, por sus características, facilidades que aporta a todo el proceso y por ser la metodología que se utilizó en la investigación precedente.

RUP

El Proceso Racional Unificado o Rational Unified Process (RUP) es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software. Sin embargo, el Proceso Unificado es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organización, diferentes niveles de aptitud y diferentes tamaños de proyecto. El Proceso Unificado está basado en componentes, lo cual quiere decir que el sistema software en construcción está formado por componentes software interconectados a través de interfaces bien definidas.

RUP constituye una metodología estándar para el desarrollo de sistemas orientados a objetos. Provee un enfoque disciplinado en la asignación de tareas y responsabilidades dentro de una organización de desarrollo de software. Su meta es asegurar la producción de software de muy alta

calidad que satisfaga las necesidades de los usuarios finales, dentro de un calendario y presupuesto predecible. Utiliza el Lenguaje Unificado de Modelado o Unified Modeling Language (UML) para visualizar, especificar, construir y documentar artefactos de un sistema de software.

Las características fundamentales de RUP son:

- Guiado por casos de uso.
- Centrado en la arquitectura.
- Iterativo e Incremental.

RUP divide en 4 fases el desarrollo del software, las cuales son:

- Inicio: Se determina la visión del proyecto.
- Elaboración: Se determina la arquitectura óptima.
- Construcción: Se obtiene la capacidad operacional inicial.
- Transición: Se obtiene el release del proyecto.

La metodología RUP define un conjunto de roles a los cuales asocia las actividades que se desarrollan dentro de un proyecto. En este trabajo se abordan los roles de diseñador, diseñador de bases de datos e implementador.

Diseñador: Este rol define las responsabilidades, las operaciones, los atributos, y las relaciones de una o varias clases y determina cómo serán ajustadas al ambiente de implementación. Además, el rol diseñador puede tener la responsabilidad de unos o más paquetes de diseño, o del diseño de subsistemas, incluyendo cualquiera de los contenidos en los paquetes o los subsistemas.

Diseñador de bases de datos: Este rol es el encargado de definir las tablas, índices, vistas, constraints, triggers y otros objetos específicos de la base de datos necesarios para almacenar, recuperar y eliminar objetos persistentes.

Implementador: Este rol es responsable de desarrollar y de probar componentes de acuerdo con los estándares adoptados por el proyecto para la integración en subsistemas más grandes. Cuando los componentes de prueba, tales como drivers o stubs deben ser creados para apoyar las pruebas, el implementador es también responsable de desarrollar y de probar los componentes de prueba y los subsistemas correspondientes.

1.8 Herramientas

CASE: Visual Paradigm

Visual Paradigm es una herramienta visual para el modelado UML que brinda soporte a todo el ciclo de vida de un producto software: análisis y diseño, construcción, pruebas y despliegue. Está disponible en varias ediciones, cada una destinada a una necesidad diferente: Enterprise, Professional, Community, Standard, Modeler y Personal. Ofrece un entorno de creación de diagramas para UML, un diseño

centrado en casos de uso y enfocado al negocio que propicia un software de calidad y provee a todo el equipo de desarrollo de un lenguaje estándar común que facilita la comunicación y capacidades de ingeniería directa e inversa. Entre las características fundamentales de esta herramienta se puede destacar que brinda la posibilidad de integrarse con los principales Entornos de Desarrollo Integrado (IDEs) y que es multiplataforma, funciona en los sistemas operativos más importantes como Linux, Windows y Mac OS X. Esta herramienta está diseñada para una amplia gama de usuarios entre los que se incluyen ingenieros de software, analistas de sistemas, analistas de negocio, arquitectos y desarrolladores y está orientada a la creación de diseños utilizando el paradigma de programación orientada a objetos. A pesar de que Visual Paradigm es un software propietario constituye la herramienta CASE utilizada para modelar y documentar los productos software que se desarrollan en la universidad ya que la misma adquirió la licencia que avala su utilización. Por todo lo antes expuesto se decide utilizar Visual Paradigm como herramienta CASE para el modelado en su versión 6.0.

Diseño:

KompoZer

KompoZer es un editor HTML del tipo WYSIWYG. Opera bajo la licencia GNU, la cual pertenecen al grupo de licencias de software libre, por lo que el código fuente está disponible para todos de forma gratuita. Es multiplataforma, funciona en los sistemas operativos más importantes como Linux, Windows y Mac OS X. Esta herramienta está diseñada para ser extremadamente fácil de utilizar, por lo que no se necesita tener amplios conocimientos de HTML o de codificación Web, es muy rápida y consume pocos recursos.

Dreamweaver

Dreamweaver es la herramienta líder de desarrollo de aplicaciones Web, es la opción profesional para crear sitios Web y aplicaciones, dado que proporciona una potente combinación de herramientas visuales de diseño, funciones de desarrollo de aplicaciones y soporte para la edición del código, características que permiten a los desarrolladores y diseñadores más o menos expertos crear rápidamente sitios Web y aplicaciones basados en estándares internacionales. Desde el avanzado soporte de diseño basado en hoja de estilo (CSS) a las funciones de codificación manual, Dreamweaver proporciona las herramientas profesionales que requiere un entorno integrado y agilizado. Los desarrolladores pueden utilizar Dreamweaver con su tecnología de servidor preferida para crear potentes aplicaciones en Internet destinadas a conectar a los usuarios a las bases de datos, las fuentes de datos dinámicos y los sistemas heredados. Es además compatible con todas las principales tecnologías de servidor como ColdFusion, PHP, ASP, ASP.NET y JSP.

A pesar de que Dreamweaver es una herramienta propietaria, sus características propias lo hacen la mejor opción a la hora de crear aplicaciones Web, una de sus ventajas determinantes sobre el KompoZer es que es compatible con las principales tecnologías del lado del servidor, mientras que KompoZer es simplemente un herramienta que a pesar de ser libre y multiplataforma, solo se emplea en el modelado visual de las páginas Web, todo esto se suma a que por necesidades propias de la aplicación, y el uso del framework de PHP CodeIgniter, se hace necesario la inclusión de código PHP en las vistas del sistema, esto deja fuera de consideración el uso del KompoZer. Por todo lo antes expuesto se decide utilizar Dreamweaver como herramienta de diseño en su versión 8.0.

Desarrollo:

Eclipse PDT

Eclipse es una plataforma de desarrollo de código abierto, fácil de usar. Integrable con Web Tools de Eclipse, extensibilidad y soporte continúo de desarrolladores PHP. Es un IDE de desarrollo multiplataforma ejecutándose en varios sistemas operativos como Windows y Linux. Posee además integración para el auto completamiento de código con frameworks de PHP. Eclipse PDT posee además facilidades para la integración con SVN, además de permitir generar Web Services incluyendo el WSDL con mucha facilidad.

PHP Designer

PHP Designer es un IDE para el desarrollo de sitios Web en PHP. Se puede decir que es más potente que el Macromedia Dreamweaver en cuanto a editor de texto para programación en PHP, XHTML, CSS y XML, aunque otros editores lo superan en la parte de diseño visual. Entre sus características fundamentales está presentar un editor todo en uno: edita, testea, analiza y publica scripts en PHP; resaltado de la sintaxis y soporte para PHP, HTML, XML, CSS, JavaScript, Java, Perl, VB, C# y SQL; completamiento automático de código mientras se escribe; integración con el manual de PHP desde su versión online; integración sencilla con navegadores externos (Internet Explorer, Netscape, Firefox y Opera); gestor de proyectos, cliente de FTP, librerías de código, navegador de clases y plantillas, entre una infinidad de características.

Zend Studio

Se trata de un programa de la casa Zend, impulsores de la tecnología de servidor PHP. Además de servir de editor de texto para páginas PHP, proporciona una serie de ayudas que pasan desde la creación y gestión de proyectos hasta la depuración de código.

El programa entero está escrito en Java, lo que a veces supone que no funcione tan rápido como otras aplicaciones de uso diario. Es un IDE multiplataforma que consta de dos partes en las que se dividen las funcionalidades de parte del cliente y las del servidor, la del cliente contiene la interfaz de edición y la ayuda. Permite además hacer depuraciones simples de scripts, ya que para disfrutar de toda la potencia de la herramienta de depuración hay que disponer de la parte del servidor. Zend Studio implementa además opciones para trabajar en grupo, al integrar el sistema de trabajo conocidos como CVS o Subversion.

Luego de haberse realizado un análisis comparativo de los editores de PHP más utilizados en la actualidad, se llega a la conclusión de que el más apropiado para el desarrollo de la aplicación es Eclipse PDT, debido a su gran integración con el framework de PHP que se usará, por ser multiplataforma, por el completamiento de código que presenta, por la integración con ayudas online que posee y por el control y seguimiento de errores que realiza, además emplea módulos (en inglés plugins) que extienden su funcionalidad, a diferencia de otros IDEs en los cuales estos son prefijados.

Administración de Bases de Datos: PgAdmin III

PgAdmin III es una herramienta gráfica para la administración de bases de datos PostgreSQL, siendo la más completa y popular con licencia Open Source. Es multiplataforma, funciona en los sistemas operativos más importantes como Linux, Solaris, Windows y Mac OS X. PgAdmin está diseñada para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de PostgreSQL y facilita enormemente la administración. La aplicación incluye un editor SQL con resaltado de sintaxis, un editor de código del lado del servidor, un agente para lanzar scripts programados e incluye soporte para el motor de replicación Slony-I. La conexión al servidor puede hacerse utilizando TCP/IP o Unix Domain Sockets (en plataformas *nix) y puede encriptarse mediante SSL para mayor seguridad. La versión utilizada es la 1.4.1.

Servidor de Aplicación: Apache

El servidor Apache es un software que está 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:

- 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. 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. Por todo lo antes expuesto se decide utilizar Apache como servidor de aplicación en su versión 2.0.59.

1.9 Conclusiones

Después de haber realizado un análisis de las normas técnicas y tecnologías aprobadas por la Dirección de Informatización de la UCI para el desarrollo de la solución propuesta, se decide realizar una aplicación Web utilizando como lenguaje de programación PHP, apoyados en el framework CodeIgniter para agilizar el desarrollo de la misma, de conjunto con el sistema gestor de base de datos PostgreSQL. Se definieron además las herramientas a usar para el desarrollo de la aplicación, las cuales son Visual Paradigm como herramienta CASE, Dreamweaver como herramienta para el diseño, Eclipse PDT como herramienta de desarrollo, PgAdmin III para la administración de la base de datos y Apache como servidor de aplicación.

Capítulo II

DISEÑO DEL SISTEMA

En este capítulo se plasman los requerimientos funcionales y no funcionales identificados. Además se documentan los artefactos que se obtienen a partir del diseño del sistema, tales como el diagrama de casos de uso del sistema, las descripciones textuales de los casos de uso del sistema, el diagrama de clases persistentes, el diagrama de clases Web, el modelo de datos y el diagrama de despliegue. También se describen los estilos arquitectónicos y patrones de diseño utilizados para el desarrollo de la aplicación.

2.1 Requerimientos Funcionales.

"Los requerimientos funcionales especifican acciones que el sistema debe ser capaz de realizar, sin tomar en consideración ningún tipo de restricción física. Por lo tanto los requerimientos funcionales especifican el comportamiento de entrada y salida del sistema y surgen de la razón fundamental de la existencia del producto." [8]

Listado de requerimientos funcionales identificados.

RF1: Autentificar Usuario
RF2: Crear Reporte
RF3: Buscar y Visualizar Reporte
RF4: Cancelar Reporte
RF5: Reclamar Reporte
RF6: Evaluar Servicio
RF7: Atender Reporte
RF8: Reasignar Reporte
RF9: Mostrar Datos Generales de Brigada
RF10: Insertar Brigada
RF11: Modificar Brigada
RF12: Eliminar Brigada

RF13: Insertar Técnico
RF14: Modificar Técnico
RF15: Eliminar Técnico
RF16: Insertar Área
RF17: Modificar Área
RF18: Eliminar Área
RF19: Insertar Edificio
RF20: Modificar Edificio
RF21: Eliminar Edificio
RF22: Insertar Local
RF23: Modificar Local
RF24: Eliminar Local
RF25: Insertar Equipo
RF26: Modificar Equipo
RF27: Eliminar Equipo
RF28: Ubicar Equipo en Local

2.2 Requerimientos no Funcionales.

"Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto más atractivo, usable, rápido o confiable. Normalmente están vinculados a requerimientos funcionales, es decir una vez que sepamos lo que el sistema debe hacer, podemos determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser." [8]

Usabilidad

- Las personas que usarán la aplicación procederán de diversos niveles de escolaridad, por lo que esta tendrá una interfaz sencilla y fácil de manejar.
- El sistema deberá incorporar la funcionalidad de paginado en la segmentación de los resultados.
- El sistema deberá incorporar la funcionalidad de ordenamiento a los listados mostrados.

Apariencia

- Se debe hacer uso de plantillas Web para mejorar la apariencia de las páginas.
- No se debe utilizar tecnología de frames.
- Se debe evitar recargar las páginas con textos, imágenes o gráficos.

- Cada rol tendrá una interfaz diferente con las acciones que este pueda realizar en el sistema.

Seguridad

- Se debe identificar al usuario al realizar una acción sobre el sistema.
- Se deben implementar varios niveles de acceso para los usuarios que correspondan con el rol que desempeñan en la aplicación.
- Se debe garantizar que la información y las funcionalidades del sistema estén disponible para los usuarios según sus niveles de acceso.
- El sistema debe contemplar la protección contra acciones no autorizadas.
- El sistema estará disponible las 24 horas del día, los 7 días de la semana.

Restricciones en el diseño y la implementación

- El sistema será concebido sobre la base de la Metodología RUP, usando la herramienta Visual Paradigm para el modelado de los artefactos que se obtendrán durante el desarrollo del software.
- El sistema se debe implementar utilizando el lenguaje de programación PHP.
- Se debe utilizar como sistema de gestión de bases de datos PostgreSQL.

Software

- Sistema Operativo Windows Advanced Server 2000 o superior, o GNU/Linux.
- Servidor Web Apache 2.0 o superior, y debe estar configurado con la extensión pgsql incluida.
- Navegador Internet Explorer 5.5 o superior, o compatible con Mozilla.

Hardware

Servidor:

- Microprocesador: Intel Pentium III o superior, 450 MHz mínimo.
- Memoria RAM: 512 MB o superior.
- Capacidad del disco duro: 50 GB o superior.

PC Cliente:

- Microprocesador: Intel Pentium II o superior, 450 MHz mínimo.
- Memoria RAM: 128 MB mínimo.

2.3 Diagrama de Casos de Uso del Sistema.

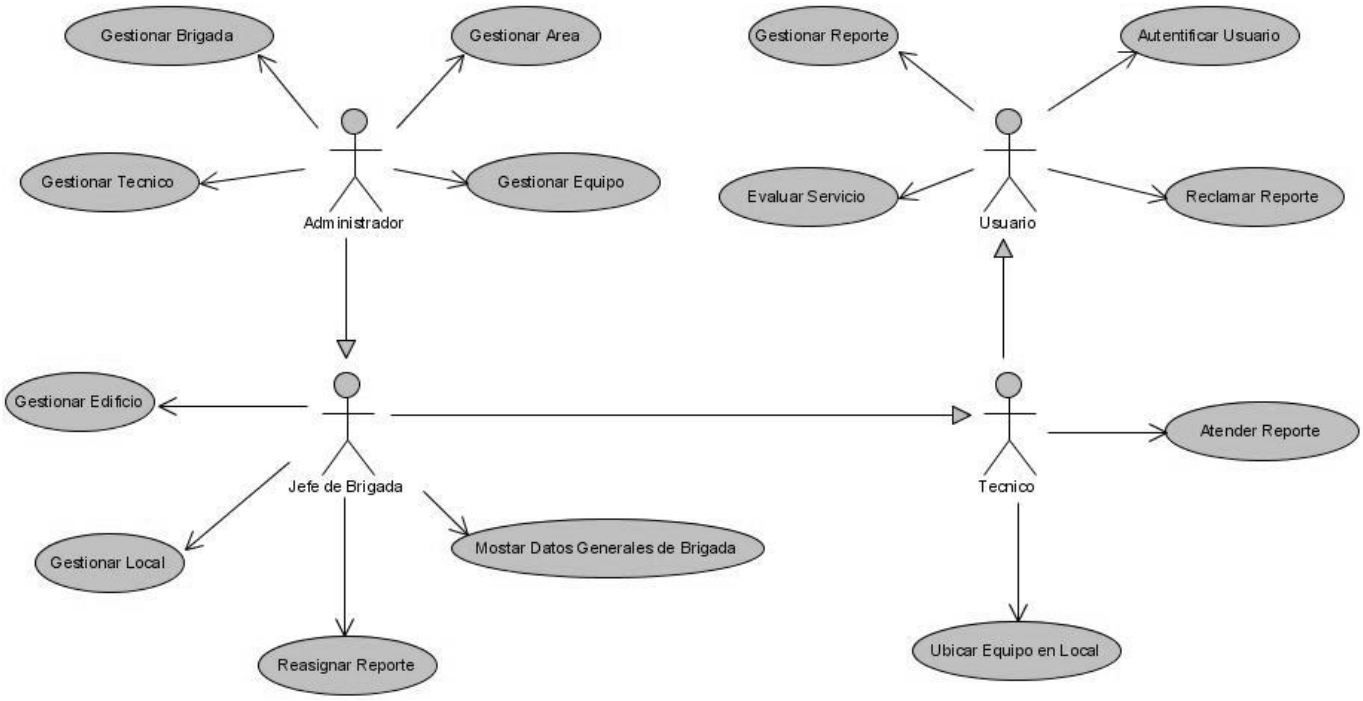


Figura 2.3.1 Diagrama de casos de uso del sistema.

2.4 Descripción de los Casos de Uso del Sistema.

Tabla 2.4.1 Descripción de los casos de uso del sistema. Autentificar Usuario.

Caso de Uso:	**Autentificar Usuario**
Actores:	Usuario (inicia)
Resumen:	El usuario que es el que inicia el caso de uso llega a la aplicación y para poder utilizarla, es decir, trabajar en ella tiene que poner su usuario y contraseña, finalizando así el caso de uso.
Precondiciones:	
Referencias:	RF1
Prioridad:	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario introduce su usuario y contraseña.	2. El sistema verifica que no existan campos vacios. 3. El sistema verifica que los datos del usuario sean correctos. 4. El sistema carga el perfil de usuario

	correspondiente en dependencia de los privilegios del usuario, finalizando así el CU.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	2.1 Si quedó algún campo vacío, el sistema muestra el mensaje “Debe introducir usuario y contraseña del dominio UCI”, y retorna a la acción 1. 3.1 Si los datos del usuario no son correctos el sistema muestra el mensaje “Error en Usuario o Contraseña”, y retorna a la acción 1.
Prototipo de Interfaz	
Poscondiciones:	El sistema deja abierta la sesión del usuario que se registró.

Tabla 2.4.2 Descripción de los casos de uso del sistema. Gestionar Reporte.

Caso de Uso:	**Gestionar Reporte**
Actores:	Usuario (inicia)
Resumen:	El CU se inicia cuando el usuario va a realizar algunas de las siguientes operaciones: 1. Crear Reporte: es cuando el usuario realiza una solicitud de servicio, finalizando así el CU. 2. Buscar y Visualizar Reporte: el usuario busca un reporte a partir del número de solicitud, finalizando así el CU. 3. Cancelar Reporte: es cuando el usuario cancela una solicitud realizada, finalizando así el CU.
Precondiciones:	Que el usuario esté autentificado en el sistema.
Referencias:	RF2, RF3 y RF4
Prioridad:	Crítica
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción Solicitud de Servicio.	2. El sistema muestra una interfaz con las opciones:

3. El usuario selecciona la acción que desea realizar.	➢ Realizar solicitud de servicio. ➢ Buscar solicitud de servicio. ➢ Cancelar solicitud de servicio. 4. El sistema, en dependencia de la acción solicitada por el usuario, muestra la interfaz correspondiente: ➢ Crear Reporte: ir al Escenario "Crear Reporte". ➢ Buscar y Visualizar Reporte: ir al Escenario "Buscar y Visualizar Reporte". ➢ Cancelar Reporte: ir al Escenario "Cancelar Reporte".
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prototipo de Interfaz	
Flujo Normal de Eventos	
Escenario "Crear Reporte"	
Acción del Actor	**Respuesta del Sistema**
2. El usuario introduce los datos de la solicitud: - Área - Edificio - Local - Equipo - Descripción del problema - Horario de atención	1. El sistema muestra la interfaz para Crear Reporte. 3. El sistema verifica que no existan campos vacios. 4. El sistema verifica que no exista la solicitud. 5. El sistema inserta los datos de la solicitud en la BD y la dirección IP desde donde se

	realizó, le informa al usuario que la misma se realizó correctamente y envía un correo al mismo con el número de la solicitud y el nombre del técnico encargado de atenderla, finalizando así el caso de uso.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3.1 Si quedó algún campo vacío, el sistema muestra el mensaje “Todos los campos son obligatorios”, y retorna a la acción 2. 4.1 Si existe la solicitud el sistema informa al usuario que esta fue realizada y muestra los datos de la misma.
Prototipo de Interfaz	
Flujo Normal de Eventos	
Escenario “Buscar y Visualizar Reporte”	
Acción del Actor	**Respuesta del Sistema**
2. El usuario introduce el número del reporte.	1. El sistema muestra la interfaz para Buscar y Visualizar Reporte, solicitando el número de la solicitud que desea visualizar. 3. El sistema busca y visualiza el reporte que coincide con el parámetro de búsqueda, finalizando así el CU.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prototipo de Interfaz	
Flujo Normal de Eventos	
Escenario “Cancelar Reporte”	
Acción del Actor	**Respuesta del Sistema**

2. El usuario introduce el número de la solicitud.	1. El sistema muestra la interfaz para Cancelar Reporte, solicitando el número de la solicitud que desea cancelar. 3. El sistema verifica que exista la solicitud. 4. El sistema elimina la solicitud de la BD, finalizando así el CU.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3.1 Si no existe la solicitud, el sistema muestra el mensaje “Esa solicitud no existe”, y retorna a la acción 1.
Prototipo de Interfaz	
Poscondiciones:	En dependencia de la acción del usuario: Se crea un nuevo reporte. Se cancela un reporte existente. Se actualiza la BD del sistema.

Tabla 2.4.3 Descripción de los casos de uso del sistema. Reclamar Reporte.

Caso de Uso:	**Reclamar Reporte**
Actores:	Usuario (inicia)
Resumen:	El CU se inicia cuando el usuario selecciona la opción Reclamar Reporte, dando la posibilidad de hacer una reclamación ya sea porque el técnico se demoró más tiempo del previsto para ir a solucionar el problema o porque el servicio prestado por el técnico no fue bueno, finalizando así el CU.
Precondiciones:	Que el usuario se haya autentificado en el sistema.
Referencias:	RF5
Prioridad:	Critico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción Reclamar Reporte. 3. El usuario introduce los datos de la reclamación.	2. El sistema muestra una interfaz con un formulario con los datos de la reclamación (número de solicitud y descripción).

	4. El sistema verifica que no hayan campos vacíos. 5. El sistema inserta la reclamación en la BD, finalizando así el CU.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	4.1 Si quedo algún campo vació, el sistema emite un mensaje “Todos los campos son obligatorios” y retorna a la acción 3.
Prototipo de Interfaz	
Poscondiciones	Se actualiza la BD del sistema.

Tabla 2.4.4 Descripción de los casos de uso del sistema. Evaluar Servicio.

Caso de Uso:	**Evaluar Servicio**
Actores:	Usuario (inicia)
Resumen:	El CU se inicia cuando el usuario selecciona la opción Evaluar Servicio, dando la posibilidad de emitir una evaluación del servicio prestado por el técnico al darle solución al reporte, finalizando así el CU.
Precondiciones:	Que el usuario se haya autentificado en el sistema.
Referencias:	RF6
Prioridad:	Critico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción Evaluar Servicio.	2. El sistema muestra una interfaz con un formulario con los datos de la evaluación (número de solicitud, evaluación (bien, regular, mal), descripción).
3. El usuario introduce los datos de la evaluación.	4. El sistema verifica que no hayan campos vacíos. 5. El sistema inserta la evaluación en la BD, finalizando así el CU.
Prototipo de Interfaz	

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	4.1 Si quedo algún campo vació, el sistema emite un mensaje “Todos los campos son obligatorios” y retorna a la acción 2.
Prototipo de Interfaz	
Poscondiciones	Se actualiza la BD del sistema.

Tabla 2.4.5 Descripción de los casos de uso del sistema. Atender Reporte.

Caso de Uso:	**Atender Reporte**
Actores:	Técnico (inicia)
Resumen:	El CU se inicia cuando el técnico selecciona la opción Atender Reporte, mostrando los reportes del mismo que no se han solucionado y permitiendo atenderlos, finalizando así el CU.
Precondiciones:	Que el técnico se haya autentificado en el sistema.
Referencias:	RF7
Prioridad:	Critico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El técnico selecciona la opción Atender Reporte. 3. El técnico selecciona el reporte que desea atender. 5. El técnico introduce los datos.	2. El sistema muestra una interfaz donde aparecen los reportes que el técnico debe atender y que no han sido solucionados, permitiendo atenderlos. 4. El sistema muestra una interfaz con un formulario para que el técnico entre los datos técnicos del reporte y el estado en que queda el mismo después de atendido. 6. El sistema verifica que no hayan campos vacíos. 7. El sistema actualiza los datos del reporte en la BD, finalizando así el CU.
Prototipo de Interfaz	
Flujos Alternos	

Acción del Actor	Respuesta del Sistema
	6.1 Si quedo algún campo vació, el sistema emite un mensaje “Todos los campos son obligatorios” y retorna a la acción 4.
Prototipo de Interfaz	
Poscondiciones	Se actualiza la BD del sistema.

Tabla 2.4.6 Descripción de los casos de uso del sistema. Reasignar Reporte.

Caso de Uso:	**Reasignar Reporte**
Actores:	Jefe de Brigada (inicia)
Resumen:	El CU se inicia cuando el jefe de brigada selecciona la opción Reasignar Reporte, permitiendo reasignarle un reporte de un técnico a otro que el determine, finalizando así el CU.
Precondiciones:	Que el jefe de brigada se haya autenticado en el sistema.
Referencias:	RF8
Prioridad:	Critico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El jefe de brigada selecciona la opción Reasignar Reporte.	2. El sistema muestra una interfaz donde aparecen los técnicos de la brigada que él atiende con la cantidad de reportes que tienen sin atender cada uno y un vínculo por cada técnico que permite visualizar los datos de los reportes pendientes del mismo.
3. El jefe de brigada teniendo en cuenta la cantidad de reportes sin atender de sus técnicos selecciona la opción Ver de un técnico especifico.	4. El sistema muestra una interfaz donde aparecen los datos de los reportes sin atender del técnico seleccionado, permitiendo reasignar los mismos.
5. El jefe de brigada selecciona el reporte que desea reasignar.	6. El sistema muestra una interfaz con todos los técnicos de la brigada, para que el jefe de brigada seleccione el técnico al que se le va a asignar el reporte.
7. El jefe de brigada selecciona el técnico y selecciona la opción Reasignar Reporte.	8. El sistema actualiza los datos del reporte en

	la BD, finalizando así el CU.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prototipo de Interfaz	
Poscondiciones	Se actualiza la BD del sistema.

Tabla 2.4.7 Descripción de los casos de uso del sistema. Mostrar Datos Generales de Brigada.

Caso de Uso:	**Mostrar Datos Generales de Brigada**
Actores:	Jefe de Brigada (inicia)
Resumen:	El CU se inicia cuando el jefe de brigada selecciona la opción Mostrar Datos de Brigada, visualizándose gráficamente el comportamiento de la brigada que él atiende teniendo en cuenta diferentes parámetros y permitiendo además ver una vista detallada del trabajo realizado por cada uno de los técnicos de su brigada, finalizando así el CU.
Precondiciones:	Que el jefe de brigada se haya autentificado en el sistema.
Referencias:	RF9
Prioridad:	Critico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El jefe de brigada selecciona la opción Mostrar Datos de Brigada.	2. El sistema muestra una interfaz donde se visualiza un gráfico que muestra la cantidad de reportes de la brigada que dirige este jefe de brigada que no se han atendido, los que están pendientes y las reclamaciones que se le han realizado a los técnicos de su brigada, permitiendo además ver de forma detallada los datos especificados por cada técnico de la brigada.
3. El jefe de brigada selecciona la opción Vista Detallada.	4. El sistema muestra una interfaz donde se visualiza la foto del técnico, su nombre, así como los reportes sin atender, los reportes

5. El jefe de brigada selecciona la opción Ver.	pendientes y las reclamaciones de cada uno de ellos, brindando la opción de seleccionar el número mostrado en cada caso para ver los datos completos de las solicitudes que se encuentran en el estado seleccionado. 6. El sistema muestra los datos de los reportes, finalizando así el CU.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prototipo de Interfaz	
Poscondiciones	

Tabla 2.4.8 Descripción de los casos de uso del sistema. Gestionar Brigada.

Caso de Uso:	**Gestionar Brigada**
Actores:	Administrador (inicia)
Resumen:	El CU se inicia cuando el administrador selecciona la opción Gestionar Brigada, mostrando las brigadas existentes y dando la posibilidad de Modificar y Eliminar estas brigadas o Adicionar una nueva, finalizando así el CU.
Precondiciones:	Que el administrador se haya autenticado en el sistema.
Referencias	RF10, RF11 y RF12
Prioridad	Critico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción de Gestionar Brigada.	2. El sistema muestra una interfaz con todas las brigadas existentes brindando la posibilidad de realizar una de las siguientes acciones: ➢ Adicionar Brigada: Ir al escenario “Adicionar Brigada”. ➢ Modificar Brigada: Ir al escenario “Modificar Brigada”. ➢ Eliminar Brigada: Ir al escenario “Eliminar Brigada”.

Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prototipo de Interfaz	
Flujo Normal de Eventos	
Escenario “ Adicionar Brigada”	
Acción del Actor	**Respuesta del Sistema**
2. El administrador introduce los datos de la brigada.	1. El sistema muestra la interfaz para Adicionar Brigada, con un formulario con los datos de la Brigada (Nombre de la brigada). 3. El sistema verifica que no existan campos vacíos. 4. El sistema verifica que la brigada no exista. 5. El sistema inserta la brigada en la BD, finalizando así el CU.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3.1 Si quedó algún campo vacío, el sistema emite un mensaje “Todos los campos son obligatorios” y retorna a la acción 1. 4.1 Si la brigada ya existe, el sistema emite el mensaje: “La brigada ya existe en la BD” y finaliza el CU.
Prototipo de Interfaz	
Flujo Normal de Eventos	
Escenario “Modificar Brigada”	
Acción del Actor	**Respuesta del Sistema**
	1. El sistema muestra la interfaz de Modificar Brigada, dándole la posibilidad al administrador

2. El administrador introduce los nuevos datos.	de modificar los datos de la brigada (nombre de la brigada). 3. El sistema verifica que no existan campos vacíos. 4. El sistema verifica que esa brigada no exista. 5.El sistema actualiza la BD, finalizando así el CU
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3.1 Si quedo algún campo vació, el sistema emite un mensaje “Todos los campos son obligatorios” y retorna a la acción 1. 4.1 Si la brigada existe, el sistema emite un mensaje: “La brigada ya existe en la BD” y retorna a la acción 1.
Prototipo de Interfaz	
Flujo Normal de Eventos	
Sesión “Eliminar Brigada ”	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la brigada que desea eliminar.	2. El sistema elimina la brigada seleccionada de la BD, finalizando así el CU.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Poscondiciones	Se actualiza la BD del sistema

Tabla 2.4.9 Descripción de los casos de uso del sistema. Gestionar Técnico.

Caso de Uso:	**Gestionar Técnico**
Actores:	Administrador (inicia)

Resumen:	El CU se inicia cuando el administrador selecciona la opción Gestionar Técnico, mostrando las brigadas existentes en la BD para a partir de ella poder ver los técnicos que le corresponden a cada una y dando la posibilidad de Modificar y Eliminar estos técnicos o Adicionar uno nuevo, finalizando así el CU.
Precondiciones:	Que el administrador se haya autenticado en el sistema.
Referencias	RF13, RF14 y RF15
Prioridad	Critica

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Gestionar Técnico.	2. El sistema muestra una interfaz con todas las brigadas existentes, al seleccionar una de ellas muestra los técnicos asociados a esta brigada, y en dependencia del técnico que haya escogido y de lo que quiera hacer muestra la interfaz correspondiente: ➢ Adicionar Técnico: Ir al escenario "Adicionar Técnico". ➢ Modificar Técnico: Ir al escenario "Modificar Técnico". ➢ Eliminar Técnico: Ir al escenario "Eliminar Técnico".
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prototipo de Interfaz	
Flujo Normal de Eventos	
Escenario " Adicionar Técnico"	
Acción del Actor	**Respuesta del Sistema**
2. El administrador introduce los datos del técnico.	1. El sistema muestra la interfaz de Adicionar un Técnico, con un formulario con los datos del técnico (usuario y cargo que desempeña). 3. El sistema verifica que no existan campos vacios. 4. El sistema verifica que ese técnico no exista en la BD.

	5. El sistema inserta el técnico en la BD, finalizando así el CU.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3.1 Si quedó algún campo vacío, el sistema emite un mensaje “Todos los campos son obligatorios” y retorna a la acción 1. 4.1 Si el técnico ya existe, el sistema emite el mensaje: “El técnico ya existe en la BD” y finaliza el CU.
Prototipo de Interfaz	
Flujo Normal de Eventos	
Escenario “Modificar Técnico”	
Acción del Actor	**Respuesta del Sistema**
2. El administrador introduce los datos del técnico.	1. El sistema muestra la interfaz de Modificar Técnico, dándole la posibilidad al administrador de escoger el técnico que desea modificar. 3. El sistema verifica que no existan campos vacios. 4. El sistema modifica los datos del técnico en la BD, finalizando así el CU.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3.1 Si quedó algún campo vacío, el sistema emite un mensaje “Todos los campos son obligatorios” y retorna a la acción 1.
Prototipo de Interfaz	
Flujo Normal de Eventos	
Sesión “Eliminar Técnico ”	

Acción del Actor	Respuesta del Sistema
1. El administrador selecciona el técnico que desea eliminar.	2. El sistema elimina el técnico de la BD, finalizando así el CU.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prototipo de Interfaz	
Poscondiciones	Se actualiza la BD del sistema.

Tabla 2.4.10 Descripción de los casos de uso del sistema. Gestionar Área.

Caso de Uso:	**Gestionar Área**
Actores:	Administrador (inicia)
Resumen:	El CU se inicia cuando el administrador selecciona la opción Gestionar Área, mostrando las áreas existentes y dando la posibilidad de Modificar y Eliminar estas áreas o Adicionar una nueva, finalizando así el CU.
Precondiciones:	Que el administrador este autentificado en el sistema.
Referencias	RF16, RF17 y RF18
Prioridad	Critica
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Gestionar Área.	2. El sistema muestra una interfaz con todas las áreas existentes brindando la posibilidad de realizar una de las siguientes acciones: ➢ Adicionar Área: Ir al escenario “Adicionar Área”. ➢ Modificar Área: Ir al escenario “Modificar Área”. ➢ Eliminar Área: Ir al escenario “Eliminar Área”.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Flujo Normal de Eventos	
Escenario “Adicionar Área”	

Acción del Actor	Respuesta del Sistema
2. El administrador introduce los datos del área.	1. El sistema muestra la interfaz para Adicionar Área, con un formulario con los datos del área (Nombre del área). 3. El sistema verifica que no existan campos vacios. 4. El sistema verifica que esa área no exista. 5. El sistema inserta el área en la BD, finalizando así el CU.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3.1 Si quedo algún campo vacio, el sistema emite un mensaje “Todos los campos son obligatorios” y retorna a la acción 1. 4.1 Si el área existe, el sistema muestra el mensaje: “El área ya existe en la BD “, finalizando así el CU.
Flujo Normal de Eventos	
Escenario “Modificar Área”	
Acción del Actor	**Respuesta del Sistema**
2. El administrador introduce los nuevos datos.	1. El sistema muestra la interfaz de Modificar Área, dándole la posibilidad al administrador de modificar los datos del área (nombre del área). 3. El sistema verifica que no existan campos vacios. 4. El sistema verifica que esa área no exista. 5. El sistema actualiza la BD, finalizando así el CU.
Prototipo de Interfaz	
Flujos Alternos	

Acción del Actor	Respuesta del Sistema
	3.1 Si quedo algún campo vacio, el sistema emite un mensaje “Todos los campos son obligatorios” y retorna a la acción 1. 4.1 Si el área existe, el sistema muestra el mensaje: “El área ya existe en la BD”, finalizando así el CU
Flujo Normal de Eventos	
Escenario “Eliminar Área”	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador selecciona el área a eliminar.	2. El sistema elimina esa área en la BD, finalizando así el CU.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	Respuesta del Sistema
Prototipo de Interfaz	
Poscondiciones	Se actualiza la BD del sistema.

Tabla 2.4.11 Descripción de los casos de uso del sistema. Gestionar Edificio.

Caso de Uso:	**Gestionar Edificio**
Actores:	Jefe de Brigada (inicia)
Resumen:	El CU se inicia cuando el jefe de brigada selecciona la opción Gestionar Edificio, mostrando las áreas existentes en la BD para a partir de ella poder ver los edificios que le corresponden a cada una y dando la posibilidad de Modificar y Eliminar estos edificios o Adicionar uno nuevo, finalizando así el CU.
Precondiciones:	Que el jefe de brigada se haya autenticado en el sistema.
Referencias	RF19, RF20 y RF21
Prioridad	Critica
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El jefe de brigada selecciona la opción Gestionar	2. El sistema muestra una interfaz con todas las áreas existentes, al seleccionar una de ellas, el sistema muestra los edificios asociados a esta y

Edificio.	en dependencia del edificio que haya escogido y de la operación que desee realizar muestra la interfaz correspondiente: ➢ Adicionar Edificio: Ir al escenario "Adicionar Edificio". ➢ Modificar Edificio: Ir al escenario "Modificar Edificio". ➢ Eliminar Edificio: Ir al escenario "Eliminar Edificio".
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prototipo de Interfaz	
Flujo Normal de Eventos	
Escenario " Adicionar Edificio"	
Acción del Actor	**Respuesta del Sistema**
2. El jefe de brigada introduce los datos del edificio.	1. El sistema muestra la interfaz de Adicionar un Edificio, con un formulario con los datos del edificio (Área y Nombre del Edificio). 3. El sistema verifica que no existan campos vacios. 4. El sistema verifica que ese edificio no exista en esa área. 5. El sistema inserta el edificio en la BD, finalizando así el CU.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3.1 Si quedó algún campo vacío, el sistema emite un mensaje "Todos los campos son obligatorios" y retorna a la acción 1. 4.1 Si el edificio ya existe, el sistema emite el mensaje: "El edificio ya existe en la BD", finalizando así el CU.

Prototipo de Interfaz	
Flujo Normal de Eventos	
Escenario "Modificar Edificio"	
Acción del Actor	**Respuesta del Sistema**
2. El jefe de brigada introduce los datos del edificio.	1. El sistema muestra la interfaz de Modificar Edificio, dándole la posibilidad al Jefe de Brigada de escoger el edificio que desea modificar. 3. El sistema verifica que no existan campos vacios. 4. El sistema modifica el edificio en la BD finalizando así el CU.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3.1 Si quedó algún campo vacío, el sistema emite un mensaje "Todos los campos son obligatorios" y retorna a la acción 1.
Prototipo de Interfaz	
Flujo Normal de Eventos	
Sesión "Eliminar Edificio "	
Acción del Actor	**Respuesta del Sistema**
1. El jefe de brigada selecciona el edificio que desea eliminar.	2. El sistema elimina el edificio de la BD, finalizando así el CU.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prototipo de Interfaz	
Poscondiciones	Se actualiza la BD del sistema.

Tabla 2.4.12 Descripción de los casos de uso del sistema. Gestionar Local.

Caso de Uso:	**Gestionar Local**
Actores:	Jefe de Brigada (inicia)

Resumen:	El CU se inicia cuando el jefe de brigada selecciona la opción Gestionar Local, mostrando las áreas existentes en la BD y los edificios que le corresponden para a partir de ahí obtener el local y dar la posibilidad de Modificar y Eliminar estos locales o Adicionar uno nuevo, finalizando así el CU.
Precondiciones:	Que el jefe de brigada se haya autenticado en el sistema.
Referencias	RF22, RF23 y RF24
Prioridad	Critica

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El jefe de brigada selecciona la opción de Gestionar Local.	2. El sistema muestra una interfaz con todas las áreas existentes, al seleccionar una de ellas muestra los edificios asociados a esta área, una vez seleccionado uno de estos edificios se van a mostrar todos los locales que tiene el edificio y en dependencia del local que haya escogido y de lo que quiera hacer muestra la interfaz correspondiente: ➢ Adicionar Local: Ir al escenario “Adicionar Local”. ➢ Modificar Local: Ir al escenario “Modificar Local”. ➢ Eliminar Local: Ir al escenario “Eliminar Local”.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prototipo de Interfaz	
Flujo Normal de Eventos	
Escenario “ Adicionar Local”	
Acción del Actor	**Respuesta del Sistema**
2. El jefe de brigada introduce los datos del local.	1. El sistema muestra la interfaz de Adicionar un Local, con un formulario con los datos del local (Área, Edificio y Nombre del Local). 3. El sistema verifica que no existan campos vacios.

	4. El sistema verifica que ese local no existe en la BD. 5. El sistema inserta el local en la BD, finalizando así el CU.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3.1 Si quedó algún campo vacío, el sistema emite un mensaje “Todos los campos son obligatorios” y retorna a la acción 1. 4.1 Si el local ya existe, el sistema emite el mensaje: “El local ya existe en la BD” y finaliza el CU.
Prototipo de Interfaz	
Flujo Normal de Eventos	
Escenario “Modificar Local”	
Acción del Actor	**Respuesta del Sistema**
2. El jefe de brigada introduce los datos del local.	1. El sistema muestra la interfaz de Modificar Local, dándole la posibilidad al jefe de brigada de escoger el local que desea modificar. 3. El sistema verifica que no existan campos vacios. 4. El sistema modifica el local en la BD, finalizando así el CU.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3.1 Si quedó algún campo vacío, el sistema emite un mensaje “Todos los campos son obligatorios” y retorna a la acción 1.
Prototipo de Interfaz	
Flujo Normal de Eventos	

Sesión “Eliminar Local ”	
Acción del Actor	**Respuesta del Sistema**
1. El jefe de brigada selecciona el local que desea eliminar.	2. El sistema elimina el local de la BD, finalizando así el CU.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prototipo de Interfaz	
Poscondiciones	Se actualiza la BD del sistema.

Tabla 2.4.13 Descripción de los casos de uso del sistema. Gestionar Equipo.

Caso de Uso:	**Gestionar Equipo**
Actores:	Administrador (inicia)
Resumen:	El CU se inicia cuando el administrador selecciona la opción Gestionar Equipo, mostrando los equipos existentes y dando la posibilidad de Modificar y Eliminar estos equipos o Adicionar uno nuevo, finalizando así el CU.
Precondiciones:	Que el administrador se haya autenticado.
Referencias	RF25, RF26 y RF27
Prioridad	Critico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Gestionar Equipo.	2. El sistema muestra una interfaz con todos los equipos existentes brindando la posibilidad de realizar una de las siguientes acciones: ➢ Adicionar Equipo: Ir al escenario “Adicionar Equipo”. ➢ Modificar Equipo: Ir al escenario “Modificar Equipo”. ➢ Eliminar Equipo: Ir al escenario “Eliminar Equipo”.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**

Prototipo de Interfaz	
Flujo Normal de Eventos	
Escenario “ Adicionar Equipo”	
Acción del Actor	**Respuesta del Sistema**
2. El administrador introduce los datos del equipo.	1. El sistema muestra la interfaz para Adicionar Equipo, con un formulario con los datos del Equipo (Nombre del equipo). 3. El sistema verifica que no existan campos vacíos. 4. El sistema verifica que el equipo no exista. 5. El sistema inserta el equipo en la BD, finalizando así el CU.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3.1 Si quedó algún campo vacío, el sistema emite un mensaje “Todos los campos son obligatorios” y retorna a la acción 1. 4.1 Si el equipo ya existe, el sistema emite el mensaje: “El equipo ya existe en la BD” y finaliza el CU.
Prototipo de Interfaz	
Flujo Normal de Eventos	
Escenario “Modificar Equipo”	
Acción del Actor	**Respuesta del Sistema**
2. El administrador introduce los nuevos datos.	1. El sistema muestra la interfaz de Modificar Equipo, dándole la posibilidad al administrador de modificar los datos del equipo (nombre del equipo). 3. El sistema verifica que no existan campos vacíos. 4. El sistema verifica que ese equipo no exista.

	5. El sistema actualiza la BD, finalizando así el CU
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3.1 Si quedo algún campo vació, el sistema emite un mensaje “Todos los campos son obligatorios” y retorna a la acción 1. 4.1 Si el equipo existe, el sistema emite un mensaje: “El Equipo ya existe en la BD”, finalizando así el CU.
Prototipo de Interfaz	
Flujo Normal de Eventos	
Escenario “Eliminar Equipo”	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona el equipo que desea eliminar.	2. El sistema elimina el equipo seleccionado de la BD, finalizando así el CU.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Poscondiciones	Se actualiza la BD del sistema

Tabla 2.4.14 Descripción de los casos de uso del sistema. Ubicar Equipo en Local.

Caso de Uso:	**Ubicar Equipo en Local**
Actores:	Técnico (inicia)
Resumen:	El CU se inicia cuando el técnico selecciona la opción Ubicar Equipo en Local, permitiéndole ubicar equipos en un local determinado, finalizando así el CU.
Precondiciones:	Que el técnico se haya autentificado en el sistema.
Referencias:	RF28
Prioridad:	Critico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El técnico selecciona la opción Ubicar Equipo en Local. 3. El técnico selecciona el local y el equipo.	2. El sistema muestra una interfaz para seleccionar el local, así como el equipo que va a ubicar en el mismo. 4. El sistema verifica que ese equipo no este ubicado en ese local. 5. El sistema inserta en la BD la información de que el equipo está ubicado en ese local, finalizando así el CU.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	4.1 Si esta ubicado en el local, el sistema emite un mensaje “Ese equipo ya existe en ese local” y retorna a la acción 2.
Prototipo de Interfaz	
Poscondiciones	Se actualiza la BD del sistema.

2.5 Descripción de Estilos Arquitectónicos y Patrones de Diseño.

Estilos Arquitectónicos.

Los estilos arquitectónicos definen las reglas generales de organización en términos de un patrón y las restricciones en la forma y la estructura de un grupo de sistemas de software. Tienen gran importancia ya que permiten sintetizar estructuras de soluciones y definen los posibles patrones de las aplicaciones.

Entre los estilos arquitectónicos más conocidos se encuentran:

1. Estilos de Flujo de Datos
   a. Tubería y filtros
2. Estilos Centrados en Datos
   a. Arquitecturas de Pizarra o Repositorio
3. Estilos de Llamada y Retorno
   a. Modelo-Vista-Controlador (MVC)
   b. Arquitecturas en Capas

c. Arquitecturas Orientadas a Objetos
d. Arquitecturas Basadas en Componentes

4. Estilos de Código Móvil
   a. Arquitectura de Máquinas Virtuales
5. Estilos heterogéneos
   a. Sistemas de control de procesos
   b. Arquitecturas Basadas en Atributos
6. Estilos Peer-to-Peer
   a. Arquitecturas Basadas en Eventos
   b. Arquitecturas Orientadas a Servicios
   c. Arquitecturas Basadas en Recursos

Debido a la utilización del framework de PHP CodeIgniter, y por las ventajas de su uso en la implementación de la aplicación se hace uso del patrón arquitectónico Modelo Vista Controlador.

Modelo Vista Controlador

Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos:

- Modelo: representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.
- Vista: presenta el modelo en un formato adecuado para interactuar con el usuario.
- Controlador: responde a eventos o acciones del usuario e invoca cambios en el modelo y la vista.

El patrón MVC se ve frecuentemente en aplicaciones Web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y el controlador representa la lógica de negocio. En la implementación de la aplicación se hará uso de este patrón arquitectónico. Ver Anexo 1.

2.6 Diagrama de Clases del Diseño.

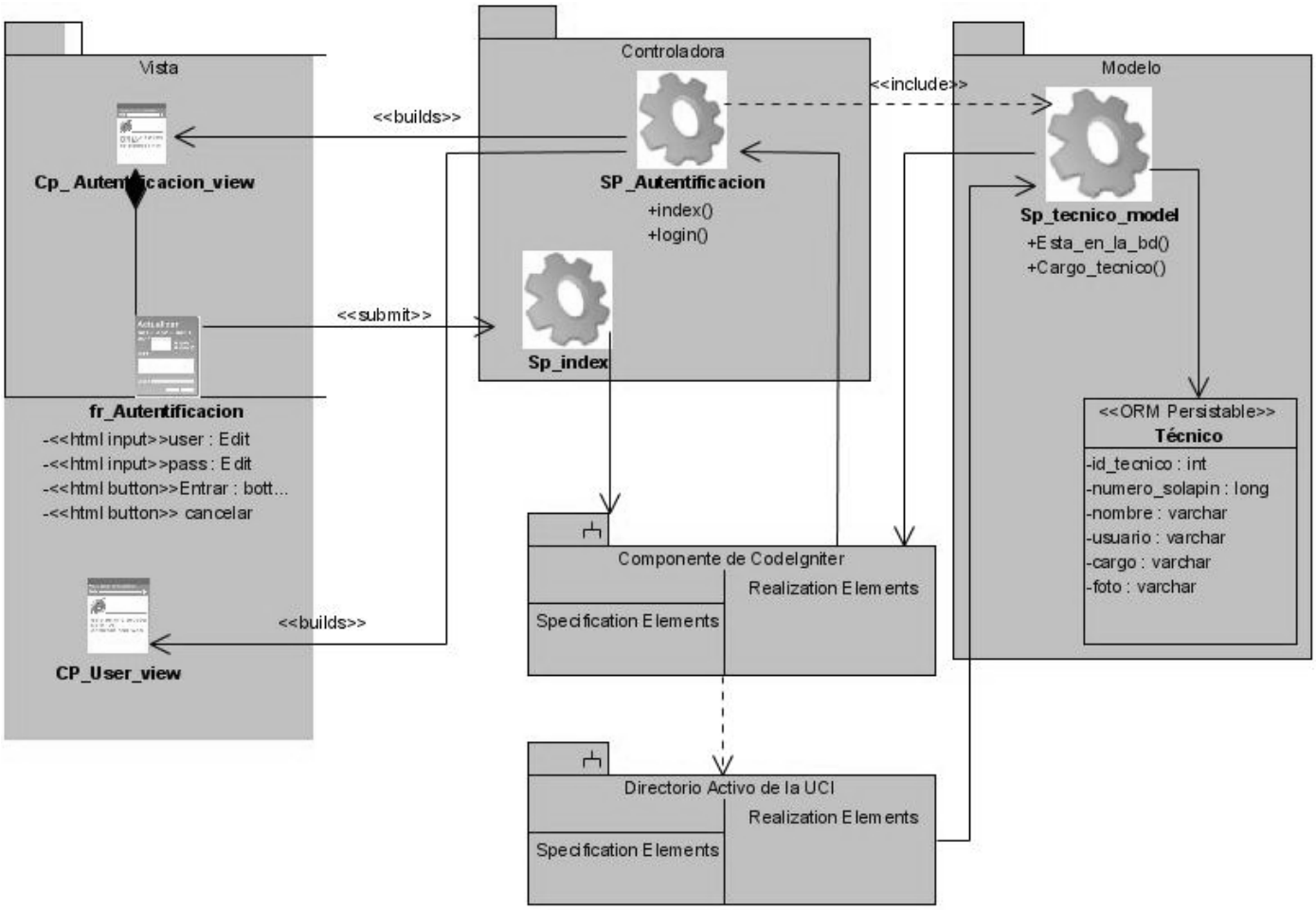


Figura 1.6.1 Diagrama de Clases Web. Autentificar Usuario.

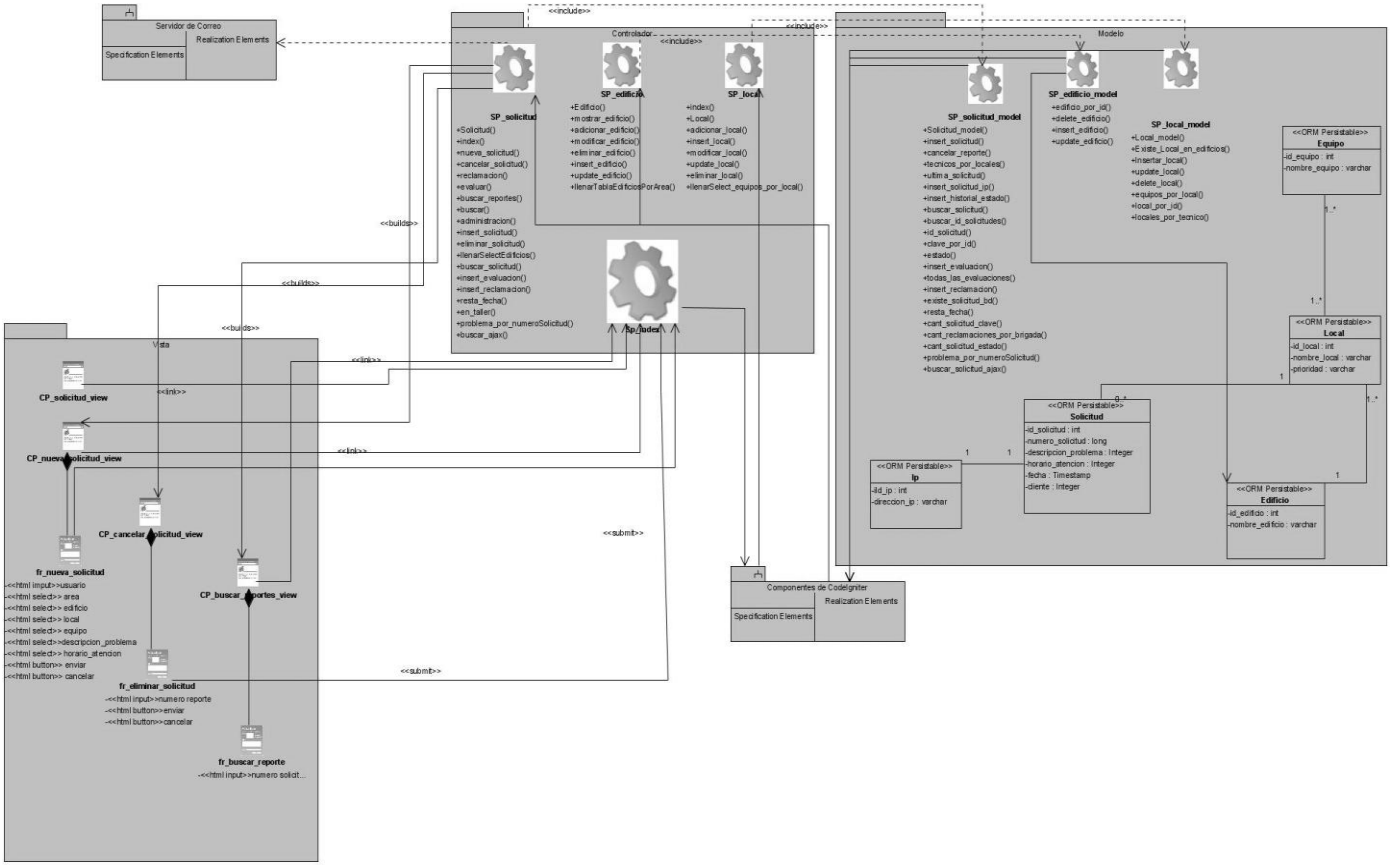


Figura 2.6.2 Diagrama de Clases Web. Gestionar Reporte.

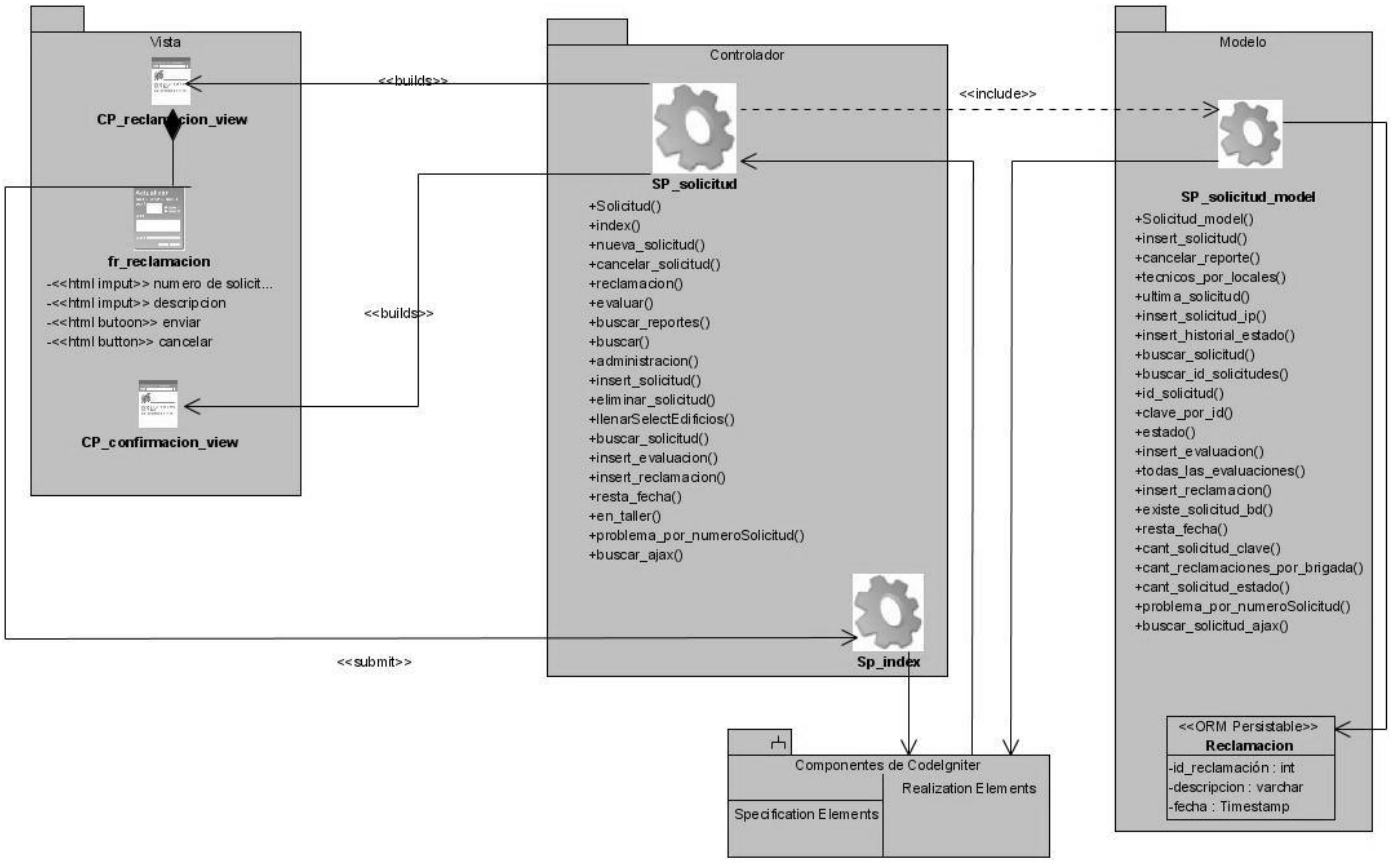


Figura 2.6.3 Diagrama de Clases Web. Reclamar Reporte.

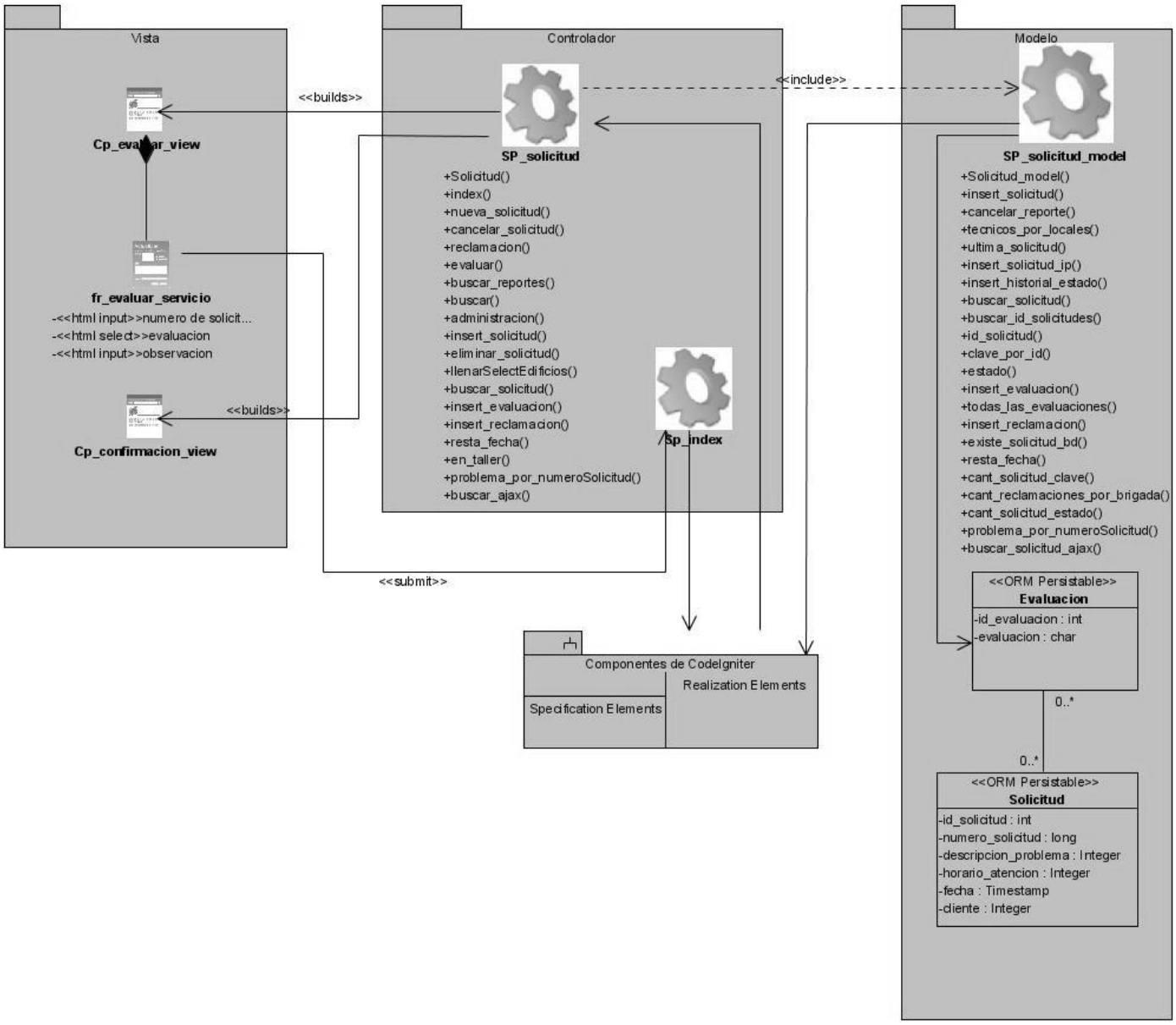


Figura 2.6.4 Diagrama de Clases Web. Evaluar Servicio.

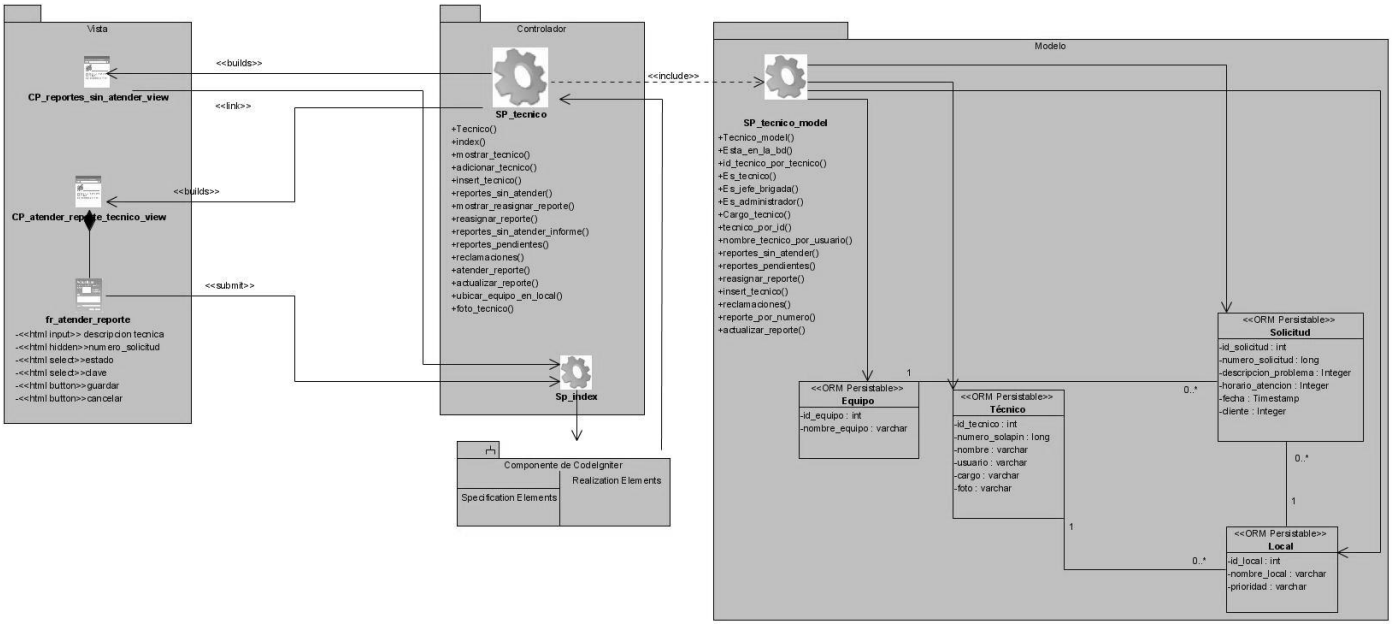


Figura 2.6.5 Diagrama de Clases Web. Atender Reporte.

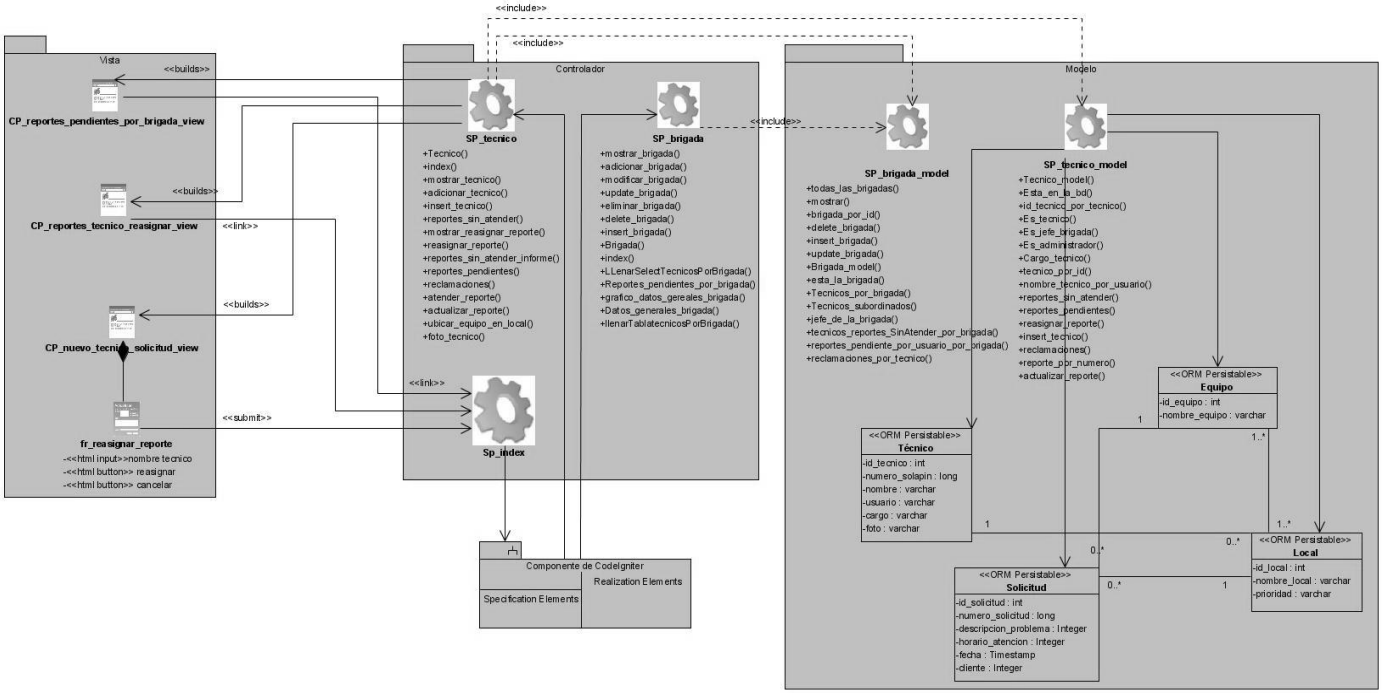


Figura 2.6.6 Diagrama de Clases Web. Reasignar Reporte.

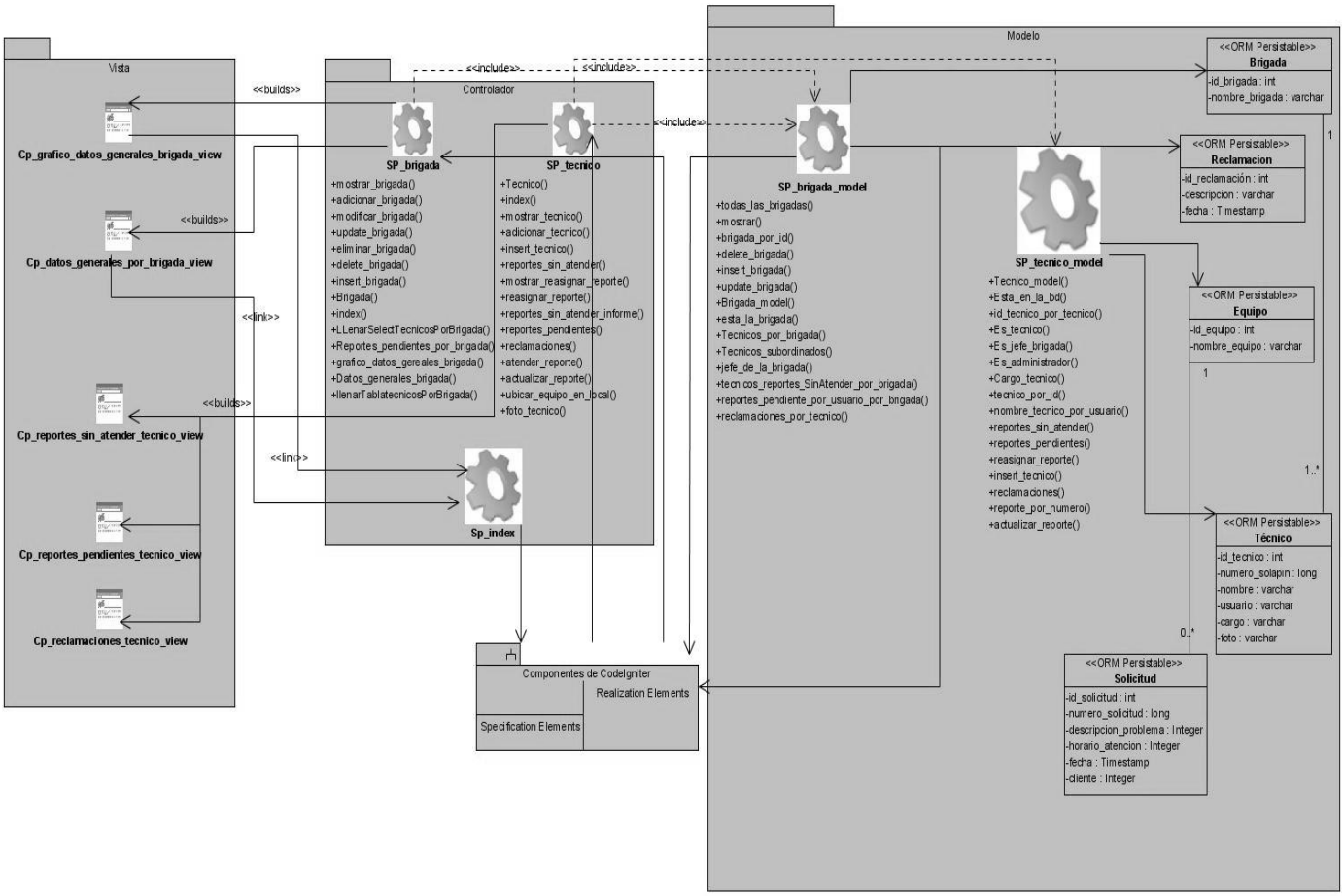


Figura 2.6.7 Diagrama de Clases Web. Mostar Datos Generales de Brigada.

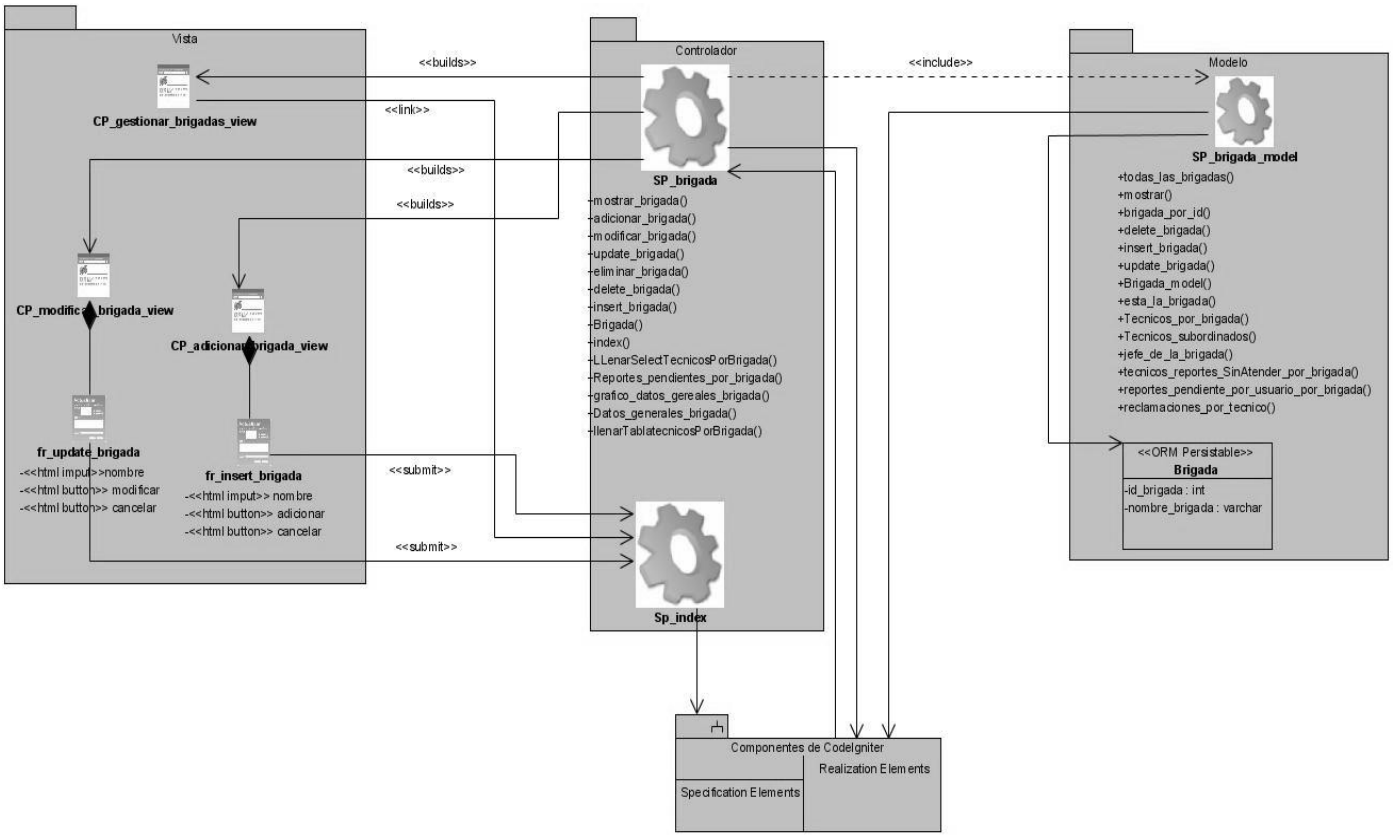


Figure 2.6.8 Diagrama de Clases Web. Gestionar Brigada.

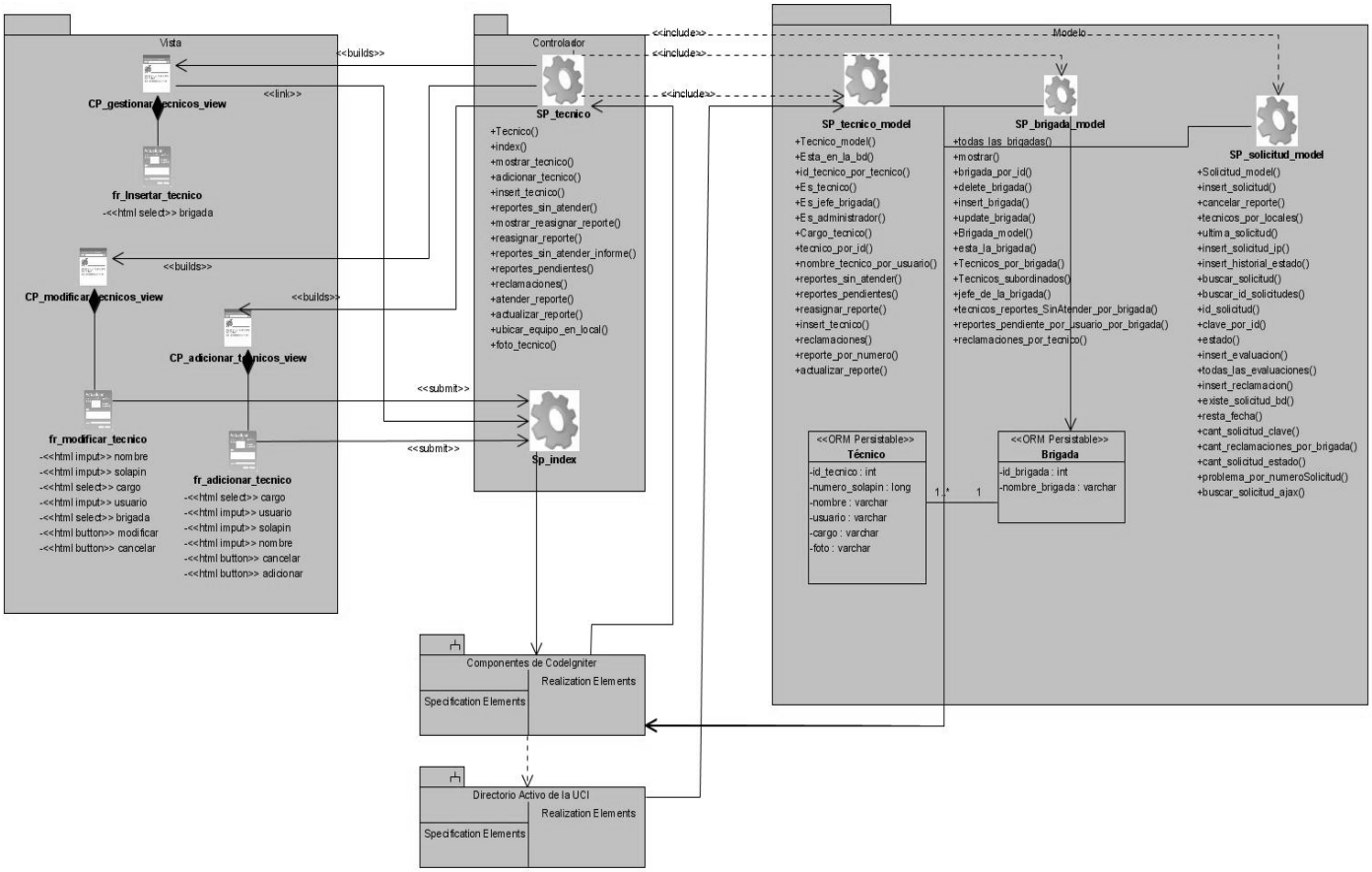


Figura 2.6.9 Diagrama de Clases Web. Gestionar Técnico.

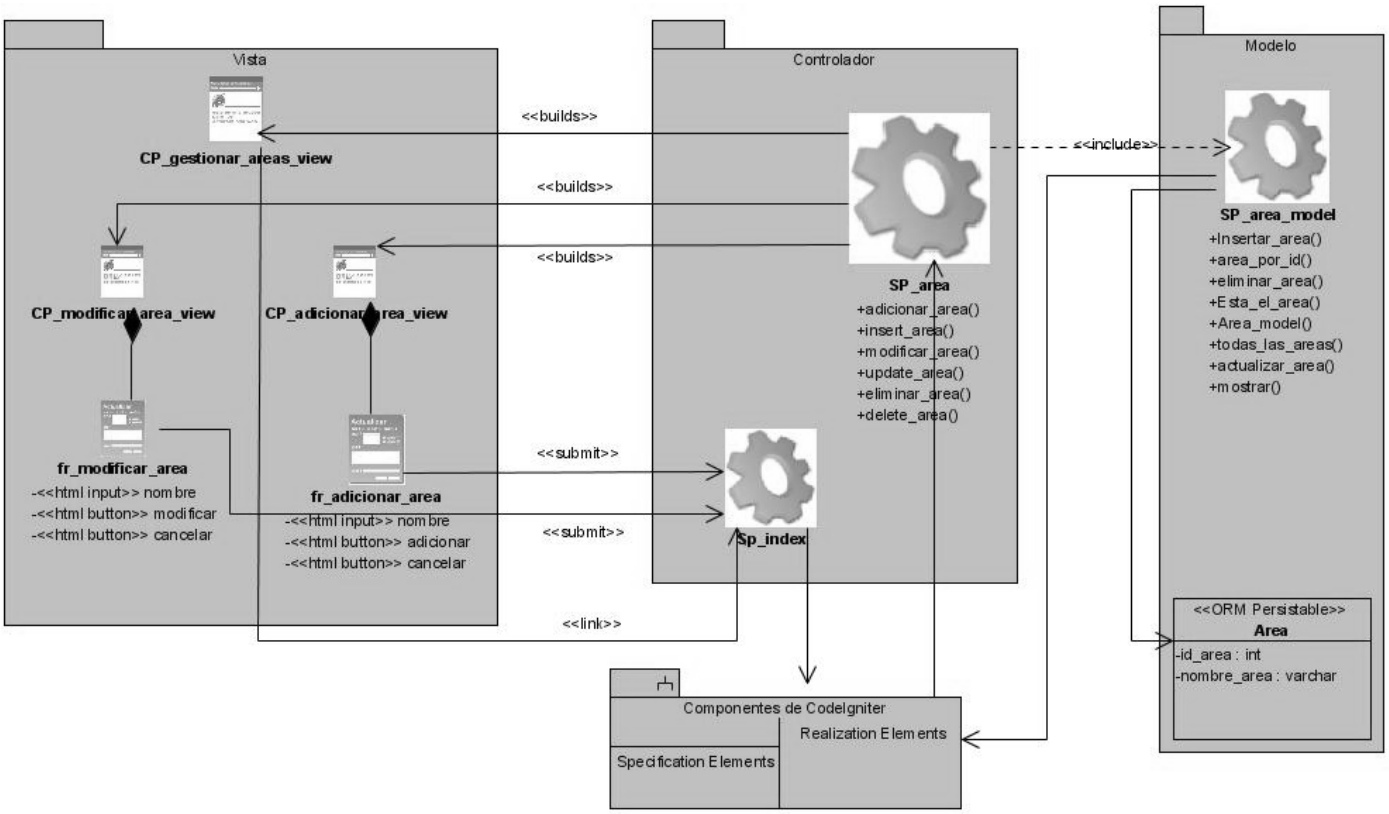


Figura 2.6.10 Diagrama de Clases Web. Gestionar Área.

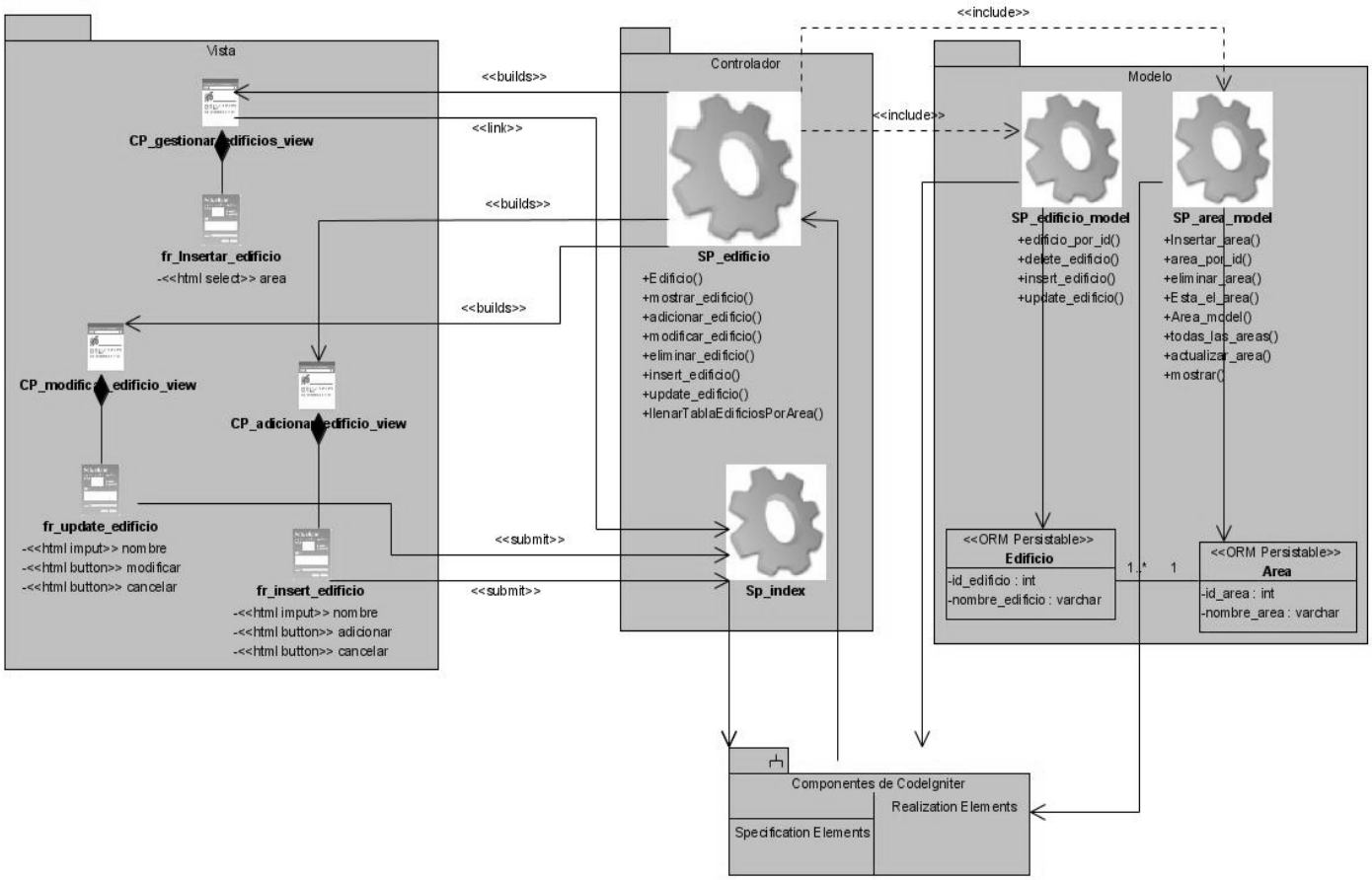


Figura 2.6.11 Diagrama de Clases Web. Gestionar Edificio.

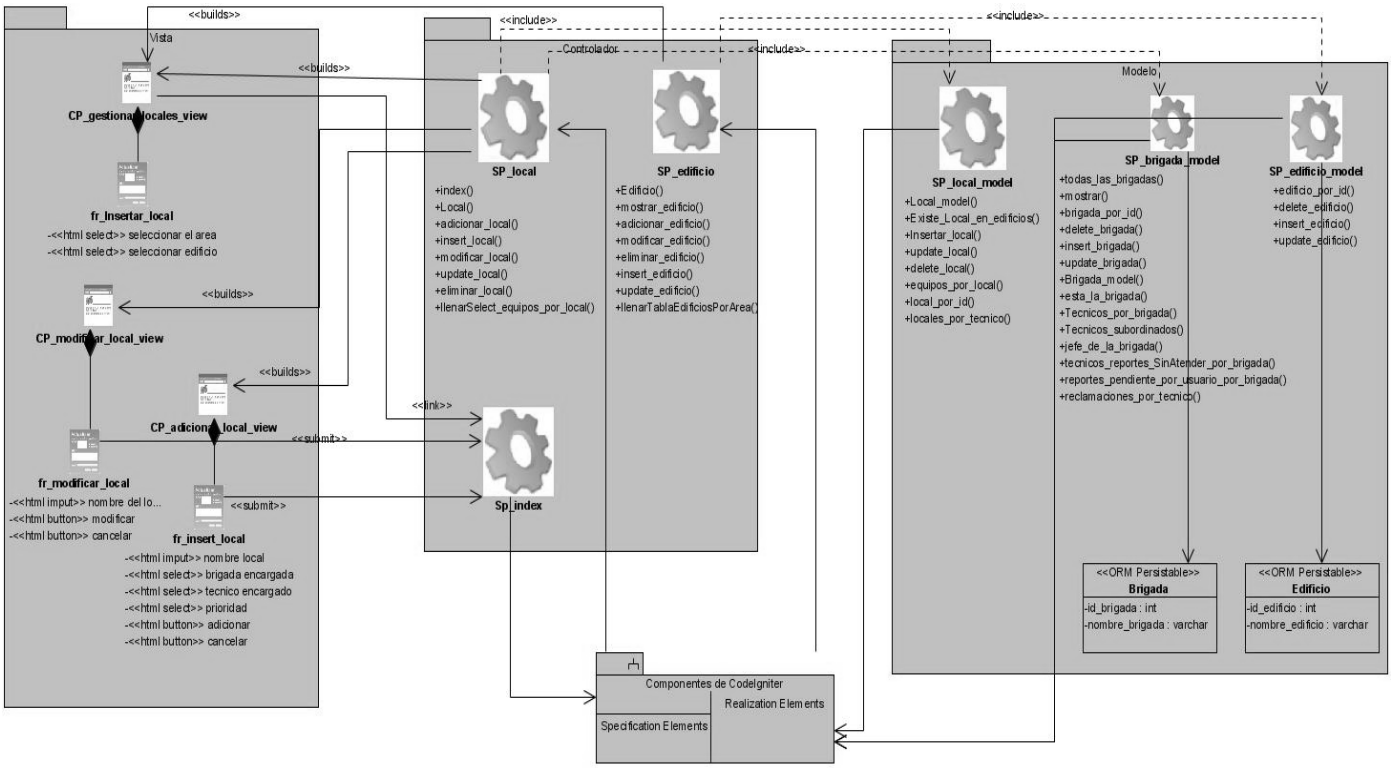


Figura 2.6.12 Diagrama de Clases Web. Gestionar Local.

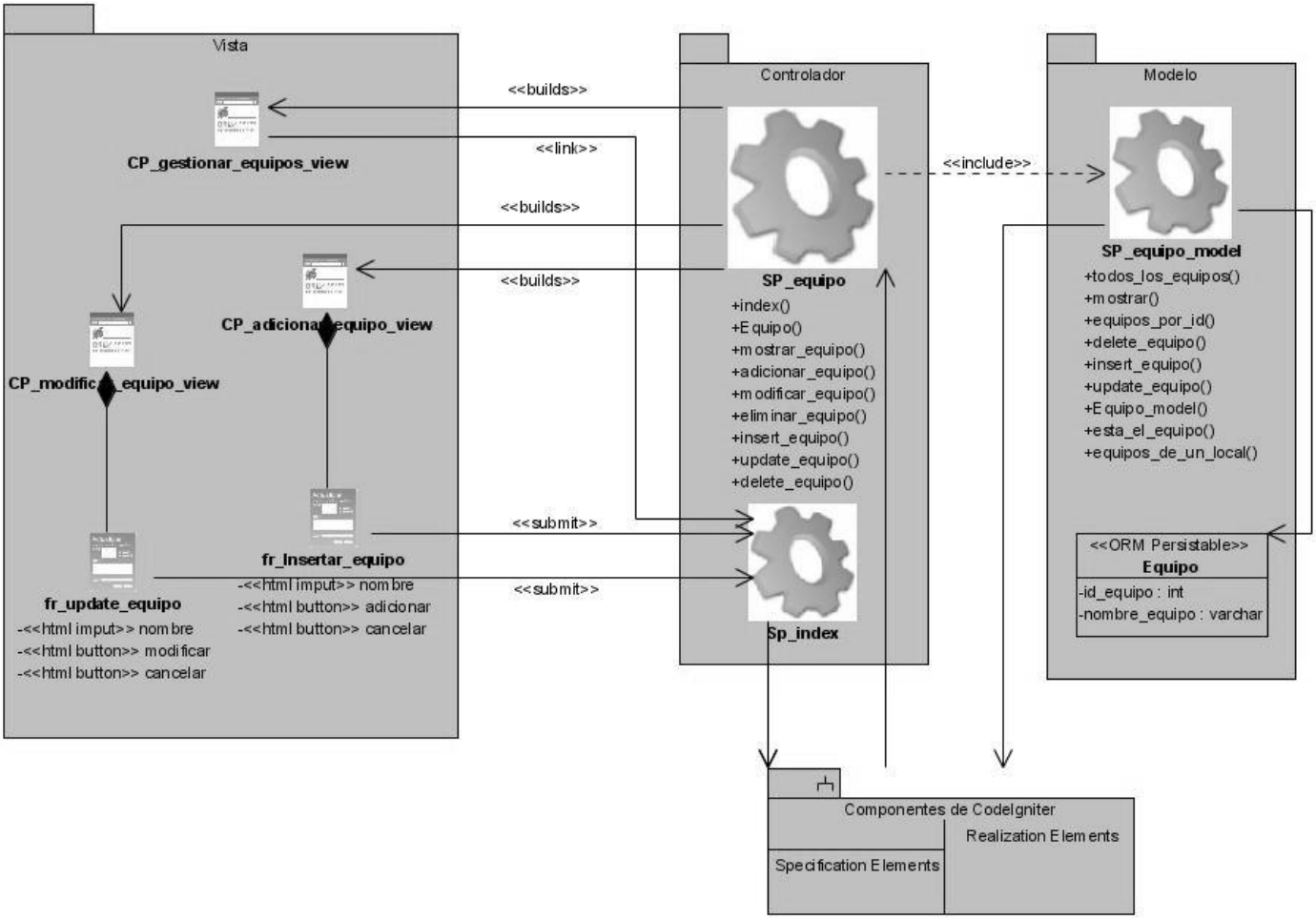


Figura 2.6.13 Diagrama de Clases Web. Gestionar Equipo.

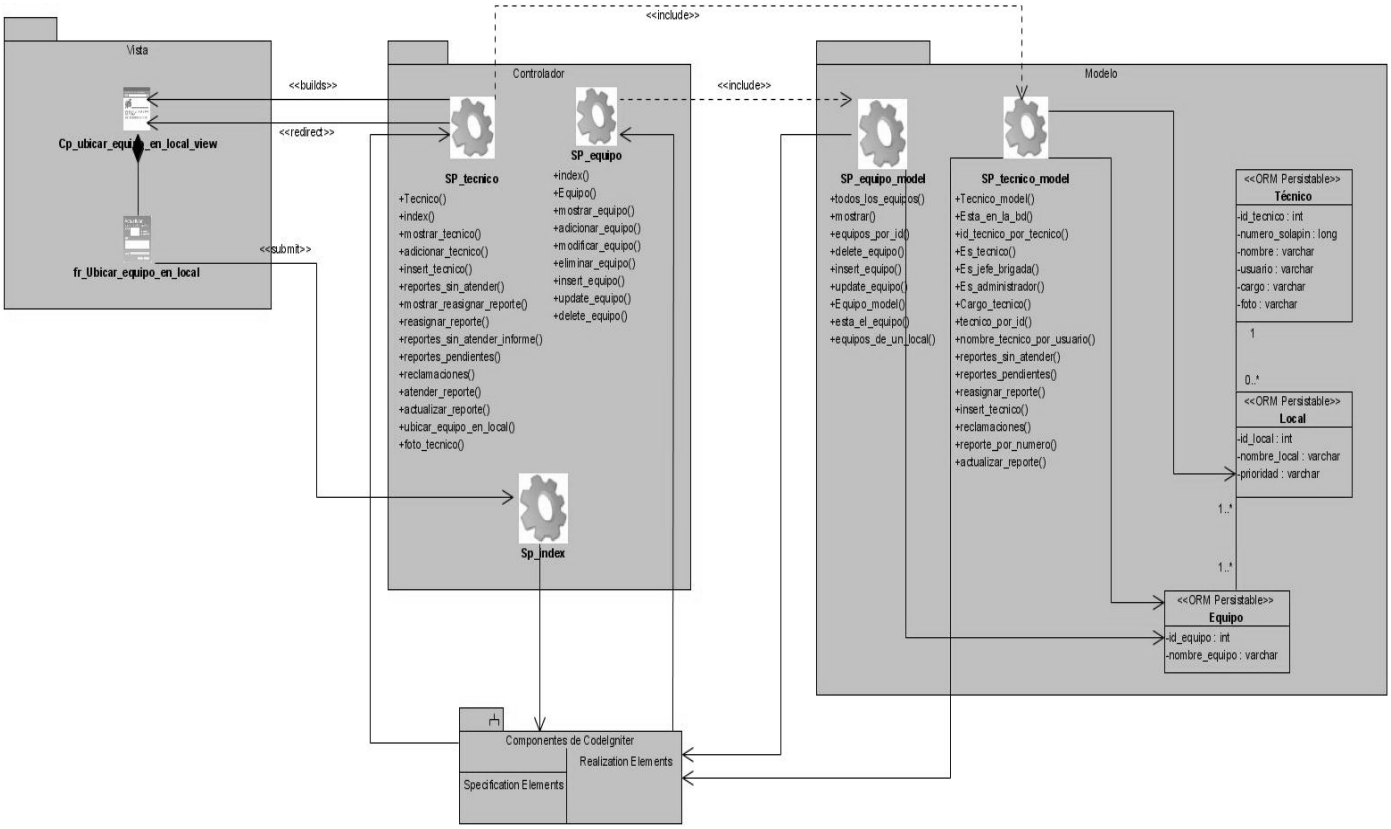


Figura 2.6.14 Diagrama de Clases Web. Ubicar Equipo en Local.

2.7 Diagrama de Clases Persistentes.

Una clase persistente es una clase entidad que tiene la capacidad de mantener su valor en el espacio y en el tiempo. En el diagrama de clases persistentes se muestran dichas clases y las relaciones entre ellas.

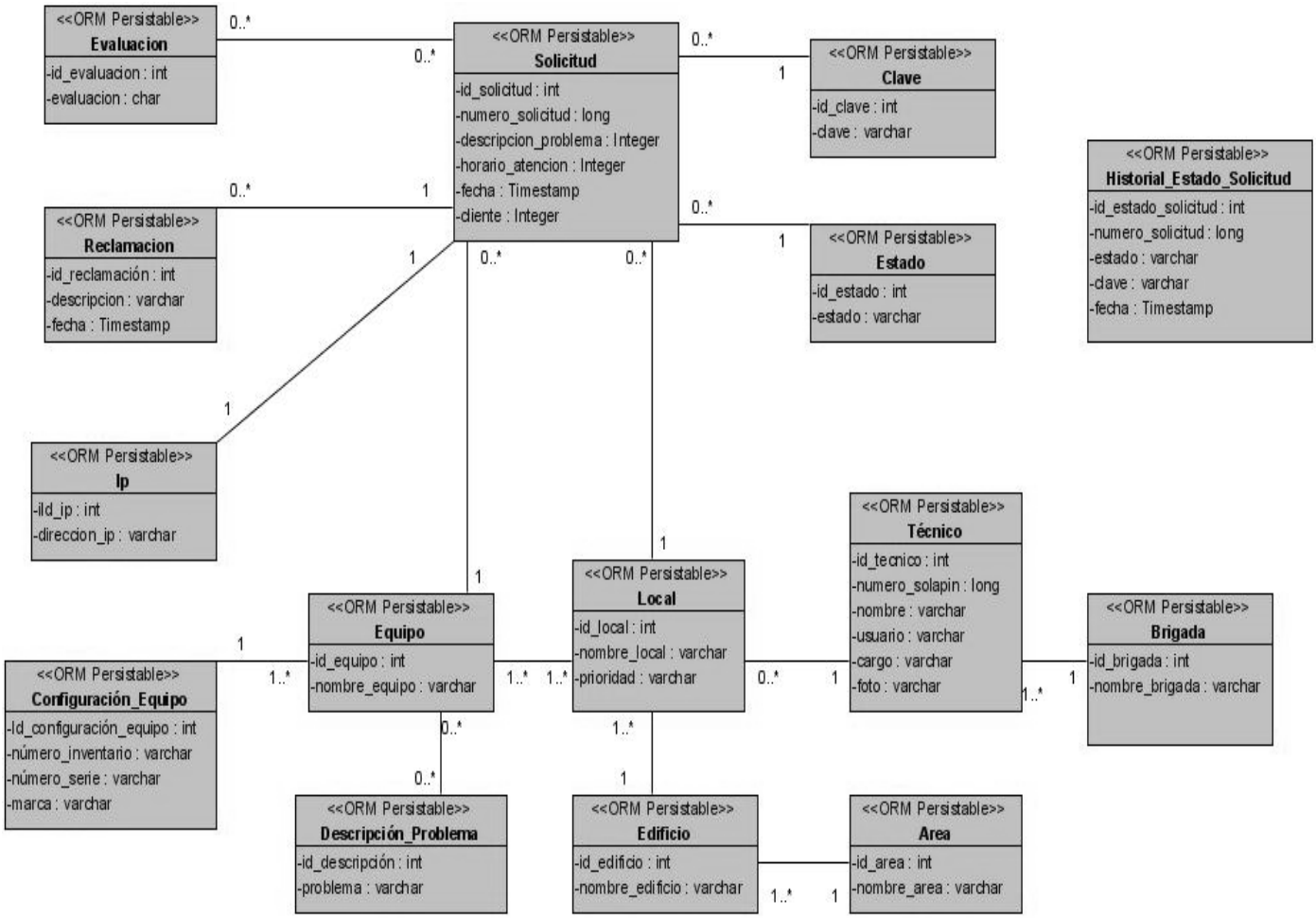


Figura 2.7.1 Diagrama de Clases Persistentes.

2.8 Modelo de Datos.

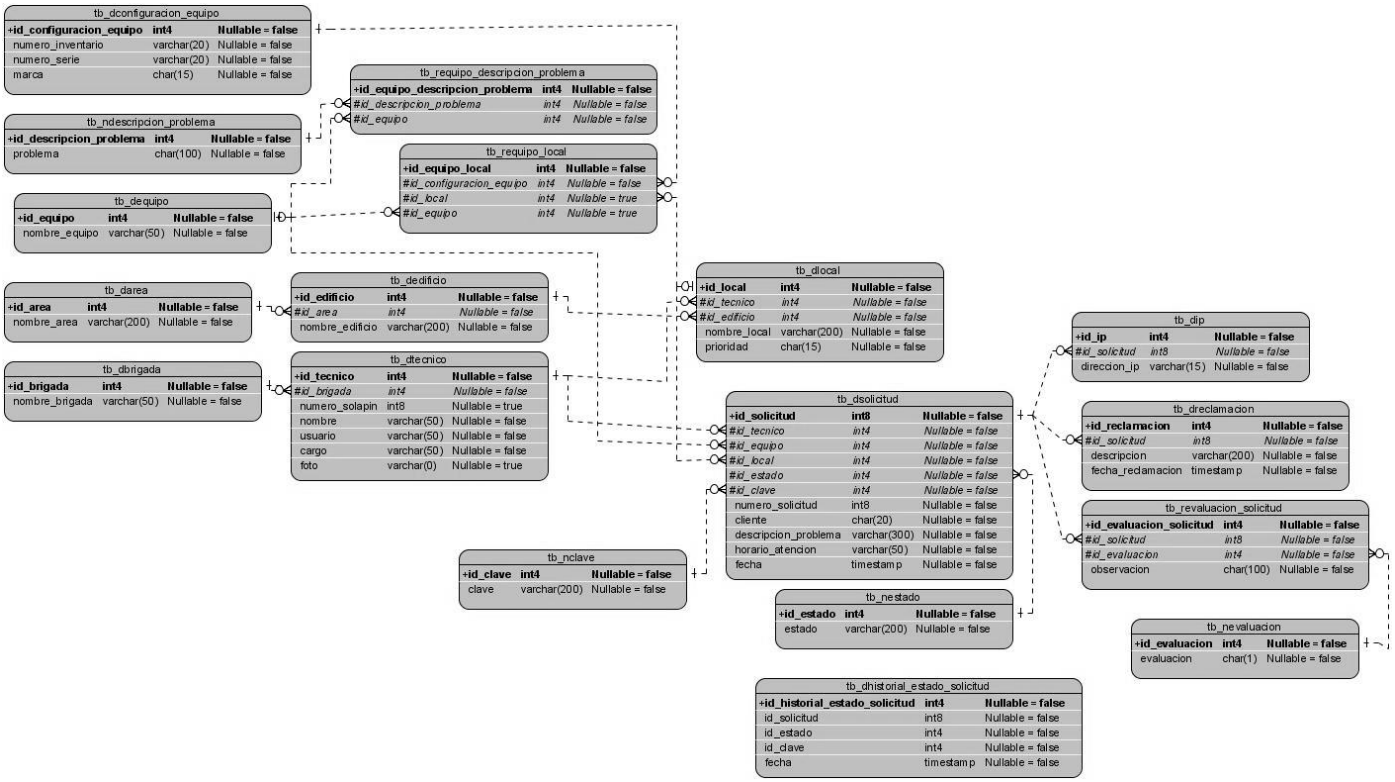


Figura 2.8.1 Modelo Físico de Datos.

2.9 Diagrama de Despliegue.

El diagrama de despliegue se utiliza para capturar los elementos de configuración del procesamiento y las conexiones entre esos elementos. También se utiliza para visualizar la distribución de los componentes de software en los nodos físicos. Este consiste en:

Nodos: elementos de procesamiento con al menos un procesador, memoria y posiblemente otros dispositivos.

Dispositivos: nodos estereotipados sin capacidad de procesamiento en el nivel de abstracción que se modela.

Conectores: expresan el tipo de conector o protocolo utilizado entre el resto de los elementos del modelo.

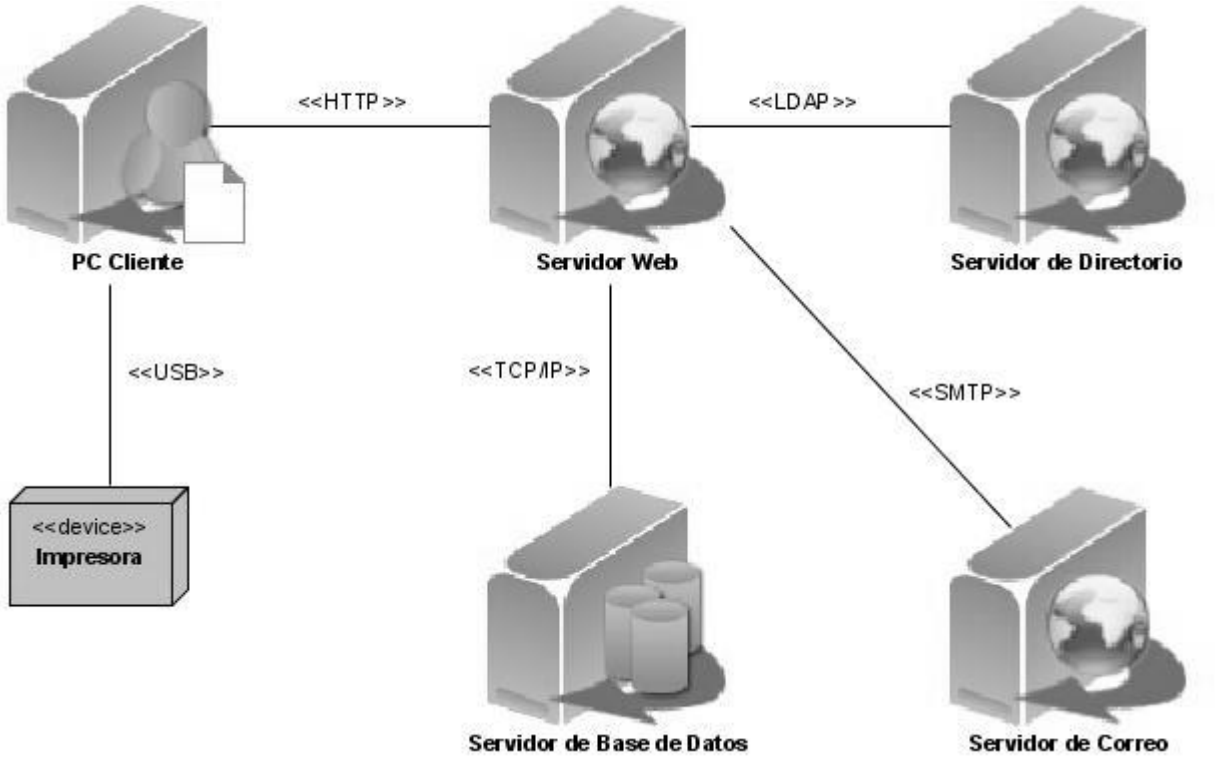


Figura 2.9.1 Diagrama de Despliegue.

2.10 Conclusiones

En relación a lo planteado en la investigación precedente se realizó un refinamiento de los requerimientos funcionales y no funcionales identificados, teniéndose en cuenta las necesidades planteadas por los clientes. En este capítulo se contempló el diseño del sistema, reflejando el diagrama de casos de uso del sistema, las descripciones textuales de los casos de uso del sistema, el diagrama de clases Web, el diagrama de clases persistentes, el modelo de datos y el diagrama de despliegue. Se plantea además la utilización del patrón arquitectónico MVC para el desarrollo de la aplicación.

Capítulo III

IMPLEMENTACIÓN DEL SISTEMA

En este capítulo se plasman los artefactos que se obtienen como resultado de la implementación del sistema. Se especifican los componentes que conforman el sistema y sus relaciones. Se incluye además el código fuente de las principales clases y pantallas de la aplicación y se brinda una descripción del mismo.

3.1 Diagrama de Componentes.

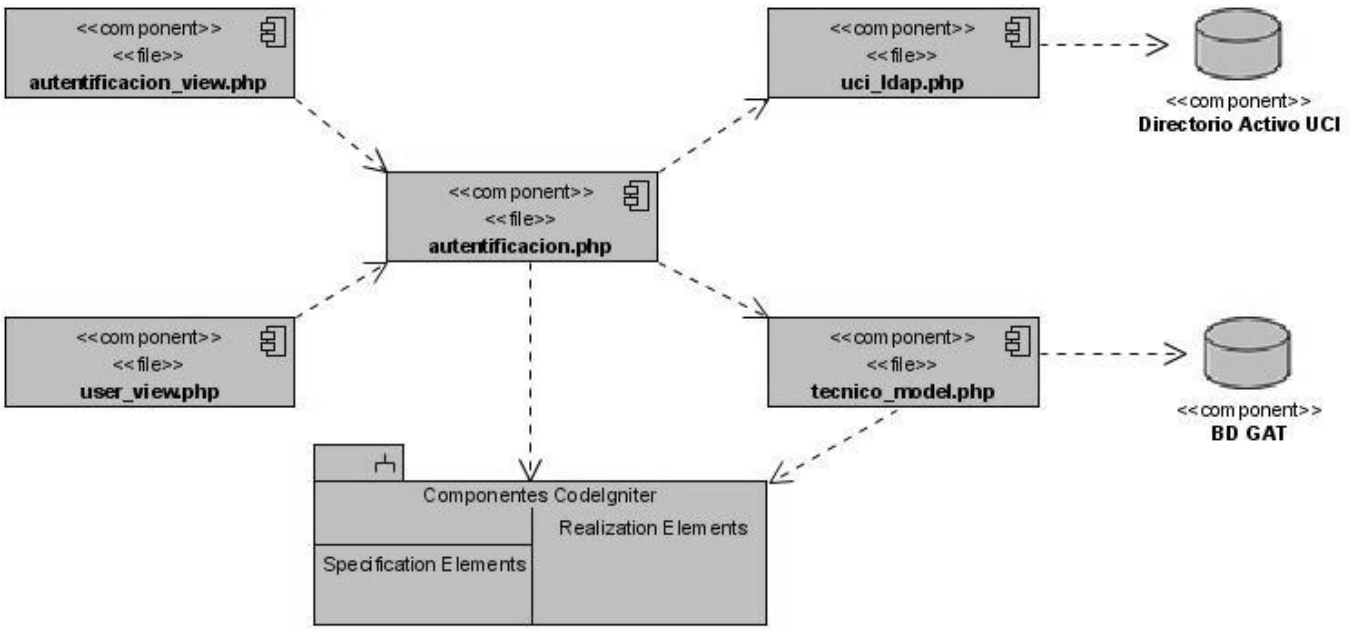


Figura 3.1.1 Diagrama de Componentes. Autentificar Usuario.

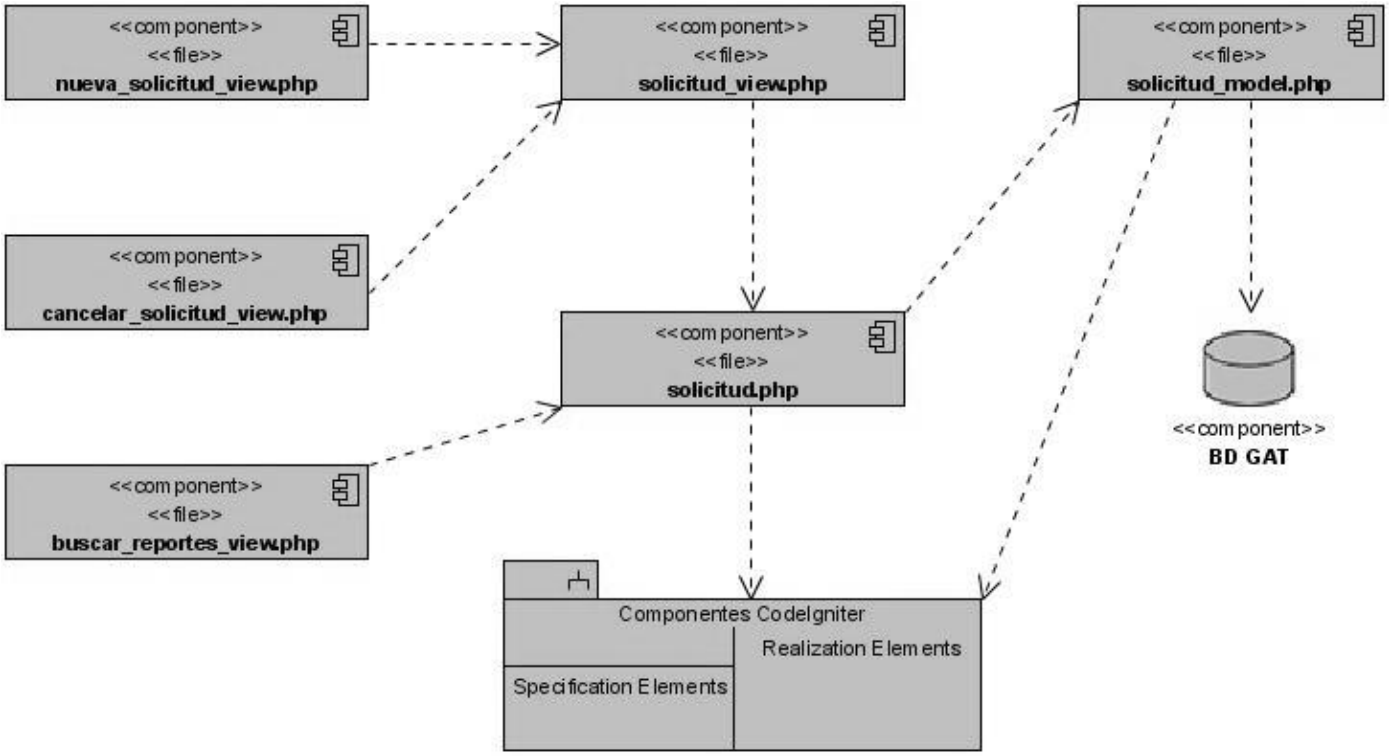


Figura 3.1.2 Diagrama de Componentes. Gestionar Reporte.

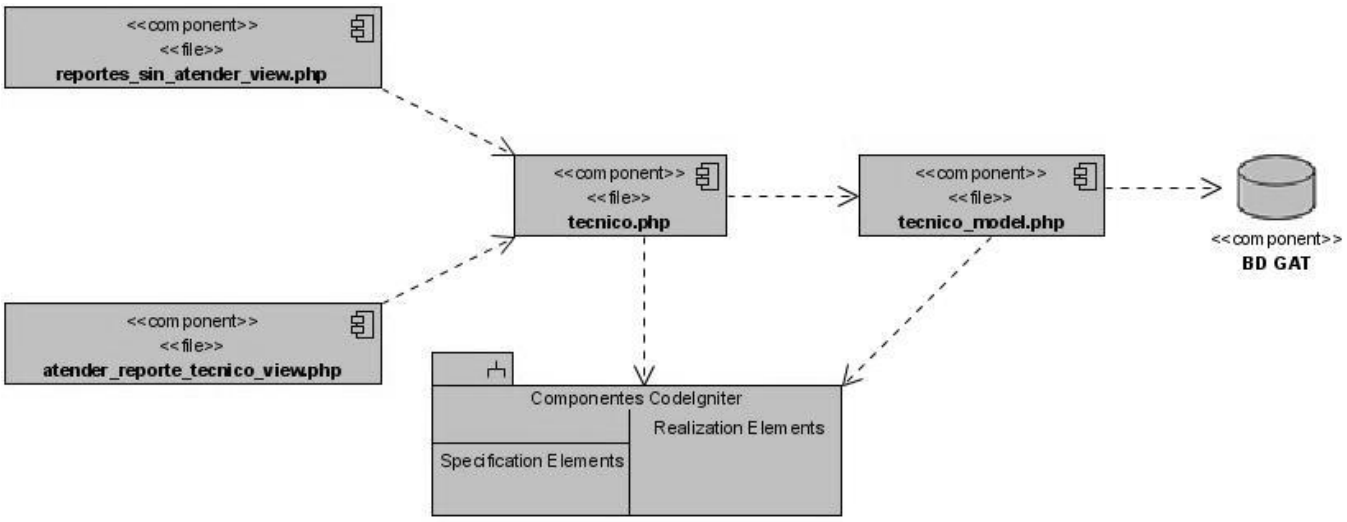


Figura 3.1.3 Diagrama de Componentes. Atender Reporte.

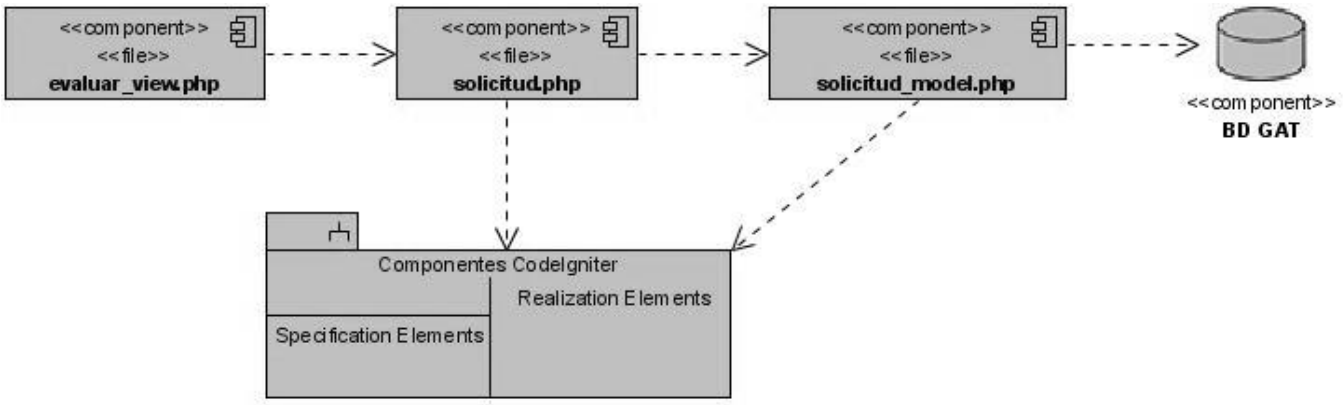


Figura 3.1.4 Diagrama de Componentes. Evaluar Servicio.

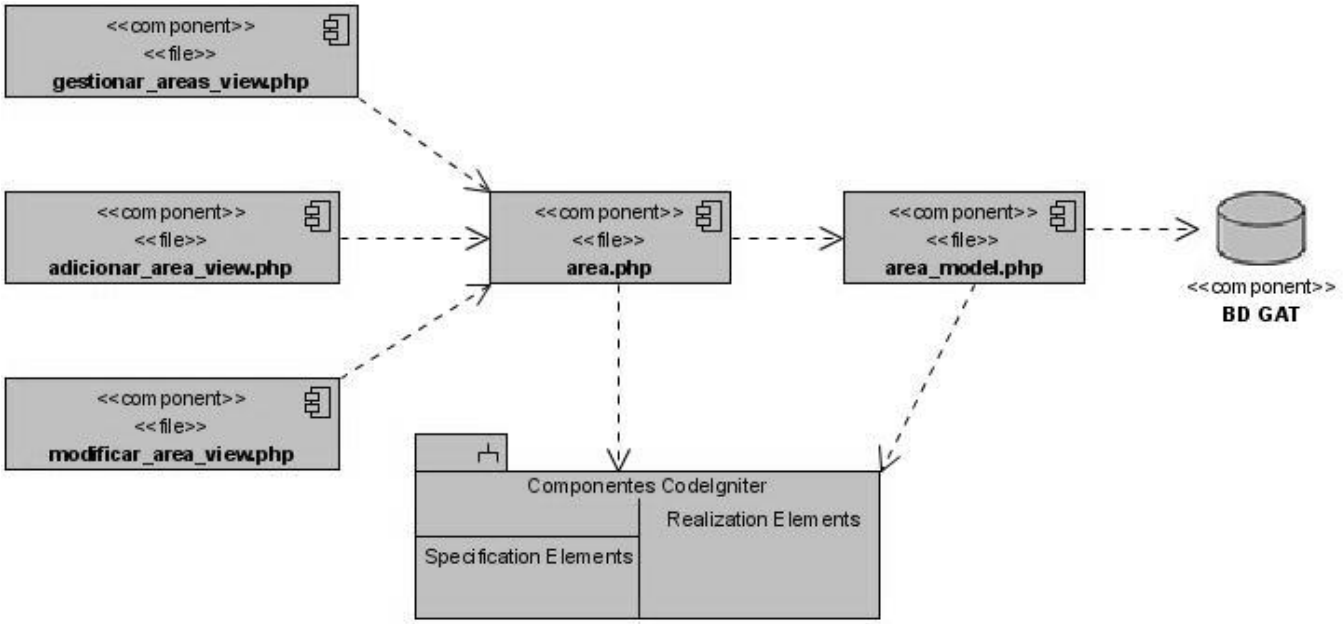


Figura 3.1.5 Diagrama de Componentes. Gestionar Área.

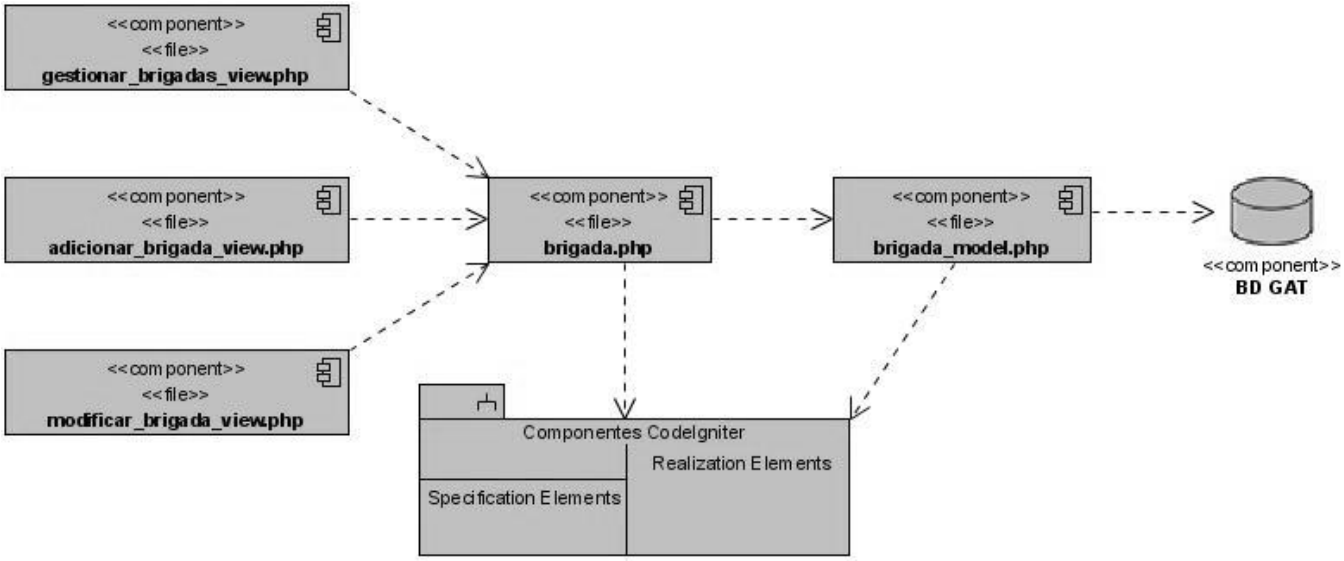


Figure 3.1.6 Diagrama de Componentes. Gestionar Brigada.

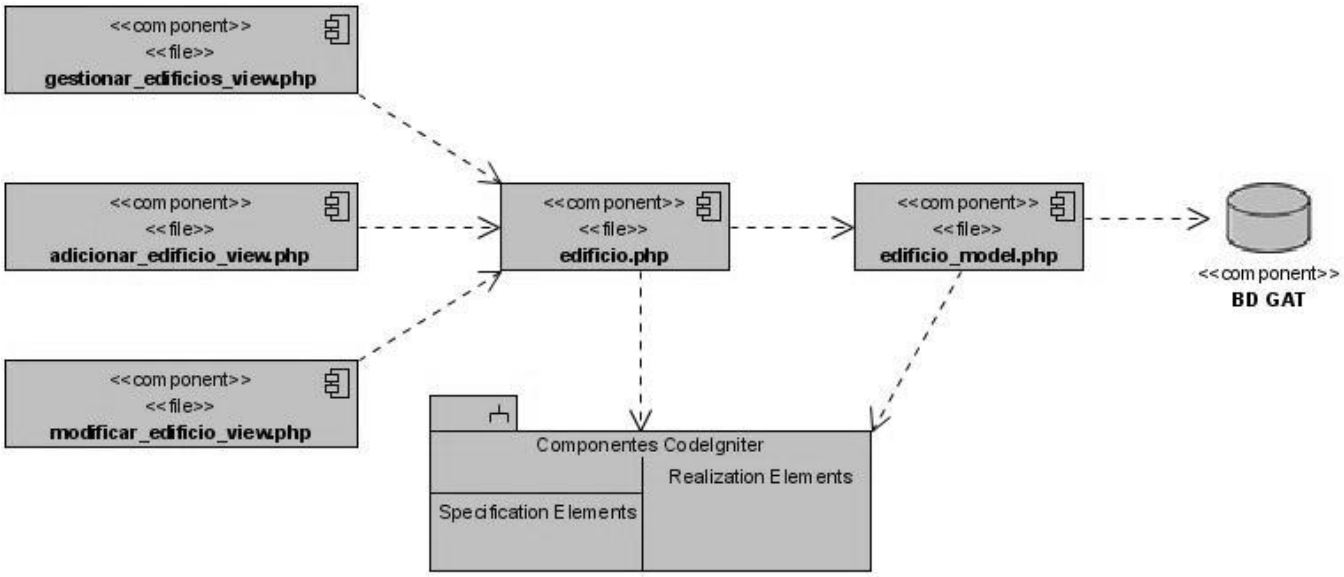


Figura 3.1.7 Diagrama de Componentes. Gestionar Edificio.

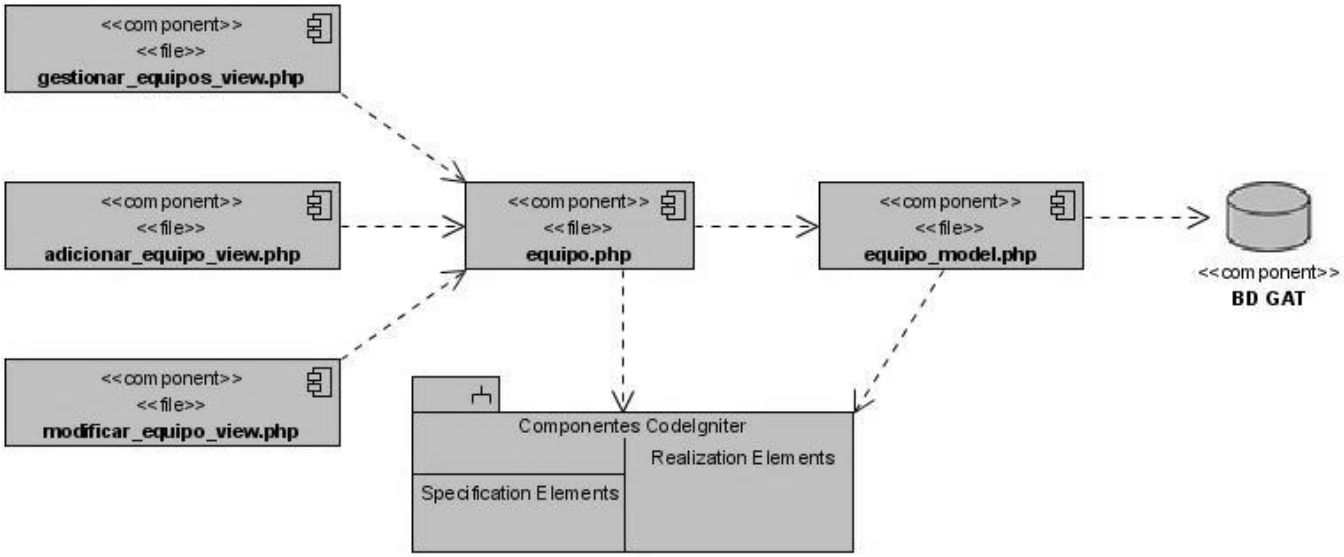


Figura 3.1.8 Diagrama de Componentes. Gestionar Equipo.

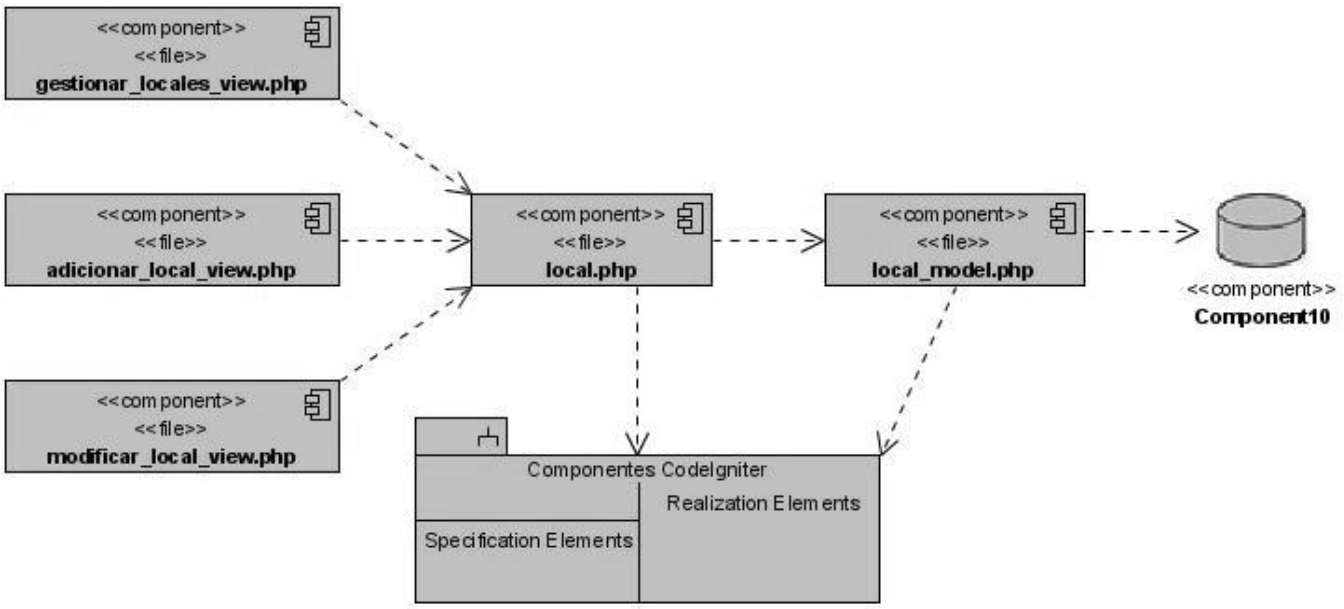


Figura 3.1.9 Diagrama de Componentes. Gestionar Local.

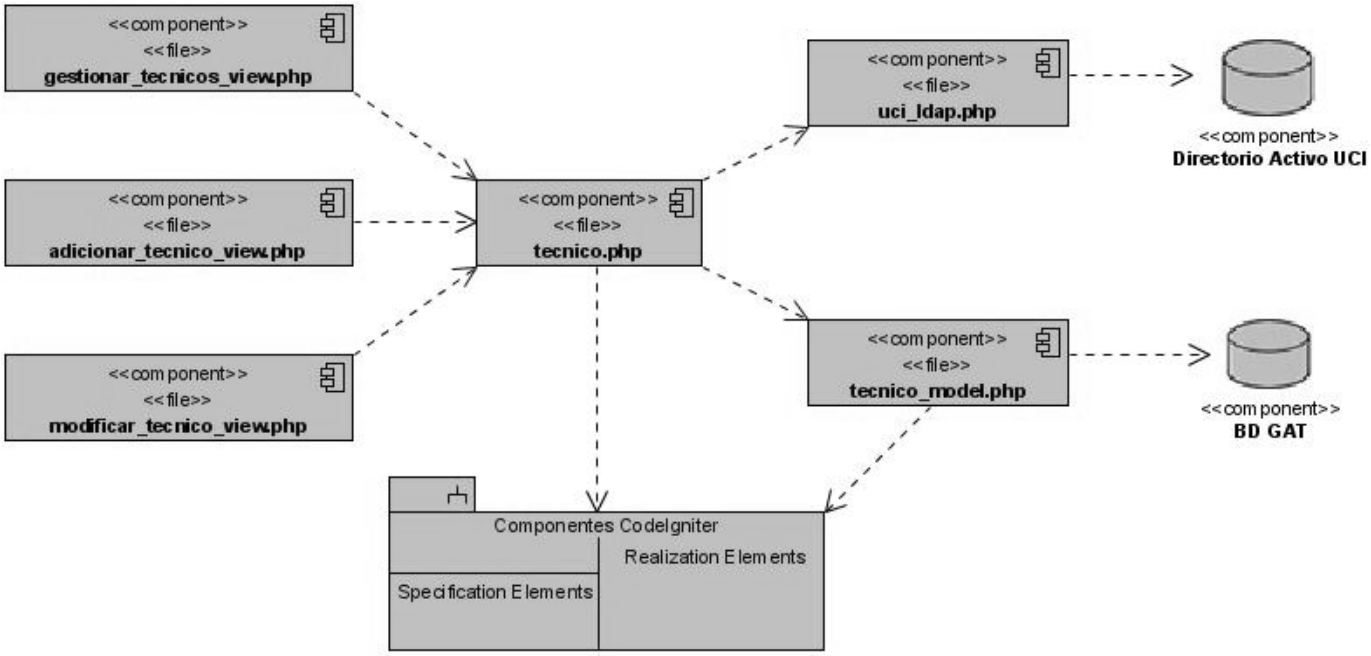


Figura 3.1.10 Diagrama de Componentes. Gestionar Técnico.

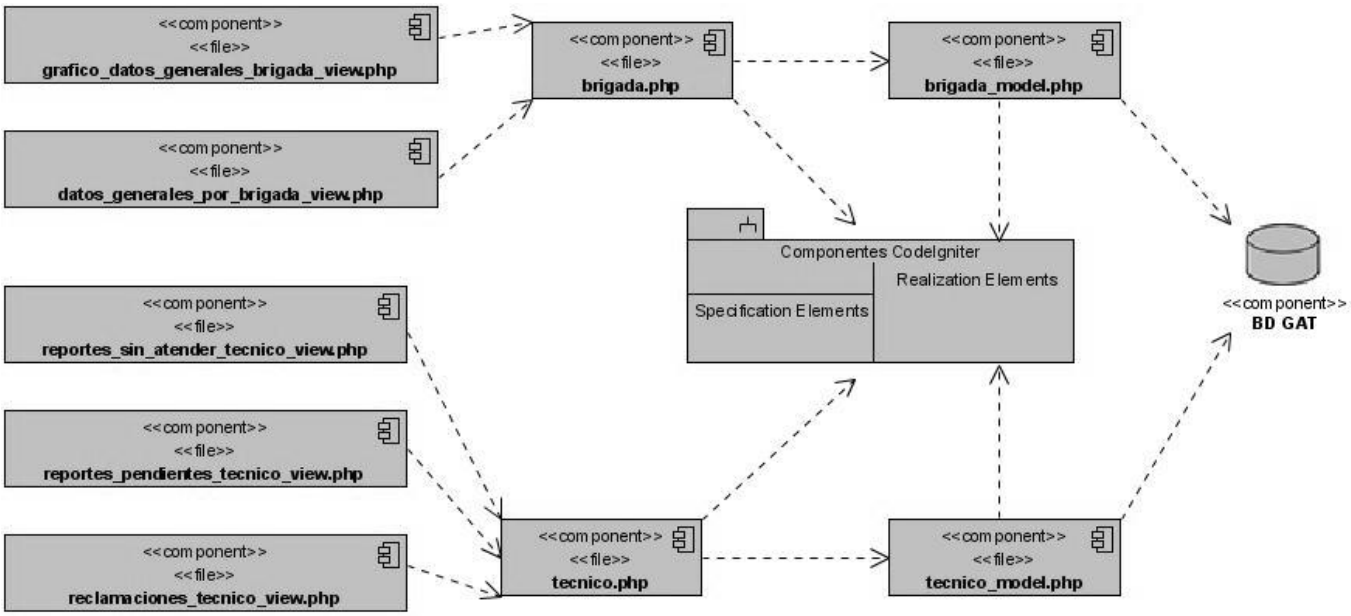


Figura 3.1.11 Diagrama de Componentes. Mostar Datos Generales de Brigada.

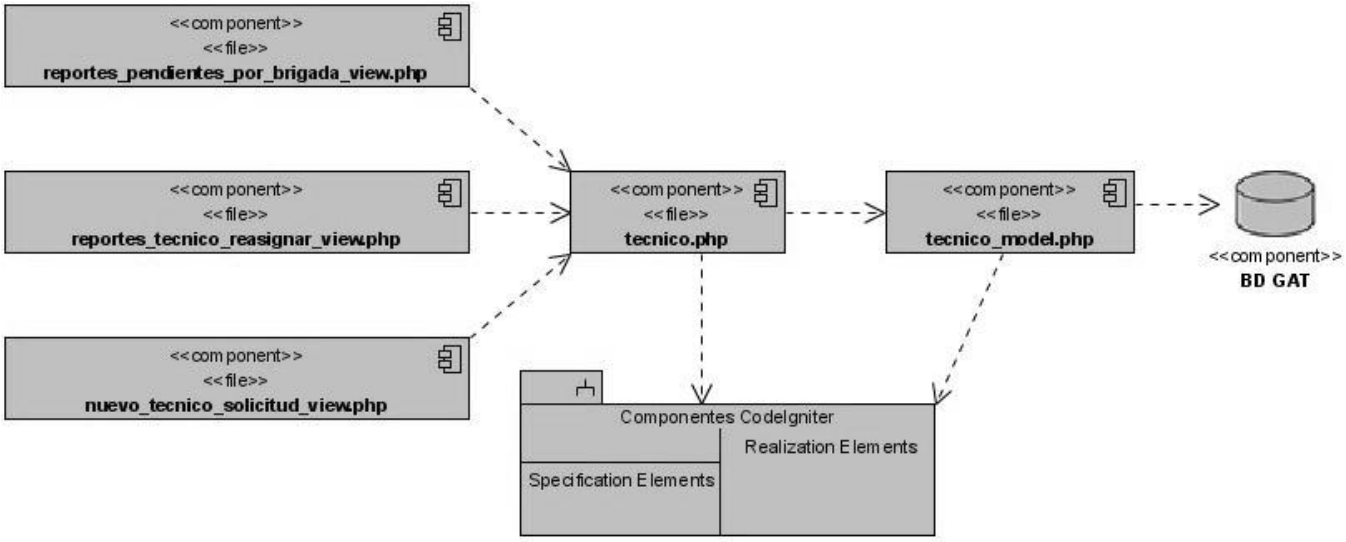


Figura 3.1.12 Diagrama de Componentes. Reasignar Reporte.

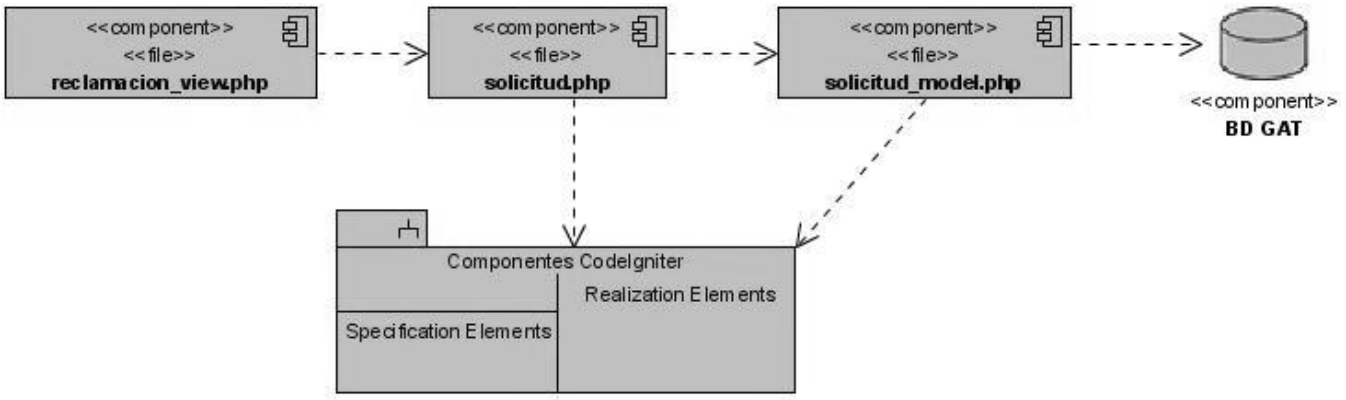


Figura 3.1.13 Diagrama de Componentes. Reclamar Reporte.

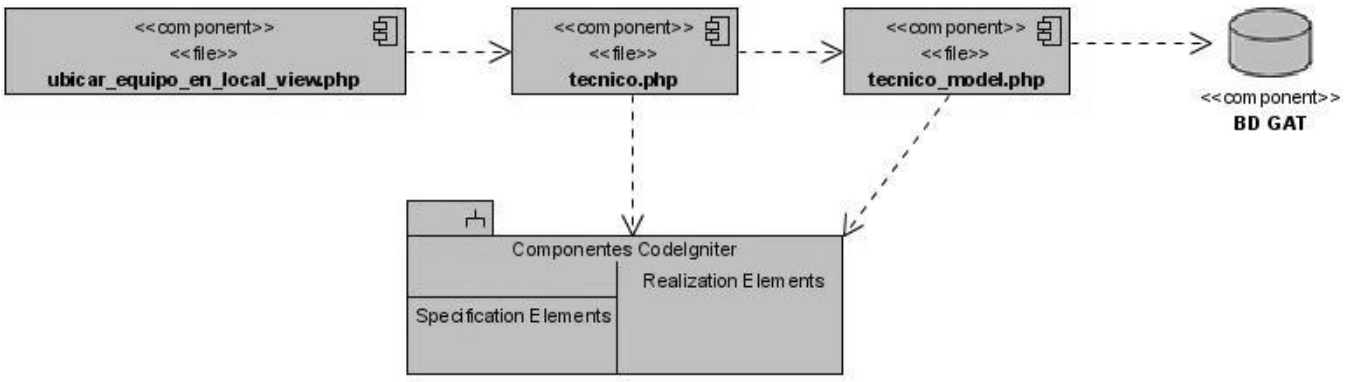


Figura 3.1.14 Diagrama de Componentes. Ubicar Equipo en Local.

3.2 Código fuente de las principales clases y pantallas de la aplicación.

Código de la primera vista de la aplicación, donde el usuario introduce sus credenciales para loguearse en esta (Vista).

```
<head>
<title>Sistema de Gesti&oacute;n de Tecnolog&iacute;a de la Informaci&oacute;n. Universidad de las Ciencias Inform&aacute;ticas.</title>
<style type="text/css">
body {margin: 0px;}
</style>
<script language="javascript" src="<?php echo base_url();?>js/Validaciones_login.js"></script>
</head>
<body>
<table style="font-family:Cambria; font-weight:bold; font-size:18px" align="center" width="987" height="596" border="0" cellspacing="0" cellpadding="0" background="<?php echo base_url(); ?>images/Fondo.jpg">
  <tr>
    <td width="987" height="394" colspan="2" valign="bottom" style="color:#FF0000">
    <?php if(isset($msg)){echo $msg;} ?>
          <br />
          <br />
          </td>
  </tr>
  <tr>
    <td width="394" height="203" align="right" valign="top">
          Usuario: <br /><br />
          Contrase&ntilde;a: 
          </td>
    <td width="594" height="203" valign="top">
          <form name="login" id="login" action="<?php echo base_url(); ?>Autentificacion/login" method="post" onSubmit="return Login();">
          <input name="user" id="user" type="text" width="180" /><br /><br />
    <input name="pass" id="pass" type="password" width="180" /><br /><br />
    <input name="entrar" type="submit" value="  Entrar  " />
```

```
        </form>
        </td>
  </tr>
</table>
</body>
```

Clase que consta de las funciones login() e index(), index() muestra la vista para que el usuario se autentifique (vista anterior), y login() donde se recogen las credenciales del usuario y se le da acceso al sistema según los permisos que este tenga (Controlador).

```
class Autentificacion extends Controller {

        function Autentificacion()
        {
                parent::Controller();
                $this->load->model('tecnico_model');
   }

        function index()
        {
         $this->load->view('autentificacion_view');
        }

        function login()
        {
         $this->load->library('uci_ldap');
         $user=$this->input->post('user');
         $pass=$this->input->post('pass');
        if(isset($user,$pass) && $user!='' && $pass!='')
   {
      $usuario=new uci_ldap();
      $server='ucidc2.uci.cu';
      $permiso =$usuario->Login($server,addslashes($user),addslashes($pass));
    if($permiso==true)
```

```
___
        {
          $ip=$_SERVER['REMOTE_ADDR'];
          $nombre=$usuario->displayname;
          $mail=$usuario->mail;
          $foto=$usuario->foto;

          $data = array(
            'ip'=>$ip,
            'nombre'=>$nombre,
            'mail'=>$mail,
            'foto'=>$foto);

          if(tecnico_model::Esta_en_la_bd($user)==false)
              {
$sesion=array('cargo'=>'usuario','nombre'=>$nombre,'email'=>$mail,'usuario'=>$user,'login'=>true);
               $this->session->set_userdata($sesion);
               $this->load->view('user_view', $data);
              }
           else
                    {
                      $cargo=tecnico_model::Cargo_tecnico($user);

$sesion=array('cargo'=>$cargo,'nombre'=>$nombre,'email'=>$mail,'usuario'=>$user,'login'=>true);
                $this->session->set_userdata($sesion);
                      $this->load->view('user_view', $data);
                     }

        }
        else
        {
         $data=array('msg'=>'Error en Usuario o Contraseña');
         $this->load->view('autentificacion_view', $data);
        }
    }
```

```
    else
     {
     redirect('Autentificacion/index');
     }
         }
}
```

Clase desde donde se accede a la base de datos para consultar, insertar, modificar y eliminar registros (Modelo), dentro de esta clase se tiene, por ejemplo, la función Esta_en_la_bd($usuario) que permite conocer si un usuario que este accediendo al sistema está registrado en la base de datos para que de esta forma se le asignen los permisos correspondientes.

```
class Tecnico_model extends Model {

  function Tecnico_model()
  {
   parent::Model();
  }

  function Esta_en_la_bd($usuario)
  {
        //$query = $this->db->get_where('tb_dtecnico', array('usuario' => $usuario));
        $sql="SELECT *FROM tb_dtecnico WHERE tb_dtecnico.usuario = '$usuario'";
        $query = $this->db->query($sql);
        if($query->num_rows>0)
        return true;
        return false;
  }

  function id_tecnico_por_tecnico($tecnico)
  {
         $this->db->where('usuario',$tecnico);
```

```
___
            $query = $this->db->get('tb_dtecnico');
            $id_tecnico=$query->result_array();
            return $id_tecnico[0]['id_tecnico'];
    }

     function Es_tecnico($usuario)
    {
            $sql="SELECT *FROM tb_dtecnico where tb_dtecnico.usuario='$usuario' and
tb_dtecnico.rol='tecnico'";
            $query = $this->db->query($sql);
                        if($query->num_rows>0)
            return true;
            return false;
    }

    function Es_jefe_brigada($usuario)
    {
            $sql="SELECT *FROM tb_dtecnico where tb_dtecnico.usuario='$usuario' and
tb_dtecnico.rol='jefe de brigada'";
            $query = $this->db->query($sql);
          if($query->num_rows>0)
            return true;
            return false;
    }

    function Es_administrador($usuario)
    {
            $sql="SELECT *FROM tb_dtecnico where tb_dtecnico.usuario='$usuario' and
tb_dtecnico.rol='administrador'";
            $query = $this->db->query($sql);
          if($query->num_rows>0)
            return true;
            return false;
    }
```

```
function Cargo_tecnico($usuario)
{
 $this->db->select('cargo');
        $this->db->where('usuario',$usuario);
        $query=$this->db->get('tb_dtecnico');
        $cargo=$query->result_array();
        return $cargo[0]['cargo'];
}

function tecnico_por_id($id)
        {
         $this->db->where('id_tecnico',$id);
         $query = $this->db->get('tb_dtecnico');
         return $query->result_array();
        }

function nombre_tecnico_por_usuario($usuario_tecnico)
        {
         $this->db->select('nombre') ;
         $this->db->where('usuario',$usuario_tecnico);
         $query = $this->db->get('tb_dtecnico');
         return $query->result_array();

        }

function reportes_sin_atender($usuario_tecnico)
        {
         $id_tecnico= Tecnico_model::id_tecnico_por_tecnico($usuario_tecnico);
                $query=$this->db->query("SELECT
public.tb_dsolicitud.numero_solicitud,
public.tb_dlocal.nombre_local,
public.tb_dequipo.nombre_equipo,
public.tb_dsolicitud.descripcion_problema,
```

```
  public.tb_dsolicitud.cliente,
  public.tb_dsolicitud.fecha
FROM
 public.tb_dsolicitud
 INNER JOIN public.tb_dlocal ON (public.tb_dsolicitud.id_local=public.tb_dlocal.id_local)
 INNER JOIN public.tb_dtecnico ON (public.tb_dlocal.id_tecnico=public.tb_dtecnico.id_tecnico)
 INNER JOIN public.tb_dequipo ON (public.tb_dsolicitud.id_equipo=public.tb_dequipo.id_equipo)
WHERE
  (public.tb_dsolicitud.id_tecnico=$id_tecnico) and (public.tb_dsolicitud.id_estado=2 or
public.tb_dsolicitud.id_estado=1)
  ");
            return $query->result();
           }

   function reportes_pendientes($tecnico)
   {
            $id_tecnico= Tecnico_model::id_tecnico_por_tecnico($tecnico);
                    $query=$this->db->query("SELECT
  public.tb_dsolicitud.numero_solicitud,
  public.tb_dlocal.nombre_local,
  public.tb_dequipo.nombre_equipo,
  public.tb_dsolicitud.descripcion_problema,
  public.tb_dsolicitud.cliente,
  public.tb_dsolicitud.fecha
FROM
 public.tb_dsolicitud
 INNER JOIN public.tb_dlocal ON (public.tb_dsolicitud.id_local=public.tb_dlocal.id_local)
 INNER JOIN public.tb_dtecnico ON (public.tb_dlocal.id_tecnico=public.tb_dtecnico.id_tecnico)
 INNER JOIN public.tb_dequipo ON (public.tb_dsolicitud.id_equipo=public.tb_dequipo.id_equipo)
WHERE
  (public.tb_dsolicitud.id_tecnico=$id_tecnico and public.tb_dsolicitud.id_estado=3 )
  ");
   if ($query->num_rows()>0)
     {
```

```
    return $query->result();
  }
  else
  {
        return false;
  }
}

function reasignar_reporte($numero_reporte , $nuevo_id_tecnico)
{
    $data = array( 'id_tecnico' => $nuevo_id_tecnico);
    $this->db->where('numero_solicitud', $numero_reporte);
    $this->db->update('tb_dsolicitud', $data);

}

function insert_tecnico($id_brigada,$numero_solapin,$nombre,$usuario,$cargo,$rol,$foto)
        {
        $data =array(
              'id_brigada'=>$id_brigada,
                        'numero_solapin'=>$numero_solapin,
                        'nombre'=>$nombre,
                        'usuario'=>$usuario,
                        'cargo'=>$cargo,
                        'rol'=>$rol,
                        'foto'=>$foto,
        );
        $this->db->insert('tb_dtecnico', $data);

        }

function reclamaciones($tecnico)
{
        $query =$this->db->query("SELECT
```

```
  public.tb_dreclamacion.descripcion,
  public.tb_dreclamacion.fecha_reclamacion,
  public.tb_dsolicitud.cliente,
  public.tb_dlocal.nombre_local
FROM
 public.tb_dsolicitud
 INNER JOIN public.tb_dreclamacion ON
(public.tb_dsolicitud.id_solicitud=public.tb_dreclamacion.id_solicitud)
 INNER JOIN public.tb_dtecnico ON (public.tb_dsolicitud.id_tecnico=public.tb_dtecnico.id_tecnico)
 INNER JOIN public.tb_dlocal ON (public.tb_dsolicitud.id_local=public.tb_dlocal.id_local)
WHERE
  (public.tb_dtecnico.usuario = '$tecnico')");

  if($query->num_rows()>0)
   {
    return $query->result_array();
   }
  else
   {
          return false;
   }
   }

          function reporte_por_numero($numero_solicitud)
          {
          $query=$this->db->query("SELECT
  public.tb_dlocal.nombre_local,
  public.tb_dequipo.nombre_equipo
  FROM
 public.tb_dsolicitud
 INNER JOIN public.tb_dlocal ON (public.tb_dsolicitud.id_local=public.tb_dlocal.id_local)
  INNER JOIN public.tb_dequipo ON (public.tb_dsolicitud.id_equipo=public.tb_dequipo.id_equipo)
WHERE
  (public.tb_dsolicitud.numero_solicitud=$numero_solicitud)
```

```
  ");
   if ($query->num_rows()>0)
     {
      return $query->result();
     }
     else
     {
           return false;
     }

           }

           function actualizar_reporte($numero_solicitud,$descripcion_tecnica,$id_estado,$id_clave)
           {

                   $data = array(
              'descripcion_problema' => $descripcion_tecnica,
              'id_estado' => $id_estado,
              'id_clave' => $id_clave
           );

    $this->db->where('numero_solicitud', $numero_solicitud);
    $this->db->update('tb_dsolicitud', $data);



           }
}
```

Vista desde donde los usuarios podrán hacer la solicitud de servicio. Este segmento de código permite saber el tipo de usuario que está logueado en la aplicación y en dependencia de esto se le muestra la vista pertinente para realizar la solicitud, en este caso pueden ser dos vista, la que se le muestra a usuarios comunes ($this->load->view('solicitud2')) y la que se le muestra al técnico encargado de recoger las solicitudes por vía telefónica ($this->load->view('solicitud1')).

```
<?php
        if(!($this->session->userdata('cargo')=='recepcionista'))
        $this->load->view('solicitud1');
        else
        $this->load->view('solicitud2');
        ?>
```

Esta es una de las principales clases de la aplicación, en esta se tienen programadas todas las acciones necesarias para crear o cancelar una solicitud de servicio, además de otras funciones que cumplen objetivos específicos dentro del sistema.

```
class Solicitud extends Controller {
        function Solicitud()
        {
                parent::Controller();
                $this->load->scaffolding('tb_dsolicitud');
                $this->load->model('solicitud_model');
           $this->load->model('area_model');
           $this->load->model('equipo_model');
           $this->load->model('local_model');
           $this->load->model('tecnico_model');
        }

        function index()
        {
        if(!($this->session->userdata('login'))==true)
        {
         redirect('Autentificacion/index');
        }
        else
        {
         $this->load->view('solicitud_view');
        }
        }
```

```
function nueva_solicitud()
{
if(!($this->session->userdata('login'))==true)
{
 redirect('Autentificacion/index');
}
else
{
        $areas=Area_model::todas_las_areas();
        $data['areas']=$areas;
        $this->load->view('nueva_solicitud_view',$data);
}
}

function cancelar_solicitud()
{
if(!($this->session->userdata('login'))==true)
{
 redirect('Autentificacion/index');
}
else
{
 $this->load->view('cancelar_solicitud_view');
}
}

function reclamacion()
{
if(!($this->session->userdata('login'))==true)
{
 redirect('Autentificacion/index');
}
else
```

```
{
 $this->load->view('reclamacion_view');
}
}
function evaluar()
{
if(!($this->session->userdata('login'))==true)
{
 redirect('Autentificacion/index');
}
else
{
        $evaluaciones=Solicitud_model::todas_las_evaluaciones();
        $data['evaluaciones']=$evaluaciones;
        $this->load->view('evaluar_view',$data);
}
}

function buscar_reportes()
{
if(!($this->session->userdata('login'))==true)
{
 redirect('Autentificacion/index');
}
else
{
 $this->load->view('buscar_reportes_view');
}
}

function buscar()
{
if(!($this->session->userdata('login'))==true)
{
```

```
 redirect('Autentificacion/index');
}
else
{
        $numero_solicitud=$this->input->post('numero_reporte');
        $resultado=Solicitud_model::buscar_solicitud($numero_solicitud);
        $data['resultado']=$resultado;
        $this->load->view('buscar_reportes_view',$data);
}
}

function administracion()
{
if(!($this->session->userdata('login'))==true)
{
 redirect('Autentificacion/index');
}
else
{
 $this->load->view('administracion_view');
}
}
```

La función insert_solicitud() es una de las funciones más importantes en la aplicación, por lo que se explicará detalladamente.

```
function insert_solicitud()
{
//se crea el número de la solicitud (número que estará conformado por la fecha actual + un número incremental que se obtendrá de la base de datos)
  $this->load->helper('date'); //cargando un helper para el trabajo con las fechas del framework CodeIgniter
    $datestring = "%Y%m%d";//dandole el formato a la fecha
  $time = time();
```

```
 $numero_solicitud = mdate($datestring, $time); //en las cuatro siguientes líneas de código se conforma el número de la solicitud
    $temp = Solicitud_model::ultima_solicitud();
    $ultimo_id = $temp[0]['id_solicitud']+1;
    $numero_solicitud.= $ultimo_id;//este

    //local de la solicitud
    $id_local = $this->input->post('local'); //obteniendo los datos del local que el usuario introdujo en la vista

    //equipo de la solicitud
    $id_equipo = $this->input->post('equipo'); //obteniendo los datos del equipo que el usuario introdujo en la vista

    //id del técnico encargado
    $local = $this->input->post('local');
    $tecnico = Solicitud_model::tecnicos_por_locales($local); //según el local que el usuario introdujo en la vista, se determina según los registros de la base de datos cuál es el técnico encargado de atender la solicitud
        $tecnico_encargado = $tecnico[0]['id_tecnico'];//este

        //cliente
        $usuario = $this->input->post('usuario'); //obteniendo los datos del usuario que hace la solicitud

        //horario atención
        $horario_atencion = $this->input->post('horario'); //obteniendo el horario de atención que el usuario escogió para atender su solicitud

        //descripción del problema
        $problema = $this->input->post('problema'); //obtenido la descripción del problema asociado a la solicitud que realiza el usuario

        //fecha de la solicitud
```

$datestring = "%m/%d/%Y %h:%i %a"; //en las tres líneas de código siguiente se determina la fecha actual en la que se esta realizando la solicitud

```
$time = time();
$fecha = mdate($datestring, $time);//este
//echo $fecha;

//se verifica si la solicitud existe en la base de datos
$resultado=Solicitud_model::existe_solicitud_bd($id_local, $id_equipo ,$problema);
if ($resultado==false) //si no está se inserta
{

//insertando la solicitud
Solicitud_model::insert_solicitud($tecnico_encargado, $id_equipo, $id_local, 1, 1,
$numero_solicitud, $usuario, $problema, $horario_atencion, $fecha);

//guardando IP desde donde se realiza la solicitud
        $ip =$this->input->ip_address();
        $id_solicitud=Solicitud_model::ultima_solicitud();
        $id_solicitud=$id_solicitud[0]['id_solicitud'];
        Solicitud_model::insert_solicitud_ip($id_solicitud, $ip);

//insertando en la tabla historial de la solicitud
$id_estado=1;//estado por defecto
$id_clave=1;//clave por defecto
Solicitud_model::insert_historial_estado($id_solicitud, $id_estado, $id_clave, $fecha);
```

//este segmento de código es para enviarle al usuario un correo con los datos de la solicitud que acaba de realizar, este incluye el nombre del técnico encargado de atenderlo y el número de la solicitud

```
        /*$this->load->library('email');
        $this->email->from('jloliva@estudiantes', 'Grupo de Desarrollo');
        $correo_usuario=$this->session->userdata('email');;
        $this->email->to($correo_usuario);
        $this->email->subject('Solicitud de Servicios');
```

```
        $mensaje= "correo de prueba, por favor no responder.";
        $this->email->message($mensaje);
        $this->email->send();
        echo $this->email->print_debugger();*/
    $nombre_usuario = $this->session->userdata('nombre');
    $msg ='<strong>'. $nombre_usuario .'</strong>'. 'ha realizado una solicitud de servicio satisfactoriamente, a continuación recibirás un correo con los datos de tu reporte ,es importante que conserve el número de la solicitud para posibles reclamaciones o por si desea emitir un criterio del servicio recibido.
    Atentamente: Grupo Asistencia Técnica';
    $data=array ('msg'=>$msg);
    $this->load->view('informacion_view',$data);
    }
    else  // si la solicitud existe, se obtienen sus datos y se le muestran al usuario
     {
     $nombre_equipo= $resultado[0]->nombre_equipo;
     $nombre_local =$resultado[0]->nombre_local;
     $direccion_ip = $resultado[0]->direccion_ip;
     $fecha=$resultado[0]->fecha;
     $cliente=$resultado[0]->cliente;
     $estado = $resultado[0]->estado;
     //$clave = $resultado[0]->clave;
     $msg = ' El equipo  '.$nombre_equipo .'  del local  '.$nombre_local.' ya fue reportado por usuario '.$cliente .
     ' desde la pc '.$direccion_ip . ' el  '.$fecha . '  y dicha solicitud se encuentra en estado '.$estado;
     $data = array('msg'=>$msg);
     $this->load->view('informacion_view',$data);
     }
   }

        function eliminar_solicitud() //función que elimina un reporte de la base de datos
        {
        $numero_solicitud=$this->input->post('numero_solicitud');
        $id_solicitud = Solicitud_model::id_solicitud($numero_solicitud);
```

```
        if($id_solicitud!=false)
        {
          Solicitud_model::cancelar_reporte($id_solicitud);
          $msg='Solicitud'.$numero_solicitud.'cancelada correctamente';
          $data['msg']=$msg;
      $this->load->view('informacion_view',$data);
        }
        else
        {
          $msg='Dicha solicitud no existe';
          $data['msg']=$msg;
      $this->load->view('informacion_view',$data);
        }
        }

        function buscar_solicitud() //función que permite a los usuarios, teniendo el número de la solicitud buscar saber el estado en que se encuentra su reporte
        {
    $numero=$this->input->post('numero_reporte');
    $resultado=Solicitud_model::buscar_solicitud($numero);
    if($resultado==false)
    {
      $data['msg']='0 Registros Encontrados';
      $data['resultado']=false;
        $this->load->view('resultados_reportes_view',$data);
    }
    else
    {
      //tecnico
        $id_tecnico= $resultado[0]['id_tecnico'];
      $tecnico = Tecnico_model::tecnico_por_id($id_tecnico);
      $tecnico= $tecnico[0]['nombre'];

      //local
```

```
        $id_local= $resultado[0]['id_local'];
    $local = Local_model::local_por_id($id_local);
    $local= $local[0]['nombre_local'];

    //equipo
    $id_equipo= $resultado[0]['id_equipo'];
    $equipo = Equipo_model::equipos_por_id($id_equipo);
    $equipo= $equipo[0]['nombre_equipo'];

    //problema
    $problema= $resultado[0]['descripcion_problema'];

    //estado
    $id_estado= $resultado[0]['id_estado'];
    $estado =Solicitud_model::estado($id_estado);
    $estado= $estado[0]['estado'];

    //clave
    $id_clave= $resultado[0]['id_clave'];
    $clave =Solicitud_model::clave_por_id($id_clave);
    $clave= $clave[0]['clave'];

    //llamando a la vista y pasándole los datos de la consulta
    $data['tecnico']=$tecnico;
    $data['local']=$local;
    $data['equipo']=$equipo;
    $data['problema']=$problema;
    $data['estado']=$estado;
    $data['clave']=$clave;
    $data['resultado']=true;
    $this->load->view('resultados_reportes_view',$data);
  }
      }
```

```
function insert_evaluacion() //función que permite al usuario insertar una evaluación del servicio recibido por el técnico
{
    $numero_solicitud = $this->input->post('numero_solicitud');
    $id_solicitud = Solicitud_model::id_solicitud($numero_solicitud);
    $id_evaluacion = $this->input->post('evaluacion');
    $observacion = $this->input->post('observacion');
    Solicitud_model::insert_evaluacion($id_solicitud, $id_evaluacion, $observacion);
    $this->evaluar();
}
function insert_reclamacion() //función que permite a los usuarios insertar una reclamación por alguna inconformidad en el servicio recibido
{
    $this->load->helper('date');
    $numero_solicitud = $this->input->post('numero_solicitud');
    $id_solicitud = Solicitud_model::id_solicitud($numero_solicitud);
    $datestring = "%m/%d/%Y %h:%i %a";
    $time = time();
    $fecha = mdate($datestring, $time);
    $descripcion = $this->input->post('descripcion');
    Solicitud_model::insert_reclamacion($id_solicitud, $descripcion, $fecha);
    redirect('Autentificacion/login');
}

function resta_fecha() //función para saber si una solicitud de servicio tiene mas de 24 horas
{
$fecha_menor= Solicitud_model::resta_fecha(60)    ;
$fecha_menor= $fecha_menor[0]->fecha;
$fecha_menor = strtotime($fecha_menor);
//$fecha_menor= timespan($fecha_menor);
//echo $fecha_menor ;

/*echo '<br/><br/>';
$fecha_mayor = Solicitud_model::resta_fecha(55);
```

```
$fecha_mayor= $fecha_mayor[0]->fecha;
$fecha_mayor = strtotime($fecha_mayor);
//$fecha_mayor= timespan($fecha_mayor);
//echo $fecha_mayor ;*/

$diferencia =   timespan($fecha_menor);
echo $diferencia;
$tamanno =strlen($diferencia);
if ( $tamanno >20)
{
echo 'Este reporte tiene mas de 24 horas';

}
else
{
echo 'Este reporte esta en tiempo';
}
}

function en_taller() //función que permite conocer las solicitudes que se encuentran en el taller
{
  $reportes_en_taller = Solicitud_model::en_taller();
  if($reportes_en_taller==false)
   {
    echo 'No hay nada';
   }

  else
   {
    return $reportes_en_taller;
   }
}
```

3.3 Conclusiones

Teniendo en cuenta los requerimientos funcionales y no funcionales identificados y el diagrama de clases del diseño propuesto se llevó a cabo la implementación de la aplicación. Además quedaron identificados los componentes que conforman el sistema y sus relaciones. También quedó plasmado el código fuente de las principales clases y pantallas de la aplicación, brindando una descripción del mismo.

Conclusiones

- Se diseñó el sistema. Partiendo del refinamiento de los requerimientos identificados se obtuvieron 14 casos de uso del sistema con sus respectivas descripciones. Se obtuvo además el diagrama de clases del diseño, el diagrama de clases persistentes, el diagrama de despliegue y el modelo de datos, quedando elaborada la propuesta de cómo concebir la aplicación Web, utilizando metodología RUP, lenguaje de programación PHP, el framework de desarrollo CodeIgniter y sistema gestor de base de datos PostgreSQL.
- Se implementó el módulo Solicitud de Servicios del Sistema de Gestión de Tecnología de la Información de la Universidad de las Ciencias Informáticas, cumpliendo con las funcionalidades previstas para el mismo y teniendo en cuenta las normas técnicas y tecnologías aprobadas por la Dirección de Informatización de la UCI para el desarrollo de sistemas.

Recomendaciones

Se recomienda:

- Realizar las pruebas de calidad a la aplicación.
- Confeccionar la ayuda de la aplicación.
- Extender la aplicación a otras instituciones que brinden servicios de soporte técnico.

Referencias Bibliográficas

[1]. Dirección de Informatización. Arquitectura para los Sistemas que Conforman la Intranet Universitaria, 2007. Disponible en: http://uddi.uci.cu/docs/arquitectura.2007.5.9.pdf

[2]. Rodríguez, Mailyn García y Puentes, Lianny de Diego. Modelación del sistema de reportes para la dirección de Gestión Tecnológica de la Universidad de las Ciencias Informáticas, 2007. Disponible en: http://bibliodoc.uci.cu/TD/TD_0355_07.pdf

[3]. Reynoso, Carlos Billy. Introducción a la Arquitectura de Software, 2006. Disponible en: http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/intro.mspx

[4]. Gutiérrez, Fernando García y Delegido, Fernando López. SOA Desarrollos prácticos y bases tecnológicas, 2007.

[5]. Mendoza, Deivis Ricardo Álvarez y Hurtado, Aliennis Mercedes González. PSPSOFT, sistema de apoyo a la docencia para la asignatura de Gestión de Software, 2007. Disponible en: http://bibliodoc.uci.cu/TD/TD_0403_07.pdf

[6]. Ávalos, Heydi Menéndez y Hernández, José Alexander Valdés. Sistema de Gestión de Solicitudes de Insumos del Área de Investigaciones del Centro de Inmunología Molecular, 2007.

[7]. Colectivo de Autores. Achour, Mehdi; Betz, Friedhelm; Dovgal, Antony; Lopes, Nuno; Magnusson, Hannes; Richter, Georg; Seguy, Damien y Vrana, Jakub. Manual de PHP, 2004. Disponible en: http://www.php.net/manual/es

[8]. Colectivo de Autores. Jacobson, Ivar; Booch, Grady y Rumbaugh, James. El Proceso Unificado de Desarrollo Volumen I, The Addison Wesley Longman Inc., 1999.

Bibliografía

1. Dirección de Informatización. Arquitectura para los Sistemas que Conforman la Intranet Universitaria, 2007. Disponible en: http://uddi.uci.cu/docs/arquitectura.2007.5.9.pdf

2. Rodríguez, Mailyn García y Puentes, Lianny de Diego. Modelación del sistema de reportes para la dirección de Gestión Tecnológica de la Universidad de las Ciencias Informáticas, 2007. Disponible en: http://bibliodoc.uci.cu/TD/TD_0355_07.pdf

3. Reynoso, Carlos Billy. Introducción a la Arquitectura de Software, 2006. Disponible en: http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/intro.mspx

4. Gutiérrez, Fernando García y Delegido, Fernando López. SOA Desarrollos prácticos y bases tecnológicas, 2007.

5. Mendoza, Deivis Ricardo Álvarez y Hurtado, Aliennis Mercedes González. PSPSOFT, sistema de apoyo a la docencia para la asignatura de Gestión de Software, 2007. Disponible en: http://bibliodoc.uci.cu/TD/TD_0403_07.pdf

6. Ávalos, Heydi Menéndez y Hernández, José Alexander Valdés. Sistema de Gestión de Solicitudes de Insumos del Área de Investigaciones del Centro de Inmunología Molecular, 2007.

7. Colectivo de Autores. Achour, Mehdi; Betz, Friedhelm; Dovgal, Antony; Lopes, Nuno; Magnusson, Hannes; Richter, Georg; Seguy, Damien y Vrana, Jakub. Manual de PHP, 2004. Disponible en: http://www.php.net/manual/es

8. Colectivo de Autores. Jacobson, Ivar; Booch, Grady y Rumbaugh, James. El Proceso Unificado de Desarrollo Volumen I, The Addison Wesley Longman Inc., 1999.

9. Ayuda extendida del Rational Rose Enterprise Edition 2003.

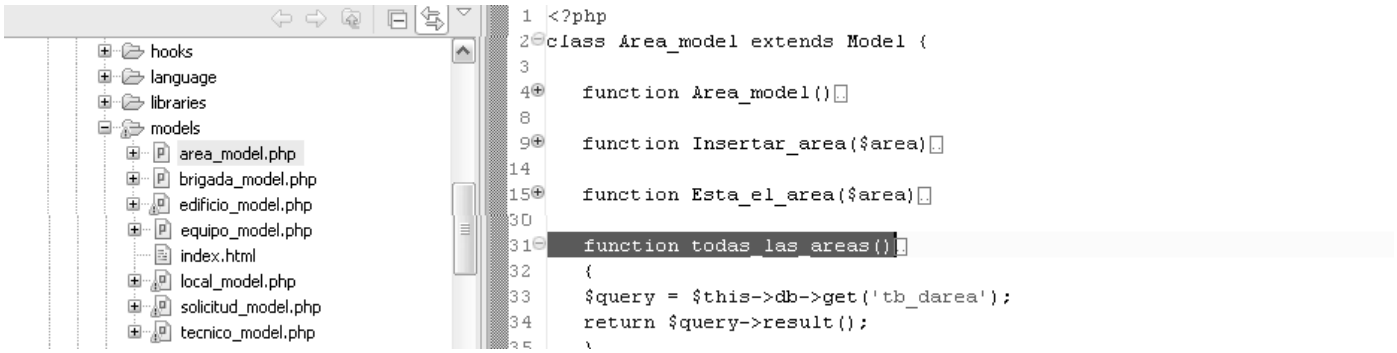
10. CakePHP. Disponible en: http://www.cakephp.org/

11. Symfony. Disponible en: http://www.symfony-project.org/

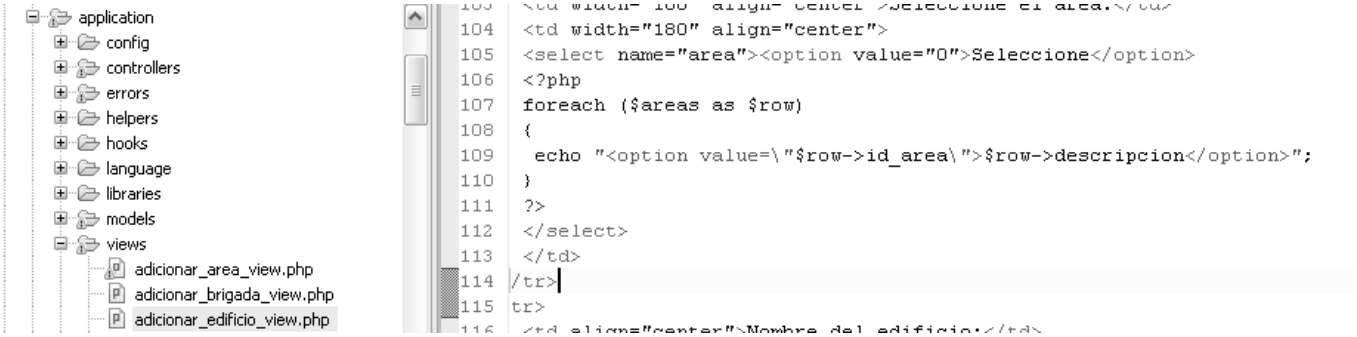
12. CodeIgniter. Disponible en: http://codeigniter.com/

Anexos

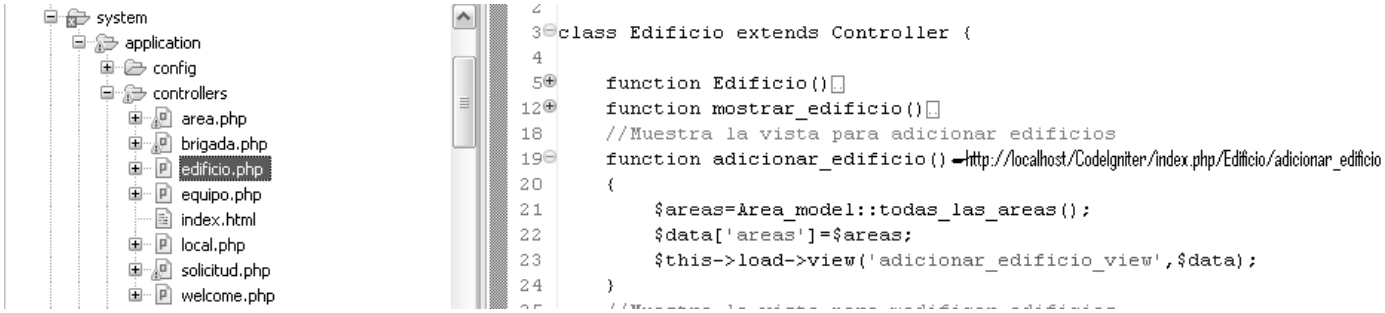
Anexo 1. Ejemplos de código donde se evidencia el patrón MVC.



Anexo 1.1 Ejemplo de un Modelo.



Anexo 1.2 Ejemplo de una Vista.



Anexo 1.3 Ejemplo de un Controlador.

Anexo 2. WSDL de los servicios brindados.

```
<?xml version='1.0' encoding='UTF-8'?>

<!-- WSDL file generated by Zend Studio. -->
```

```
<definitions name="gat_server" targetNamespace="urn:gat_server"
xmlns:typens="urn:gat_server" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
      <types>
            <xsd:schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:gat_server">
                  <xsd:complexType name="ClaveArray">
                        <xsd:complexContent>
                              <xsd:restriction base="soapenc:Array">
                                    <xsd:attribute ref="soapenc:arrayType"
wsdl:arrayType="typens:Clave[]"/>
                              </xsd:restriction>
                        </xsd:complexContent>
                  </xsd:complexType>
                  <xsd:complexType name="EstadoArray">
                        <xsd:complexContent>
                              <xsd:restriction base="soapenc:Array">
                                    <xsd:attribute ref="soapenc:arrayType"
wsdl:arrayType="typens:Estado[]"/>
                              </xsd:restriction>
                        </xsd:complexContent>
                  </xsd:complexType>
                  <xsd:complexType name="SolicitudTallerArray">
                        <xsd:complexContent>
                              <xsd:restriction base="soapenc:Array">
                                    <xsd:attribute ref="soapenc:arrayType"
wsdl:arrayType="typens:SolicitudTaller[]"/>
                              </xsd:restriction>
                        </xsd:complexContent>
                  </xsd:complexType>
                  <xsd:complexType name="Clave">
                        <xsd:all>
                              <xsd:element name="clave" type="xsd:anyType"/>
                              <xsd:element name="id_clave" type="xsd:anyType"/>
                        </xsd:all>
                  </xsd:complexType>
                  <xsd:complexType name="Estado">
                        <xsd:all>
                              <xsd:element name="estado" type="xsd:anyType"/>
                              <xsd:element name="id_estado" type="xsd:anyType"/>
                        </xsd:all>
                  </xsd:complexType>
                  <xsd:complexType name="SolicitudTaller">
                        <xsd:all>
                              <xsd:element name="clave" type="xsd:anyType"/>
                              <xsd:element name="descripcion_problema"
type="xsd:anyType"/>
                              <xsd:element name="estado" type="xsd:anyType"/>
                              <xsd:element name="fecha" type="xsd:anyType"/>
                              <xsd:element name="id_solicitud" type="xsd:anyType"/>
                              <xsd:element name="nombre_equipo" type="xsd:anyType"/>
                              <xsd:element name="nombre_local" type="xsd:anyType"/>
                        </xsd:all>
```

```
                </xsd:complexType>
            </xsd:schema>
        </types>
        <message name="__construct"/>
        <message name="__constructResponse"/>
        <message name="actualizarEstadoClave">
            <part name="idSolicitud"/>
            <part name="idestado"/>
            <part name="idclave"/>
        </message>
        <message name="actualizarEstadoClaveResponse">
            <part name="actualizarEstadoClaveReturn" type="xsd:anyType"/>
        </message>
        <message name="listadoClave"/>
        <message name="listadoClaveResponse">
            <part name="listadoClaveReturn" type="typens:ClaveArray"/>
        </message>
        <message name="listadoEstado"/>
        <message name="listadoEstadoResponse">
            <part name="listadoEstadoReturn" type="typens:EstadoArray"/>
        </message>
        <message name="obtenerResportesEnTaller"/>
        <message name="obtenerResportesEnTallerResponse">
            <part name="obtenerResportesEnTallerReturn"
type="typens:SolicitudTallerArray"/>
        </message>
        <portType name="ServiciosWebPortType">
            <operation name="__construct">
                <input message="typens:__construct"/>
                <output message="typens:__constructResponse"/>
            </operation>
            <operation name="actualizarEstadoClave">
                <input message="typens:actualizarEstadoClave"/>
                <output message="typens:actualizarEstadoClaveResponse"/>
            </operation>
            <operation name="listadoClave">
                <input message="typens:listadoClave"/>
                <output message="typens:listadoClaveResponse"/>
            </operation>
            <operation name="listadoEstado">
                <input message="typens:listadoEstado"/>
                <output message="typens:listadoEstadoResponse"/>
            </operation>
            <operation name="obtenerResportesEnTaller">
                <input message="typens:obtenerResportesEnTaller"/>
                <output message="typens:obtenerResportesEnTallerResponse"/>
            </operation>
        </portType>
        <binding name="ServiciosWebBinding" type="typens:ServiciosWebPortType">
            <soap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
            <operation name="__construct">
                <soap:operation soapAction="urn:ServiciosWebAction"/>
                <input>
                    <soap:body namespace="urn:gat_server" use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
```

```
                </input>
                <output>
                        <soap:body namespace="urn:gat_server" use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </output>
            </operation>
            <operation name="actualizarEstadoClave">
                <soap:operation soapAction="urn:ServiciosWebAction"/>
                <input>
                        <soap:body namespace="urn:gat_server" use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </input>
                <output>
                        <soap:body namespace="urn:gat_server" use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </output>
            </operation>
            <operation name="listadoClave">
                <soap:operation soapAction="urn:ServiciosWebAction"/>
                <input>
                        <soap:body namespace="urn:gat_server" use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </input>
                <output>
                        <soap:body namespace="urn:gat_server" use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </output>
            </operation>
            <operation name="listadoEstado">
                <soap:operation soapAction="urn:ServiciosWebAction"/>
                <input>
                        <soap:body namespace="urn:gat_server" use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </input>
                <output>
                        <soap:body namespace="urn:gat_server" use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </output>
            </operation>
            <operation name="obtenerResportesEnTaller">
                <soap:operation soapAction="urn:ServiciosWebAction"/>
                <input>
                        <soap:body namespace="urn:gat_server" use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </input>
                <output>
                        <soap:body namespace="urn:gat_server" use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </output>
            </operation>
      </binding>
      <service name="gat_serverService">
            <port name="ServiciosWebPort" binding="typens:ServiciosWebBinding">
                <soap:address
location="http://10.10.2.6:5800/CodeIgniter/servicios/serviciosWDSL2.php"/>
            </port>
```

```
      </service>
</definitions>
```

Glosario de Términos

B

BD: Base de **D**atos, es un conjunto de datos que se pueden ingresar, actualizar, eliminar u obtener siempre que sea necesario.

BSD: Licencia de software otorgada principalmente para los sistemas BSD (**B**erkeley **S**oftware **D**istribution) y pertenece al grupo de licencias de software Libre.

G

GAT: Grupo de **A**sistencia **T**écnica.

H

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

HTML: Hyper**T**ext **M**arkup **L**anguage, en ingles. Lenguaje de Marcado de Hipertexto, es el lenguaje de marcas y etiquetas con el que se generan las páginas Web.

HTTP: Hyper**T**ext **T**ransfer **P**rotocol. Protocolo de Transferencia de Hipertextos. Modo de comunicación para solicitar páginas Web.

I

IBM: International **B**usiness **M**achines, conocida coloquialmente como el Gigante Azul, es una empresa que fabrica y comercializa hardware, software y servicios relacionados con la informática.

Interfaz: Permite la interacción entre el usuario y el sistema.

M

MIT: Licencia de software otorgada principalmente para los sistemas MIT (**M**assachusetts **I**nstitute of **T**echnology) y pertenece al grupo de licencias de software Libre.

S

SOAP: Son las siglas en inglés de **S**imple **O**bject **A**ccess **P**rotocol, es un protocolo estándar creado por Microsoft, IBM y otros, que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Es uno de los protocolos utilizados en los servicios Web.

Software Libre: Es el software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente.

U

UCI: Universidad de las **C**iencias **I**nformáticas.

UDDI: Son las siglas del catálogo de negocios de Internet denominado **U**niversal **D**escription, **D**iscovery, and **I**ntegration.

W

Web: Una página Web es una fuente de información adaptada para la World Wide Web y accesible mediante un navegador de Internet. Ésta información se presenta generalmente en formato HTML.

WSDL: Son las siglas de **W**eb **S**ervices **D**escription **L**anguage, un formato XML que se utiliza para describir servicios Web.

WWW: World **W**ide **W**eb, en ingles o telaraña mundial en español. Es un sistema de documentos de hipertexto enlazados y accesibles a través de Internet. Con un navegador Web, un usuario visualiza páginas Web que pueden contener texto, imágenes u otros contenidos multimedia, y navega a través de ellas usando hiperenlaces.

WYSIWYG: Acrónimo del inglés **W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et ("lo que ves es lo que obtienes"). Se refiere a las funciones que brindan los editores de textos que permiten escribir un documento viendo directamente el resultado final.

X

XHTML: Acrónimo del inglés e**X**tensible **H**yper**T**ext **M**arkup **L**anguage (lenguaje extensible de marcado de hipertexto). Es el lenguaje de marcado pensado para sustituir a HTML como estándar para las páginas Web.

XML: Son las siglas en inglés de e**X**tensible **M**arkup **L**anguage (lenguaje de marcado ampliable o extensible). Extensión o tipo de fichero que permite la comunicación, almacenamiento y transmisión de información a través de la Web.