Universidad de las Ciencias Informáticas

Facultad 3

Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas.

Sistema para la reservación de visitas a trabajadores internos en la UCI.

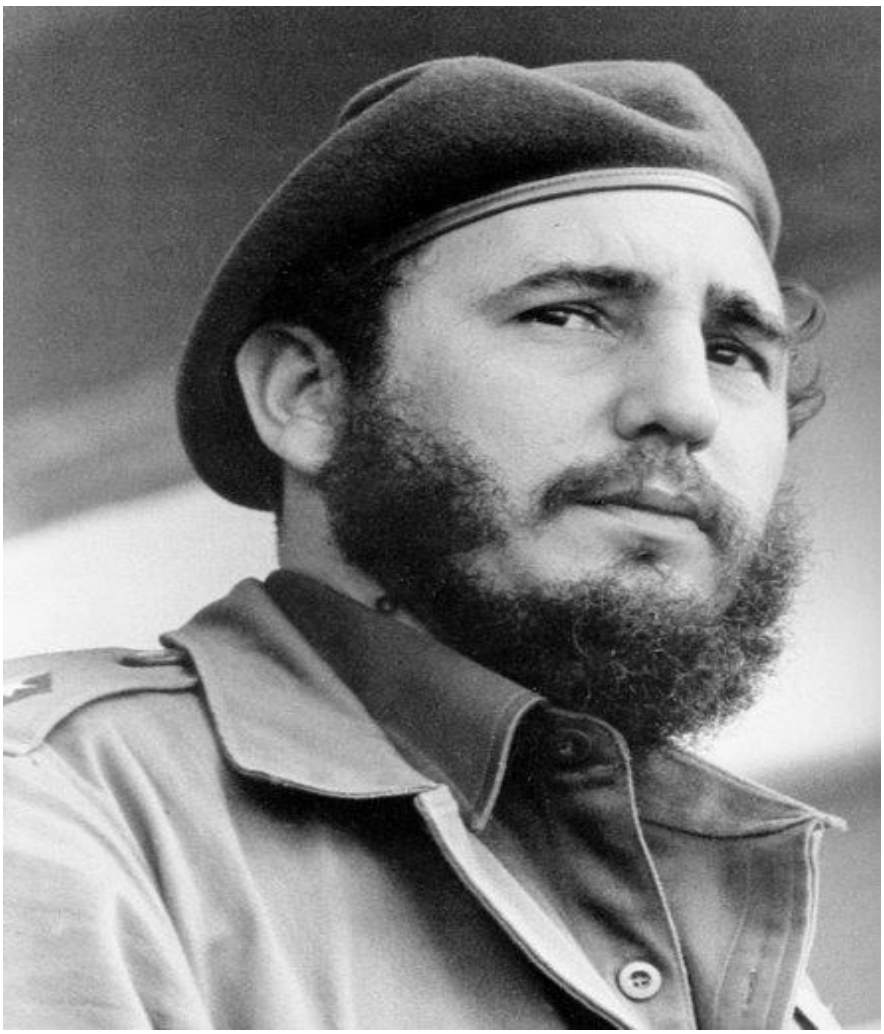
Autor: Jenny Martínez Martínez

Tutora: Ing. Dinia Zayas Romero

La Habana, 2013

"Cuando los hombres llevan en la mente un mismo ideal, nada puede incomunicarlos, ni las paredes de una cárcel, ni la tierra de los cementerios, porque un mismo recuerdo, una misma alma, una misma idea, una misma conciencia y dignidad los alienta a todos"

Fidel Castro Ruz



Declaración de autoría

Declaro ser autor del presente trabajo y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

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

Jenny Martínez Martínez

_____

Firma del Autor

Ing. Dinia Zayas Romero

_____

Firma del Tutor

Datos de Contacto

Tutora: Ing. Dinia Zayas Romero, Ingeniera en Ciencias Informáticas, graduada en el año 2007. Profesora de Práctica Profesional. Se desempeña como jefa del proyecto Cedrux 1.0

Correo electrónico: dzayas@uci.cu

DEDICATORIA

AGRADECIMIENTOS

RESUMEN

El objetivo del presente trabajo se basa en el desarrollo de un sistema de gestión para las reservaciones de visitas a trabajadores internos de la Universidad de las Ciencias Informáticas, que permita la gestión de este proceso de forma tal que garantice el control y ejecución del mismo. Actualmente el registro de las reservaciones de visitas se lleva a cabo de forma manual, y tras el aumento considerable de trabajadores internos, la gestión de la información que se maneja ha crecido notablemente. Esta situación provocó que se considerara necesario el desarrollo de una aplicación web que permita de una forma más rápida y sencilla la reservación de entrada de visitas a trabajadores internos. Además de brindar información relacionada con las reservaciones de forma general, para lo cual se propone una estructura que servirá de base para cumplir con dichas funcionalidades. En este documento se refleja todo el desarrollo del trabajo, especificando la aplicación del modelo de desarrollo elaborado por el Centro de Informatización de la Gestión de Entidades. Para el desarrollo de la misma fueron tomados en cuenta los lenguajes PHP, JavaScript, CSS y XML, regidos por el marco de trabajo Sauxe los cuales permitieron garantizar el desarrollo de un software que responda a las necesidades existentes.

Palabras claves: Reservación, trabajadores internos, visitas.

ÍNDICE

INTRODUCCIÓN

Las Tecnologías de la Información y las Comunicaciones (TIC) se encuentran en constante evolución, estas son usadas para satisfacer las distintas necesidades de la sociedad de resolver un problema dado. Las mismas se han convertido en un factor primordial en el desarrollo científico-evolutivo de la humanidad (1). A pesar de proporcionar características que permiten el desarrollo de la naturaleza innovadora de la sociedad, en ocasiones son rechazadas por el público al que están dirigidas.

Las empresas actuales realizan transformaciones importantes que incluyen tecnologías de información para apoyar en la prestación de servicios. El valor significativo y relevante que la prestación de servicios tiene para determinadas instituciones, muestra que todos los procesos relacionados a la producción, administración y uso de servicios deben ser gestionados y controlados para asegurar la calidad y soporte del cumplimiento de los objetivos de los mismos.

Cuba no se encuentra exenta de esto, en su decisión de lograr un alto nivel de superación de la sociedad cubana y la independencia tecnológica con respecto a los mercados internacionales se crea la Universidad de las Ciencias Informáticas (UCI). La UCI se enfrasca en la puesta en práctica del concepto de vinculación estudio-producción-investigación donde se potencia la formación de los estudiantes y la superación de sus trabajadores desde las actividades productivas.

Para el acceso a la UCI existen puntos de control donde se solicita la identificación a las personas, así como de los vehículos que se dirigen hacia dentro y fuera de la misma. Mediante la verificación de las identificaciones se pretende garantizar la seguridad de los recursos que se han puesto a disposición de estudiantes y trabajadores.

La UCI aloja en su seno ciudadanos de diferentes provincias del país, para los cuales brinda un conjunto de servicios, en aras de mejorar sus condiciones de vida. Entre estos servicios se encuentra la posibilidad de realización de reservaciones para recibir visitas dentro de la residencia de la universidad el cual tiene asociado además la posibilidad de solicitar acceso de vehículos. Mediante la realización de estas reservaciones se pretende controlar y organizar la información relacionada con la estancia transitoria del personal ajeno a la universidad dentro de la misma. A pesar de lo anteriormente planteado este proceso llega a ser engorroso y complejo, debido a diferentes factores, dentro de los cuales se destacan:

- Las reservaciones de dichos servicios se llevan a cabo de forma personal en la oficina correspondiente a cada uno de estos servicios, lugar a donde debe dirigirse el trabajador durante su horario laboral. Esto dificulta el acceso para los trabajadores a estos servicios ya que para realizar

sus solicitudes deben incumplir su horario laboral o ausentarse a otra actividad planificada por su centro de trabajo para ese momento.

- Existencia de gran cantidad de trabajadores internos quienes son participantes directos de esta actividad. Lo que trae como consecuencia la generación de gran cantidad de información asociada a las peticiones realizadas diariamente de estos servicios.
- La peculiaridad que cada servicio obtiene la autorización por personas distintas y en lugares diferentes, lo que provoca que exista descentralización de la información asociada a dichos servicios ya que son manejados y registrados de forma independiente.
- Registro y control manual de la información asociada a dichos servicios. Esto provoca lentitud en el proceso de registro y verificación de la información asociada a dichos servicios, así como pérdida de la misma.

A partir del análisis de lo antes planteado queda definido como **problema a resolver** ¿Cómo mejorar el control y ejecución de las reservaciones de visitas que se ofertan a los trabajadores internos de la UCI?

Se toma como **objeto de estudio**: Gestión de las reservaciones, delimitando el **campo de acción**: Reservaciones de visitas a trabajadores internos de la UCI.

Se define como **objetivo general**: Desarrollar un sistema de reservación de visitas a trabajadores internos que posibilite la ejecución y control de este proceso en la UCI.

Para dar cumplimiento al objetivo general antes planteado se definieron los siguientes **objetivos específicos:**

- Realizar la investigación a partir del Marco Teórico, para conformar una solución bien fundamentada.
- Modelar el negocio asociado al proceso de reservación de visitas, para comprender los elementos significativos del mismo.
- Realizar el análisis y diseño de la solución para determinar la forma de construcción de la solución propuesta.
- Realizar la implementación del sistema para garantizar mayor rapidez en la gestión de las reservaciones.
- Validar la solución desarrollada mediante el método de prueba seleccionado: caja negra.

Sujeto a la **idea a defender**: Si se desarrolla un sistema de reservación de visitas a trabajadores internos, se logrará mejorar la ejecución y control de este proceso en la UCI.

Este trabajo está estructurado por 4 capítulos de la siguiente forma:

CAPÍTULO 1: Fundamentación teórica: Describe el estado del arte de los sistemas de gestión de reservaciones. Además se incluyen los principales conceptos relacionados con el proceso de gestión de reservaciones, así como las técnicas, tecnologías, software usados en la actualidad, la fundamentación del uso de los lenguajes, herramientas y modelo de desarrollo del software.

CAPÍTULO 2: Características del Sistema: Este capítulo encierra las descripciones pertenecientes a los procesos de negocio, así como el levantamiento de los requisitos funcionales y no funcionales, que sirven de entrada a la realización de los diagramas correspondientes a la disciplina de análisis y diseño.

CAPÍTULO 3: Análisis y Diseño: Se describen los patrones utilizados para la definición de la arquitectura y el diseño. Se realizan los diagramas correspondientes al diseño y el almacenamiento de la información en la base de datos, los cuales intervienen de forma directa en la solución.

CAPÍTULO 4: Implementación y pruebas: Se muestran los estándares de codificación empleados para posibilitar la organización del desarrollo. Se muestran además los resultados obtenidos de la validación del diseño y de la realización de pruebas al software.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

El presente capítulo aborda los conceptos más importantes manejados para la realización del Sistema para la reservación de visitas a trabajadores internos en la UCI. Donde serán incluidos además los sistemas que existen actualmente para realizar funciones similares a las que debe realizar dicha propuesta. Son mencionadas las principales tecnologías, los lenguajes; así como las herramientas y el modelo que serán usados para desarrollo de la aplicación dando una breve descripción de las mismas.

1.2. Conceptos fundamentales

Para comprender el entorno donde se enmarca el presente trabajo, se presentan los conceptos fundamentales a tener en cuenta durante el desarrollo del mismo.

Servicio

Se refiere a una prestación para satisfacer las necesidades de las personas que no representa la producción de bienes materiales; la cual puede ser realizada bajo cierto control y regulación por la organización o entidad por la que es desarrollado con el fin de satisfacer a la sociedad (2). A partir de lo antes planteado se puede determinar que la posibilidad de reservación de visitas y de acceso de vehículos se puede categorizar como la prestación de un servicio debido a que están encaminados a la satisfacción de los trabajadores internos en la UCI.

Gestión

El término se refiere a las diferentes acciones desarrolladas con el fin de lograr la evolución de determinada operación enfrascada fundamentalmente a las actividades comerciales y trae como resultado la administración o control de la misma (3). Rigiéndose por el concepto antes señalado y enmarcado en esta investigación se puede denominar gestión al proceso llevado a cabo para el registro, verificación y control de entrada de visitas a trabajadores internos en la UCI.

Reservación

Se encuentra enmarcado fundamentalmente a las áreas de los servicios (hotelería, transporte, restaurantes) el cual esta destinado al uso exclusivo de un lugar por una persona, donde es realizada la solicitud con previa antelación a que sea brindado (4). A partir de esto puede ser categorizado como reservación a las solicitudes de autorización de entradas de vehículos y personal ajeno a la UCI realizadas por los trabajadores con previa antelación al ingreso de las mismas.

1.1 Estado de los sistemas de reservaciones en el mundo

En la actualidad existen distintas herramientas desarrolladas con el objetivo de gestionar reservaciones. A pesar de que solo existe una que se base en la especificación de las reservaciones de visitas del tipo que es necesario en la universidad, fueron analizadas otras que podrían aportar al desarrollo de la solución. Analizando la bibliografía consultada se pudo encontrar la propuesta de algunos productos web que incorporan algunas de las características que se proponen para el producto final. Las características a analizar con respecto a los productos web son:

- Realizar solicitud de acceso de visitas.
- Realizar solicitud de acceso de vehículos.
- Disponibilidad del servicio.
- Permitir almacenamiento de gran cantidad de información.

1.2.1. Internacionales

Sabre

Es una aplicación web, que permite en tiempo real monitorear listas de espera de las reservaciones realizadas ya sea para asientos de avión o habitaciones de hotel. Lo cual permite que en caso de ocurrir variaciones en las reservaciones existentes se tenga constancia inmediata, lo que posibilita minimizar las pérdidas. Tiene como objetivo automatizar el control de reservaciones para poder optimizarlos en función de las necesidades reales de la entidad.

Características:

- Gestiona las reservaciones a través de Internet para empresas y agencias de viajes a tiempo real.
- Estructura de forma gráfica la organización de la información, puede trabajar una gran cantidad de reservas en muy pocos minutos, realizando todos los pasos necesarios para una búsqueda eficaz de acuerdo a la necesidad puntual de cada cliente.
- Disponibilidad las 24 horas del día, además de posibilitar realizar variaciones en los horarios previamente programados.
- Proporciona almacenamiento para gran cantidad de información (5).

Este sistema incluye dentro de sus funcionalidades la posibilidad de realizar reservaciones de visitas a hoteles, donde proporciona una notificación inmediata en caso de existir alguna variación; funcionalidad que puede ser tomada en cuenta para la realización de la solución propuesta. Este sistema no tiene en cuenta la

posibilidad de realización de solicitudes de acceso de visitas y vehículos por lo que no es considerada una posible solución para los problemas existentes con respecto a estos servicios en la UCI.

Planyo

Es un sistema de reservas en línea que puede ser usado para hoteles, rentas de autos o yates, consultas médicas o terapeutas y salas de reuniones (6). Permite realizar las reservaciones desde una pc o en un móvil y en 20 idiomas, ejecutar configuraciones muy avanzadas sin necesidad de tener conocimiento técnico.

Características:

- Gestiona reservaciones de habitaciones y medios de transporte por roles definidos previamente en el sistema a tiempo real mediante el uso de Internet.
- Estructura de forma gráfica la organización de la información, mediante la cual es posible definir los elementos a tener en cuenta para la información asociada a las reservaciones.
- Permite aumentar el número de reservaciones dando la información de disponibilidad, además del envió de confirmación instantánea.
- Disponible las 24 horas del día.
- Proporciona almacenamiento para gran cantidad de información (7).

Este sistema incluye dentro de sus funcionalidades la posibilidad de realizar reservaciones de visitas a hoteles, casas de vacaciones, así como el alquiler de vehículos, donde proporciona una interfaz configurable y adaptable a las necesidades de la empresa que solicita su uso la cual puede ser tomada en cuenta para la realización de la solución propuesta. A pesar de estas características no es tomado en cuenta como posible solución ya que no permite la realización de solicitudes de acceso de vehículos.

1.2.2. Nacionales

Gestión Universitaria

Es una aplicación web, que emplea como gestor de base de datos PostgreSQL, servidor web Apache, como arquitectura es empleada Guud desarrollada en el Centro de Informatización Universitaria (CENIA) el cual es el encargado del desarrollo de esta solución. Para el desarrollo de esta aplicación fueron empleadas las librerías Codeigniter y JQuery, con los lenguajes PHP y JavaScript respectivamente. El sistema se encuentra en el periodo de prueba y cuenta entre sus funcionalidades la posibilidad de registro de reservaciones de visitas.

Características:

- Permite la gestión de reservaciones por roles definidos previamente en el sistema.
- Estructura de forma gráfica la organización de la información, mediante la cual es posible definir los elementos a tener en cuenta para la realización de transformaciones a las reservaciones de visitas.
- Proporciona almacenamiento para gran cantidad de información (8).

Este sistema no se encuentra en explotación, posibilita la realización de reservaciones por roles definidos previamente en el sistema, funcionalidad que puede ser empleada en el desarrollo de la solución propuesta. A pesar de ser un sistema realizado con el fin de la informatización del proceso de reservación de visitas no tiene en cuenta los requisitos necesarios para la solicitud de entrada de transporte. Además de no incluir la posibilidad de ser realizada una reservación directamente por el trabajador interno. Por lo que no brinda una solución al problema que se refiere a que los trabajadores deban dirigirse a la oficina para realizar la solicitud de la reservación en horario laboral.

1.2.3. Valoración de los sistemas estudiados

Las investigaciones realizadas arrojaron como resultado que no existe ningún sistema que cumpla con todas las características definidas por la UCI para la reservación de visitas. Aunque cubran escenarios comunes que pueden ser tomados en cuenta para la realización de esta solución, como pueden ser utilizadas algunas de las concepciones brindadas para la realización de sus funcionalidades, sirviendo de ayuda además sus respectivas interfaces de usuario.

Los sistemas estudiados a nivel mundial incluyen funcionalidades necesarias para la realización del proceso de reservación de visitas, pero tienen la desventaja que utilizan tecnologías disponibles en el mercado. Al ser privativos estos sistemas se dificulta el acceso al código fuente del programa y niegan el derecho de poder copiar alguna porción de código o interfaz de usuario y modificarlo de acuerdo con las necesidades deseadas. La descripción de cómo realizan estos procesos y las características de dicho software, así como la documentación, inclúyase manuales de usuario y vistas de diseño solo pueden ser obtenidas mediante acuerdo comercial.

En cuanto a los nacionales se percibe una carencia de aplicaciones que dispongan de los elementos que requiere la universidad para el registro del servicio de reservación. Estos no brindan una alternativa de solución a la mayoría de los problemas existentes debido a que no tienen en cuenta la necesidad de la presencia del trabajador durante su horario laboral en las oficinas de reservaciones. No tienen en cuenta una posible solución para la descentralización de la información asociada a dichos servicios ya que son

manejados de forma independiente y en el caso de la reservación de vehículos se encuentra excluida en el sistema analizado.

Estos elementos evidencian la necesidad de elaborar una solución que proporcione una respuesta a los procesos de reservación de visitas. Con la nueva propuesta de sistema se pretende garantizar un mejoramiento en cuento a ejecución y control de este proceso.

1.3. Modelo de Desarrollo

El modelo escogido para guiar el desarrollo de esta solución es el definido por el Centro de Informatización de la Gestión de Entidades (CEIGE) versión 1.1, que es el resultado de la combinación de modelos que poseen como características principales ser basado en Componentes, Iterativo e Incremental. En dicho modelo se incluye la especificación de las actividades de cada una de las fases del ciclo de vida de los proyectos que se ejecutan en el CEIGE teniendo en cuenta los procesos de CMMI (*Capability Maturity Model Integration*) nivel 2 para la UCI. Se detallan por tanto los artefactos a generar en cada momento, independientemente de las herramientas o métodos que se utilicen para ello.

Un desarrollo basado en componentes permite alcanzar un mayor nivel de reutilización de software, aún en contextos distintos de aquellos para los que fue diseñado. Permite además que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados. También, al existir un débil acoplamiento entre componentes, se puede actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema. Otro de los beneficios que ofrece es que dado que un componente puede ser construido y luego mejorado continuamente, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo (9).

Para el desarrollo de la solución propuesta no se tiene en cuenta la disciplina de pruebas de liberación ya que por razones de tiempo no es posible satisfacer todas en el periodo comprendido.

Este modelo trae asociado consigo el secionamiento por varias disciplinas las que traen aparejadas un grupo de artefactos que seran reflejados a continuacion:

Disciplina Inicio o Estudio preliminar.

- Cronograma donde se listan las actividades a realizar.

Disciplina Modelado del negocio.

- Reglas del negocio.
- Mapa de procesos de negocio.
- Modelo conceptual.

- Descripción de proceso de negocio.
- Glosario de términos.

Disciplina Requisitos

- Lista de requisitos.
- Especificación de requisitos de software.
- Criterios para validar requisitos.

Disciplina Análisis y diseño.

- Diagrama de clases del diseño.
- Diagrama de interacción (secuencia o colaboración por requisito).
- Modelo de datos.
- Diagrama de componentes.

Disciplina Implementación

- Ficheros de código.

Disciplina Pruebas internas

- Criterios para validar el diseño.
- Diseño de casos de prueba.
- Acta de liberación

1.4. Tecnologías y Herramientas.

Para el desarrollo del sistema se utilizarán varias tecnologías y herramientas, que ayudarán a lograr de forma más fácil su construcción.

1.4.1. Modelado.

BPMN

BPMN (Business Process Modeling Notation) es empleado en el desarrollo de la solución que se propone para presentar gráficamente las diferentes etapas del proceso. Este estándar de modelado de procesos de negocio ha sido diseñado específicamente para coordinar la secuencia de procesos y los mensajes que fluyen entre los diferentes procesos participantes. BPMN a diferencia del lenguaje de modelado unificado (UML) toma un perfil orientado a procesos en el modelado de sistemas, por lo tanto la combinación de ambas notaciones, al ser compatibles entre sí, ayudan a modelar con mayor precisión la situación actual y deseada en los procesos de negocio del cliente (10). La misma permitirá el modelado de

los artefactos correspondientes a los procesos identificados en el negocio como son el mapa de procesos y el diagrama de procesos pertenecientes al negocio analizado.

UML

Lenguaje de Modelado Unificado es empleado para representar gráficamente la solución propuesta, es un lenguaje que permite visualizar, especificar y construir cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar los procesos de negocio y funciones de sistema, además de elementos concretos como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables. Cuenta entre sus características que es independiente del proceso de desarrollo, está compuesto por tres clases de bloques de construcción: elementos que son abstracciones de elementos reales o ficticias (objetos, acciones, etc.), relaciones señalan las interconexiones que presentan los elementos entre sí y los diagramas que son colecciones de elementos con sus relaciones (11). Este será empleado en la elaboración del modelo conceptual así como los diagramas correspondientes a la disciplina de análisis y diseño.

1.4.2. Marco de trabajo

Sauxe 2.2

Es un marco de trabajo que fue desarrollado en la UCI como fruto del paradigma de independencia tecnológica por el que aboga el país. Este marco de trabajo, fusionado bajo tecnologías totalmente libres (entre ellas PHP, PostgreSQL, Apache) posee el desarrollo de tecnologías propias basadas en otros marcos de trabajo como Zend Framework para el manejo de la lógica de negocio, Doctrine para el acceso a datos y ExtJS para la capa de presentación. Cuenta con el estilo arquitectónico n-capas que a su vez presenta en su capa superior el patrón arquitectónico modelo vista controlador (MVC). Contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo (12).

Entre los elementos asociados al funcionamiento de este marco de trabajo se encuentran:

AJAX

JavaScript asíncrono y XML es la técnica de desarrollo Web para aplicaciones interactivas. Engloba a todo un grupo de tecnologías (XHTML, JavaScript, CSS, API y DOM) y mantiene una comunicación asíncrona con el servidor en segundo plano, lo que permite realizar continuos cambios sin necesidad de recargar las páginas (13).

ExtJS 2.2

Es una librería JavaScript que está basada completamente en la programación orientada a objetos. Cada objeto contiene lo típico: propiedades, métodos y eventos. Basa toda su funcionalidad en JavaScript a través de librerías. Así, en tiempo de ejecución carga y crea todos los objetos HTML a través del uso intenso de DOM. Los datos son obtenidos con AJAX a través de XML. Una de las grandes ventajas de utilizar ExtJS es que permite crear aplicaciones complejas utilizando componentes predefinidos. Existe un balance entre Cliente–Servidor. La carga de procesamiento se distribuye, permitiendo que el servidor al tener menor carga, pueda manejar más clientes al mismo tiempo (14).

Doctrine 1.0

Este ORM (Object Relation Mapper) es una técnica de programación que permite que las tablas de la base de datos pasen a ser clases y los registros de objetos se pueden manejar con facilidad. Presenta como ventaja la reutilización permitiendo llamar a los métodos de un objeto de datos desde distintas partes de la aplicación e incluso desde diferentes aplicaciones, seguridad ya que suelen implementar mecanismos que protegen la aplicación de los ataques más comunes como inyecciones SQL (15).

Zend Framework 1.5

Este presenta código abierto para el desarrollo de aplicaciones y servicios web con PHP 5. Es una implementación que usa código orientado a objetos. La estructura de cada componente está construida con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado. Hace énfasis fundamentalmente en la calidad del código, a través de una batería de pruebas unitarias, utilizando PHPUnit, cubriendo el código escrito para el framework. Persigue como objetivo la meta de lograr simplicidad, busca además tener una Interfaz de Programación de Aplicaciones (API) muy fácil de aprender (16).

1.4.3. Lenguajes.

PHP 5.2

Hypertext Preprocessor es un lenguaje script (no se compila para conseguir códigos máquina si no que existe un intérprete que lee el código y se encarga de ejecutar las instrucciones que contiene este código), para el desarrollo de páginas web dinámicas del lado del servidor, cuyos fragmentos de código se intercalan fácilmente en páginas HTML, debido a esto, y a que es de código abierto, es el más popular y extendido en la web. Es capaz de realizar determinadas acciones de una forma fácil y eficaz sin tener que generar programas desarrollados en un lenguaje distinto al HTML (17).

JavaScript 1.1

Es un lenguaje de programación que se puede utilizar para construir sitios web y para hacerlos más interactivos. Aunque comparte muchas de las características y de las estructuras del lenguaje Java, fue desarrollado independientemente. El lenguaje JavaScript puede interactuar con el código HTML, permitiendo a los programadores web utilizar contenido dinámico. El lenguaje JavaScript es de código abierto, por lo que cualquier persona puede utilizarlo sin comprar una licencia.

Es utilizado para crear pequeños algoritmos y funciones, encargados de realizar acciones dentro del ámbito de una página Web. Es simple y permite concretar funcionalidades con alto grado de rapidez. Su empleo es recomendable aún para personas con poca experiencia en programación, pues su sencillez permite asimilarlo y practicarlo con facilidad (18).

CSS

Hojas de Estilo en Cascada son utilizadas para representar todo lo referente a los estilos (dígase tamaños, colores, iconos, imágenes, tipografías, espacios y bordes). Constituyen el estándar para la inserción de estilos a documentos estructurados, como por ejemplo, páginas HTML o XML. El objetivo de la definición de este estándar es permitir la separación entre las normas de presentación y el propio contenido a mostrar (19).

XML

Lenguaje de Etiquetado Extensible, es un metalenguaje de definición de documentos estructurados mediante marcas o etiquetas es usado en la creación de las reglas básicas que permiten el intercambio de información estructurada entre aplicaciones; se emplea también para tareas de validación y configuraciones en el sistema. Hace posible compartir la información de una manera segura, fiable y fácil; además de los datos con los que se trabaja a todos los niveles, por todas las aplicaciones y soportes. Es también un lenguaje del lado del servidor (20).

HTML

Lenguaje de marcado hipertextual, es utilizado para definir páginas clientes de la aplicación. Se compone por un conjunto de etiquetas utilizadas para definir y ubicar los distintos elementos que componen una página web. Como un lenguaje de marcación de elementos para la creación de documentos hipertexto, HTML puede describir hasta un cierto punto la apariencia de un documento. Puede incluir uno o varios scripts, como por ejemplo: JavaScript o PHP, los cuales pueden afectar el comportamiento del HTML. Su

principal desventaja es que todos los navegadores no interpretan el código HTML de la misma manera (21).

1.4.4. Servidor Web

Apache 2.0

Es una tecnología gratuita de código abierto compatible con muchos Sistemas Operativos. Tiene todo el soporte que se necesita para tener páginas dinámicas. Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Tiene una alta configuración en la creación y gestión de registros de actividad. Apache permite la creación de ficheros de registro a medida del administrador, de este modo se puede tener un mayor control sobre lo que sucede en el servidor (22).

1.4.5. Herramientas

Visual Paradigm 6.4

Es una herramienta CASE (Ingeniería de Software Asistida por Ordenador) que utiliza UML como lenguaje de modelado, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Ayudando a construir aplicaciones de calidad más rápido, mejor y en bajo costo (23).

RapidSVN (Subversión) 1.6

Es una herramienta que está preparado para funcionar en red, y se distribuye bajo una licencia libre de tipo Apache. Esta constituyó el estándar de facto de los sistemas de gestión de versiones en el ámbito del software libre. Cuenta entre sus características que mantiene versiones no solo de archivos, sino también de directorios; también se mantienen versiones de los metadatos asociados a los directorios. Además de los cambios en el contenido de los documentos, se mantiene la historia de todas las operaciones de cada elemento, incluyendo la copia, cambio de directorio o de nombre (24).La misma será empleada para el control de la documentación asociada a la investigación así como de la solución resultante.

PostgreSQL 9.0

Es un avanzado sistema de bases de datos Objeto Relacional de código abierto, estable, de alto rendimiento y con gran flexibilidad. Permite mantener la integridad de los datos y puede manejar múltiples conexiones concurrentes de los clientes, característica muy significativa de su funcionamiento. Cuenta con una arquitectura que se caracteriza por su confiabilidad, permite una manipulación potente, flexible y

eficiente de la información. Posee una interfaz amigable y confiable que guía a los usuarios con menor experiencia a través del complejo proceso de creación haciendo todo más veloz y dinámico. Es muy portable, por lo que se puede ejecutar en la gran mayoría de sistemas operativos existentes en la actualidad y funciona perfectamente con grandes cantidades de datos (25). La cual posibilitará el almacenamiento de la información asociada a las reservaciones realizadas por los trabajadores internos.

NetBeans 7.0

Es una herramienta para programadores que permite escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extenderlo. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso. (26) Presenta entre sus características que cuenta con soporte para procesadores de anotaciones en el editor, configurable en las propiedades del proyecto, auto-completado de código y links para atributos de CSS, soporte para PHP Zend Framework (27).

1.5. Conclusiones parciales

Una vez finalizado el presente capítulo se pudo arribar a las siguientes conclusiones:

- Los conceptos analizados ayudan a lograr un mejor entendimiento de la investigación en cuestión, conformando la fundamentación teórica que sustenta el desarrollo y propiciando a que el lector adquiera conocimientos acerca del tema a tratar.
- El análisis realizado a sistemas informáticos que implementan alguna solución para la gestión de reservaciones permitió arribar a la conclusión de que no existe ningún sistema que satisfaga las necesidades existentes en la UCI en aras de una mejor organización de estos procesos.
- La presentación de las tecnologías, herramientas, lenguajes de modelado y desarrollo analizados, permitió tener un mayor conocimiento de los mismos para su posterior utilización.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

2.1. Introducción

En el presente capítulo se describen las características esenciales que debe cumplir la solución que se propone, las cuales se encuentran sustentadas por la modelación de los procesos de negocio asociados al funcionamiento actual del proceso de reservación de visitas. Se figuran, además, algunos de los artefactos propuestos en la fase de Modelación definida en el modelo de desarrollo aplicado.

2.2. Modelado de los procesos de negocio.

El modelado de procesos de negocio permite la definición de la base para percibir mejor el trabajo de una organización o entidad, documentar y publicar los procesos buscando una estandarización en la organización (28). De esta manera se identifican las necesidades puntuales, así como limitaciones que presenta el entorno, para lograr una comprensión de las características del negocio a través de la descripción de los procesos.

2.2.1. Mapa de procesos del negocio.

El mapa de procesos permite la identificación y establecimiento de los elementos que contiene el negocio. A partir del estudio del comportamiento actual del negocio fueron identificados los procesos y documentos fundamentales que intervienen, entre los que se encuentran:

Realizar reservación de visitas: Se refiere al proceso que se realiza diariamente donde el trabajador interno se dirige a la oficina de reservación, es atendido por la instructora, y solicita reservación de una visita. Este debe cumplir las características de pertenecer al primer grupo de consanguinidad (madre, padre, hermano, tío, primo), contener al menos un apellido igual al que solicita el servicio o ser pareja conyugal, que la visita sea por un periodo menor o igual a tres días, que no haya realizado más de dos reservaciones en esa semana o posea otra para este mismo familiar. Como resultado de la aplicación de este proceso es obtenido el Listado de visitas.

- Listado de visitas: Se refiere al documento resultante del proceso de realizar reservación de visitas que recoge el registro de las reservaciones aprobadas ese día por cualquiera de los implicados en este proceso.

Determinar entrada de automóvil: Se refiere al proceso que se realiza diariamente donde el trabajador interno se dirige a la oficina del director general o a la vicerrectoría de extensión, es atendido por cualquiera de los que desempeña dicha responsabilidad, y solicita la entrada de un vehículo estatal o particular y estos

determinan la aprobación o rechazo de la misma. Como resultado de este proceso es obtenido el Listado de automóviles.

- Listado de automóviles: Se refiere al documento resultante del proceso de determinar entrada de automóvil quien recoge el registro de las reservaciones de entrada de vehículos aprobadas ese día por cualquiera de los implicados en este paso.

Ejecutar control de entrada: Se refiere al proceso que se realiza diariamente cuando un visitante llega a la oficina de atención al visitante y solicita la autorización de entrada a la universidad, este es atendido por el técnico quien verifica la existencia de su reservación en el listado correspondiente, comportándose de la misma forma en caso de que vaya a ingresar en un vehículo. Como resultado se le es entregado en caso de existir su reservación el pase de visita o el de automóvil en correspondencia a su reservación.

- Pase de visita: Se refiere al documento entregado que acredita la autorización de entrada del visitante.
- Pase de automóvil: Se refiere al documento entregado que acredita la autorización de entrada del vehículo.

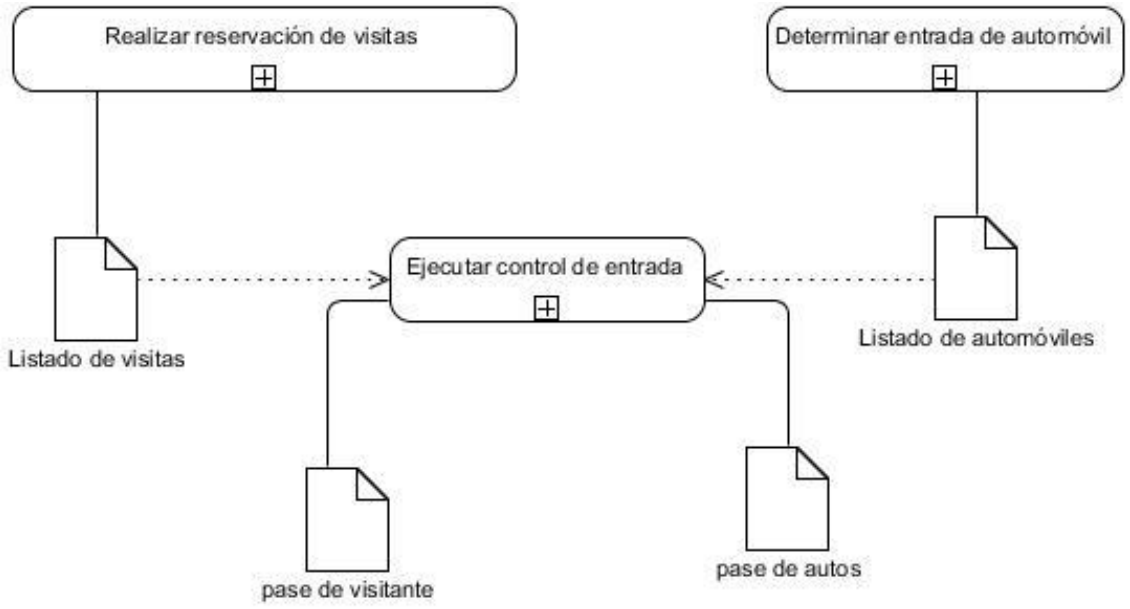


Fig. 1: Mapa de procesos de negocio.

2.2.2. Diagrama del proceso de negocio.

Los procesos de negocios constituyen un conjunto de tareas organizadas para conseguir un objetivo concreto. El mismo proporciona facilidades para determinar las personas que forman parte de un proceso,

limitando la actividad y responsabilidad de cada individuo. Lograr la estandarización de la ejecución de los procesos permite que se consiga dar respuesta a los objetivos definidos (29). Para la ejemplificación del diagrama de procesos de negocios con su respectiva descripción es empleado el proceso Realizar reservación de visitas, el resto de los mismos pueden ser consultados en el expediente asociado al desarrollo esta solución.

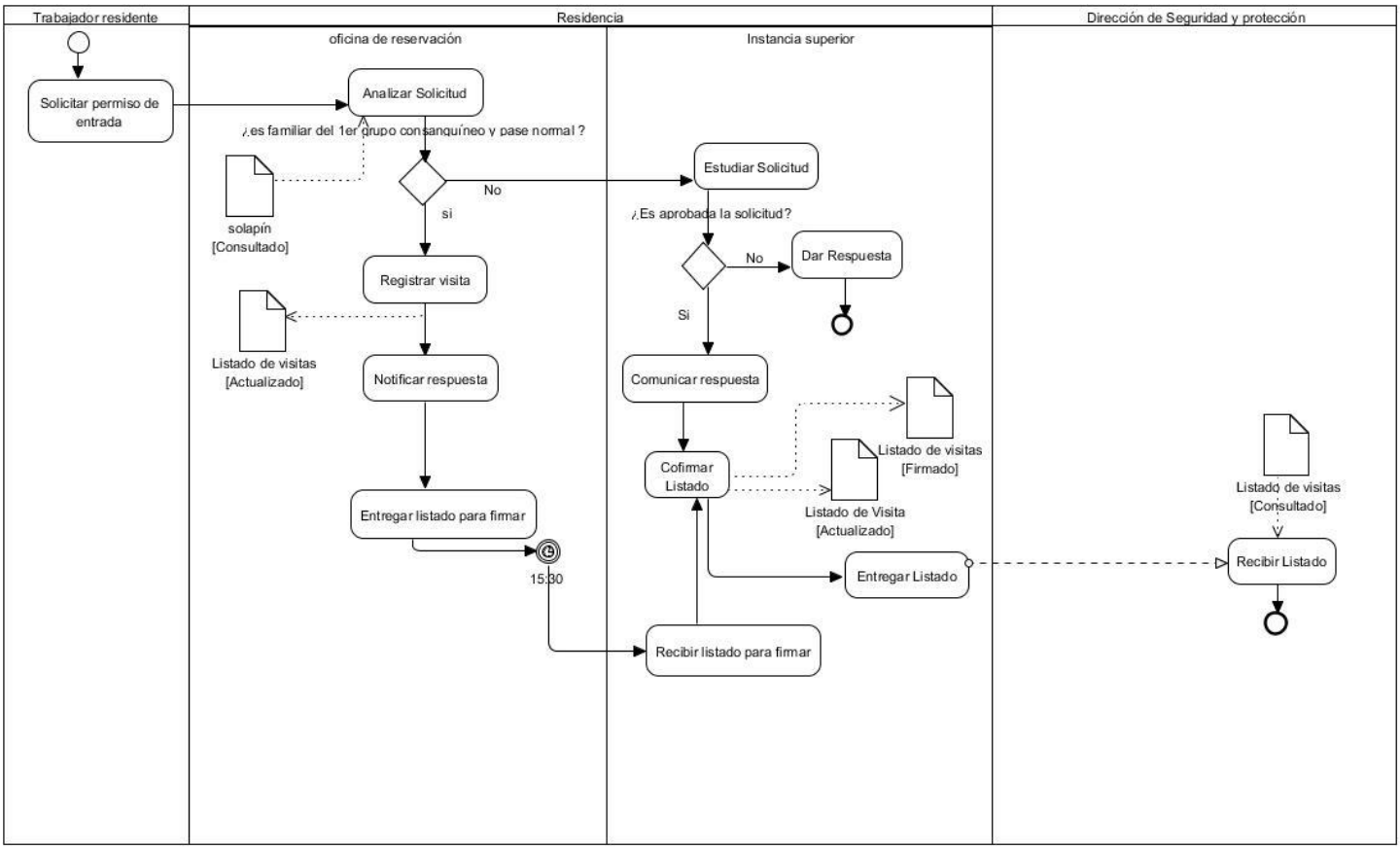


Fig. 2: Diagrama de procesos de negocio. Realizar reservación de vistas.

2.2.3. Descripción del proceso del negocio.

Tabla 1: Descripción de procesos de negocio. Realizar reservación de visita.

Objetivo	Obtener la información necesaria para el registro de una reservación de visita durante un periodo de tiempo determinado.
Evento(s) que lo genera(n)	Dirigirse el trabajador interno a la oficina de

	reservación.
Pre condiciones	
Marco legal	N/A
Clientes externos	
Entradas	Solapín del trabajador.
Flujo de eventos	
Flujo básico Realizar reservación de visita	
1.	Solicitar permiso de entrada: el trabajador interno solicita el servicio de reservación de visita.
2.	Analizar solicitud: la asistente de la dirección de residencia de trabajadores se encarga de analizar si los datos suministrados (nombre, apellidos, parentesco, fecha de entrada y fecha de salida) por el trabajador. Si la solicitud realizada cumple con las condiciones de ser pase normal y que el familiar se encuentre en el primer grupo de consanguinidad (madre, padre, abuelo, hermano, tío, primo que cargue uno de sus apellidos o esposo) es aprobado la solicitud. En caso de no cumplirse se ejecuta el **Flujo alternativo 2a.**
3.	Registrar visita: la asistente de la dirección adiciona al listado de visitas al familiar solicitado.
4.	Notificar respuesta: la asistente le comunica al trabajador que fue concedida su solicitud.
5.	Enviar listado listo para firmar: Esta actividad es realizada todos los días a las 15:30 al culminar la jornada en la oficina de reservación donde la asistente le suministra el listado de visitas al director de residencia.
6.	Recibir Listado: El director de residencia recibe el listado suministrado por la asistente.
7.	Confirmar Listado: el director de residencia adiciona al listado de visitas algún familiar en caso de existir que haya sido aprobado por él personalmente y posteriormente para declarar la validez del listado de visitas es firmado y acuño el documento.
8.	Entregar listado: luego de haber confirmado los datos pertinentes, con las respectivas firmas necesarias es entregado dicho listado para el control de entrada.
9.	Recibir listado: el listado es recogido por un encargado en la oficina de seguridad y protección para su aplicación.
10.	Se termina el proceso.

Pos-condiciones
Queda actualizado el listado de visitas con acceso permitido a la universidad.
Salidas
1. Listado con las visitas a ingresar.
Flujos paralelos
Salidas
N/A
Flujos alternos Estudiar solicitud
Flujo alternativo 2a.
1. Si el visitante no cumple con los requisitos señalados es remitido a la oficina del director general de residencia directamente quien determina la aprobación o no de la entrada.
2. En caso de ser afirmativa la respuesta el director general de residencia se lo da a conocer al trabajador. Se ejecuta el Flujo básico 7.
Flujos alternos Dar respuesta
Flujo alternativo 2a.1
1. Si la solicitud realizada no es considerada valida por el director general de residencia se le comunica al trabajador que fue rechazada la solicitud de entrada de la visita.
2. Se termina el proceso.
Salidas
N/A
Asuntos pendientes
N/A

2.2.4. Modelo conceptual.

El modelo conceptual permite interrelacionar conceptos de la realidad física no propios en un sistema de software lo que no significa que no sean tomados en cuenta para su desarrollo. El mismo puede utilizarse para capturar y expresar el entendimiento alcanzado en un área de análisis como paso previo al diseño de

un sistema de software. Es utilizado por el analista como un medio para comprender el sector industrial o de negocios al cual el sistema va a servir (30).

Conceptos Fundamentales del Dominio

Para brindar una mejor comprensión del diagrama del modelo de dominio o conceptual a continuación se realiza una breve descripción de los conceptos asociados al funcionamiento del área de análisis.

- Trabajador interno: es el encargado de dar inicio al proceso al dirigirse a la oficina de reservación a solicitar el servicio proporcionado por la misma.
- Reservación de visita: se refiere a uno de los servicios prestados, respondiendo directamente al que es realizado con el fin de obtener el permiso de entrada del visitante.
- Vehículo: se refiere a uno de los involucrados indirectamente en el proceso de permiso de entrada.
- Reservación de vehículo: se refiere a uno de los servicios que incluye el proceso, indicando directamente a la obtención del permiso de entrada en vehículo, la cual presenta la particularidad que solo pueden ser aprobadas por los directivos.
- Visitante: se refiere a uno de los beneficiados con el permiso de entrada a la institución.
- Directivo: se refiere a todos los que influyen en el proceso de reservación como instancia superior (director de residencia, director general de residencia, vicerrector de extensión) que se encargan de firmar, cuñar y actualizar en caso de ser necesario los documentos que son generados.
- Instructora: también denominada asistente, son las encargadas de registrar en el listado de entradas a los visitantes luego de su previa autorización.

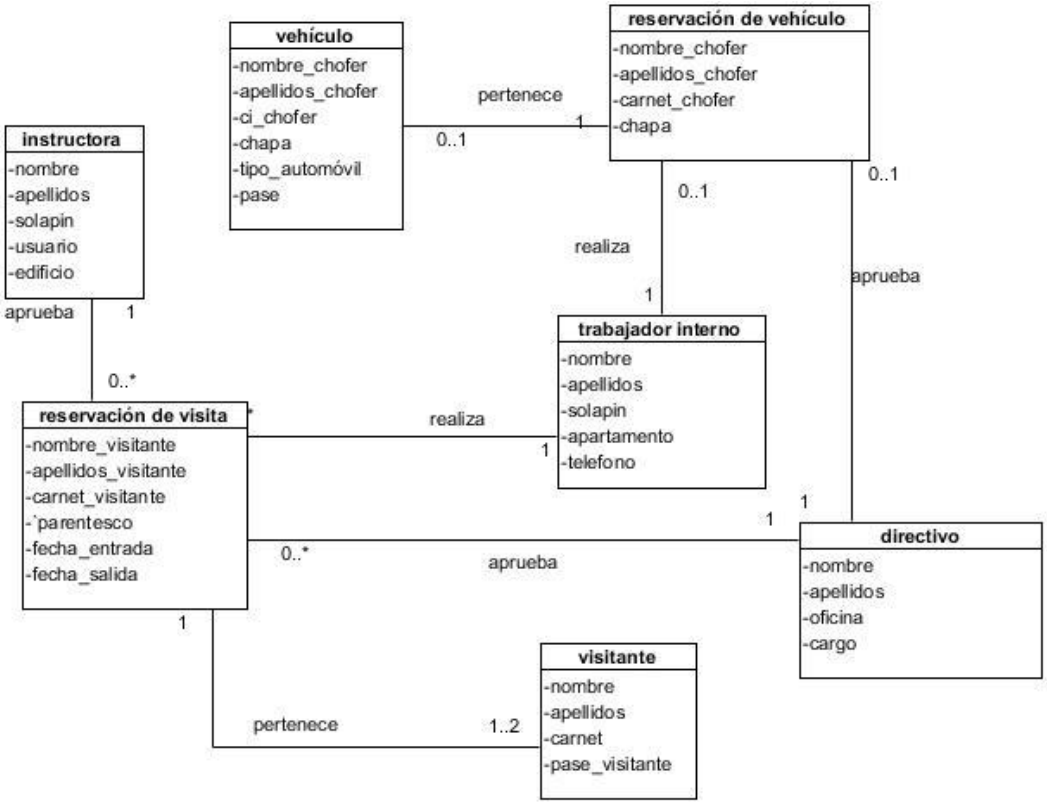


Fig. 3: Modelo conceptual o de dominio

2.3. Definición de los requisitos de software.

El objetivo de la definición de requisitos es obtener una clara comprensión del problema al cual se desea dar respuesta mediante su obtención, extraer las necesidades del usuario y derivar de ellas las funciones que debe realizar el sistema (31). Luego de ser precisados los requisitos con que debe contar el sistema, debe lograrse realizar una explicación de cada uno de ellos, de forma que sea entendible para cualquier tipo de destinatario ya sea clientes y/o usuarios.

2.3.1. Técnicas de captura de requisitos utilizadas.

Para determinar las necesidades que se pretenden responder con el desarrollo del sistema fueron aplicadas las siguientes técnicas de captura de requisitos:

Cuestionarios y Checklists

Esta permite que mediante el conocimiento previo del problema en el que se está trabajando que sea elaborado un documento que facilite el desarrollo de la entrevista. Dicho documento debe contener un conjunto de preguntas cuyas respuestas sean cortas y concretas. Este cuestionario será asistido por el

grupo de personas entrevistadas o simplemente para recoger información en forma independiente de una entrevista (32). Para la aplicación de esta técnica fueron elaboradas un conjunto de preguntas las cuales serán empleadas para la realización de las entrevistas con los diferentes involucrados.

Entrevista

Permite determinar los problemas que tienen en la entidad y determinar el porqué del desarrollo de un sistema. Las estructuradas que interviene responden a cómo y quién, favoreciendo el contacto directo y la validación. Las entrevistas no se deben improvisar, por lo que se debe ir con ella redactada para que tenga éxito (33). Para la realización de la misma se tomaron en cuenta los criterios del Director de residencia, así como los del director de seguridad y protección, trabajadores internos en la universidad e instructoras que participan en el proceso. Mediante la aplicación de las preguntas elaboradas fue posible la obtención de criterios del desarrollo actual del proceso los cuales son empleados como base para el desarrollo de la solución.

Comparación de terminología

Es utilizada como complemento de otras técnicas permitiendo identificar los términos a usar en el desarrollo del sistema. Mediante el uso de esta se pretende lograr que la solución contenga los términos mayormente asociados al funcionamiento del negocio para de esta forma lograr un mayor entendimiento por parte de los usuarios de destino (32). Mediante el uso de esta técnica han sido determinados los diferentes términos a tener en cuenta para el desarrollo de la solución los cuales responderán a parámetros a tener en cuenta en la realización de una reservación así como los procesos que en esta se describen.

Observación del usuario

Permite mediante la observación del futuro usuario determinar las tareas y objetivos que debe satisfacer el sistema para ajustarse a las características de la organización. A partir de la puesta en práctica de esta técnica pueden ser detectados elementos propios de la organización que no fueron tomados en cuenta con la aplicación de la entrevista (34). Esta técnica permitió estructurar con mayor envergadura el funcionamiento actual de los servicios analizados para determinar cada uno de los elementos que debe incluir el sistema. Para la aplicación de esta técnica fue observada la ejecución del servicio para diez trabajadores.

Observación de sistemas semejantes

Permite la determinación de elementos reutilizables a partir de la investigación de las funcionalidades y características que se satisfacen a partir del uso de alguno de estos sistemas los cuales pueden servir de

base para saber determinar los elementos a buscar y obtener una colección de requisitos de partida que permita agilizar la toma de requisitos (34). A partir de los elementos proporcionados por la analista del CENIA para el módulo de Residencia fueron tomadas como referencias las características satisfechas por el sistema desarrollado en este cetro las cuales proporcionaron un punto de partida para la determinación de las que serán incluidas en esta solución.

2.3.2. Requisitos funcionales.

A partir de la obtención de los requisitos funcionales es posible determinar las necesidades que debe satisfacer el sistema para dar respuesta a las necesidades de los clientes. Estos deben especificar de manera particular la forma en que el sistema debe reaccionar a determinadas entradas y pueden declarar además explícitamente el comportamiento inapropiado del sistema (35). Mediante la aplicación de las diferentes técnicas de captura de requisitos empleadas fue obtenida la lista de requisitos funcionales mostrada a continuación:

- RF Mostrar reporte.
- RF Solicitar prórroga.
- RF Adicionar reservación de visitas.
- RF Modificar reservación de visitas.
- RF Eliminar reservación de visitas.
- RF Listar reservación de visitas.
- RF Buscar reservación de visitas.
- RF Adicionar reservación de vehículo.
- RF Modificar reservación de vehículo.
- RF Eliminar reservación de vehículo.
- RF Listar reservación de vehículo.
- RF Buscar reservación de vehículo.
- RF Aprobar reservación de visitas.
- RF Rechazar reservación de visitas.
- RF Aprobar reservación de vehículo.
- RF Rechazar reservación de vehículo.

2.3.3. Descripción de los requisitos funcionales.

Tabla 2: Descripción del requisito funcional: Adicionar reservación de visita.

Precondiciones	El usuario tiene permisos para realizar la acción.
Flujo de eventos	
Flujo básico	
1. Se introducen los datos: Nombre Apellido Parentesco Carnet de Identidad Fecha de entrada Fecha de salida Tipo de pase Solapín	
2. El sistema valida (ver validación 1) los datos introducidos.	
3. Si los datos son correctos el sistema los registra.	
4. El sistema confirma el registro de los datos.	
5. Concluye el requisito.	
Pos-condiciones	
1. Se registró en el sistema un nuevo visitante con estado por definir.	
2. Se muestra un listado actualizado de las solicitudes de visitas.	
Flujos alternativos	
Flujo alternativo 3.a Información errónea	
1. El sistema muestra un cartel señalando el tipo de información errónea.	
2. El usuario solicita nuevamente adicionar.	
3. El usuario introduce los datos.	
4. Volver al paso 2 del flujo básico.	
Pos-condiciones	
1 N/A	
Flujo alternativo 3.b Información incompleta	
1 El sistema muestra un cartel de error.	

2	El usuario solicita nuevamente adicionar.	
3	El usuario corrige los datos.	
4	Volver al paso 2 del flujo básico.	
Pos-condiciones		
1	N/A	
Flujo alternativo *.a El usuario cancela la acción		
1	Concluye el requisito.	
Pos-condiciones		
1	No se registran los datos.	
Validaciones		
1	Se validan los datos según lo establecido.	
Relaciones	**Requisitos Incluidos**	N/A
Conceptos	Reservación de visitas Trabajador interno Visitante	
	Extensiones	N/A
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

2.3.4. Prototipos de interfaz de usuario.

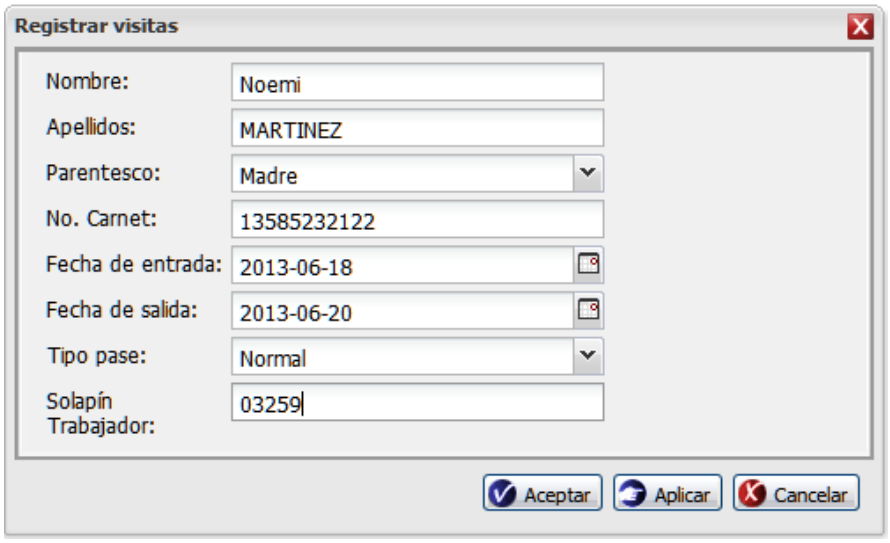


Fig. 4: Prototipo interfaz. Adicionar reservación de visitas.

2.3.5. Técnicas para la validación de los requisitos.

Los requisitos una vez definidos necesitan ser validados, lo cual permite demostrar que la definición de los mismos responde específicamente a la necesidad del usuario. Pocas son las propuestas existentes que ofrecen técnicas para la realización de la validación y algunas de ellas consisten en revisar los modelos obtenidos en la definición de requisitos con el usuario para detectar errores o inconsistencias. Las técnicas de validación de requisitos empleadas son:

Revisiones: está técnica consiste en la lectura y corrección de la completa documentación o modelado de la definición de requisitos. Con ello solamente se puede validar la correcta interpretación de la información transmitida. Más difícil es verificar consistencia de la documentación o información faltante.

Prototipos: algunas propuestas se basan en obtener de la definición de requisitos prototipos que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema.

- Resultados de la aplicación de las técnicas.

Como resultado se pudo revelar los errores u omisiones existentes en los requisitos propuestos. Mediante la revisión de las descripciones correspondientes a los requisitos propuestos se logró que estos estuviesen detallados de forma correcta y completa. Antes del momento de realización de los prototipos interfaz solo existían definidos 14 requisitos y luego de la aplicación de esta técnica quedaron definidos 16 requisitos ya

que fueron incluidos los requisitos referentes a la aprobación por separado de los tipos de reservaciones. Esto permitió la obtención de una versión inicial que responda a las características del negocio.

2.3.6. Requisitos no funcionales.

Los requisitos no funcionales permiten determinar las propiedades que deben cumplir la aplicación y el entorno donde se ejecute para su correcto funcionamiento. Estos se refieren a elementos como la fiabilidad, la respuesta en el tiempo y la capacidad de almacenamiento (36).

Apariencia o interfaz externa.

- La aplicación deberá desarrollarse en un ambiente Web.
- Debe presentar una interfaz sencilla, amigable y de fácil uso, para que pueda ser utilizada sin mucho entrenamiento por los usuarios, teniendo en cuenta que algunos de estos pueden no ser especialistas en informática, sin dejar de poseer un ambiente profesional.

Usabilidad

- El sistema debe ser fácil de utilizar para los usuarios que tengan niveles básicos de computación o no hayan trabajado con la Web con previa anterioridad.
- Las operaciones de la aplicación a informatizar serán lo más parecidas posible a los procesos que se realizan actualmente en la universidad, para así lograr menor tiempo en cuanto a la comprensión y adaptación del sistema.

Rendimiento

- El sistema debe garantizar la rapidez de respuesta ante las solicitudes de los usuarios, además debe llevar a cabo con gran velocidad el procesamiento de la información y el rápido acceso a sus páginas.
- La eficiencia del producto estará determinada en gran medida por el aprovechamiento de los recursos que se disponen en el modelo Cliente/Servidor, y la velocidad de las consultas a la Base de Datos.

Seguridad

- Confidencialidad: Al sistema deberán tener acceso todos los trabajadores internos de la universidad, todos con los permisos correspondientes, por lo que será necesaria una autenticación previa, estando la información a la que estos acceden protegida del acceso no autorizado y la divulgación.
- Integridad: La información manejada por el sistema deberá ser objeto de cuidadosa protección contra la corrupción y estados inconsistentes.

- Disponibilidad: Al sistema deberá ser posible acceder las 24 horas del día para todos los usuarios autorizados.

Portabilidad

- La herramienta desarrollada deberá ser multiplataforma (Linux o Windows)

Software

- Para el servidor debe contar como servidores Web y de bases de datos Apache y PostgreSQL respectivamente.
- Para el cliente debe incluir como navegador web Mozilla Firefox 3.0 o superior.

Hardware

- Para el servidor los requerimientos mínimos deben ser un procesador Dual Core a 3.00 GHz, 1gb de memoria RAM y una capacidad de 40gb de disco duro, además de una tarjeta de red.
- Para el cliente los requerimientos mínimos deben ser un procesador Pentium IV a 800 MHz con 128 Mb de memoria RAM, además de una tarjeta de red.

2.4. Conclusiones parciales.

Entre los principales resultados alcanzados durante el desarrollo del capítulo se resaltan:

- El análisis del negocio facilitó representar la realidad que presenta el proceso de reservación de visitas para un mejor entendimiento del funcionamiento de cada proceso que será informatizado.
- El empleo de las técnicas de captura de requisitos facilitó la elaboración del listado de requisitos funcionales, logrando identificar y especificar los aspectos del proceso de negocio, que definirán la estructura del sistema que se pretende desarrollar como propuesta de solución.
- Fueron obtenidos 16 requisitos funcionales y 14 no funcionales lo que permite adquirir una adecuada comprensión de las necesidades existentes y obtener la información necesaria para el posterior desarrollo de la solución.
- Las técnicas de validación aplicadas a los requisitos dieron una idea al cliente de la estructura de la interfaz y el comportamiento del sistema, siendo emitida la satisfacción del mismo con las funcionalidades que se identificaron y certifica además que estas son las que deben ser implementadas.

CAPÍTULO 3. ANÁLISIS Y DISEÑO DEL SISTEMA

3.1. Introducción

Lograr transformar los requisitos funcionales en el diseño del futuro sistema para una posterior implementación a partir de la arquitectura definida, será el punto de partida del presente capítulo. Se tendrán en cuenta los diferentes patrones y se obtendrán un conjunto de artefactos que serán de gran valor para la fase de construcción como son: el diagrama de componentes, diagrama de clases del diseño, diagrama de secuencia, el modelo de datos.

3.2. Arquitectura de la solución.

Mediante el uso del marco de trabajo Sauxe para el desarrollo del sistema es necesaria la adopción del estilo arquitectónico n-capas que constituye una organización tal que cada capa proporciona servicios a la capa superior y se sirve de las prestaciones que le brinda la inferior. La división del sistema en capas facilita el diseño modular, en la que cada capa encapsula un aspecto concreto del sistema y permite además la construcción de sistemas débilmente acoplados, lo que significa que si se minimizan las dependencias entre capas, resulta más fácil sustituir la implementación de una capa sin afectar al resto del sistema. (12)

Las capas que estarán representadas en el desarrollo del sistema:

- Capa de Presentación: En esta capa se emplea las facilidades que brinda el Marco de Trabajo ExtJS para la construcción de interfaces amigables a la vista de los usuarios.
- Capa de Control o Negocio: En esta capa se emplea el patrón de arquitectura Modelo Vista Controlador (MVC). De forma vertical al modelo descrito hasta este momento, estarán los aspectos fundamentales del sistema así como el encargado del tratamiento de la seguridad a nivel de aplicación Web.
- Capa de Acceso a Datos: En esta capa estará presente el ORM (Object Relational Mapping) Doctrine, como Marco de Trabajo de persistencia para la comunicación con el servidor de datos mediante el protocolo PDO (PHP Data Objects) (12).

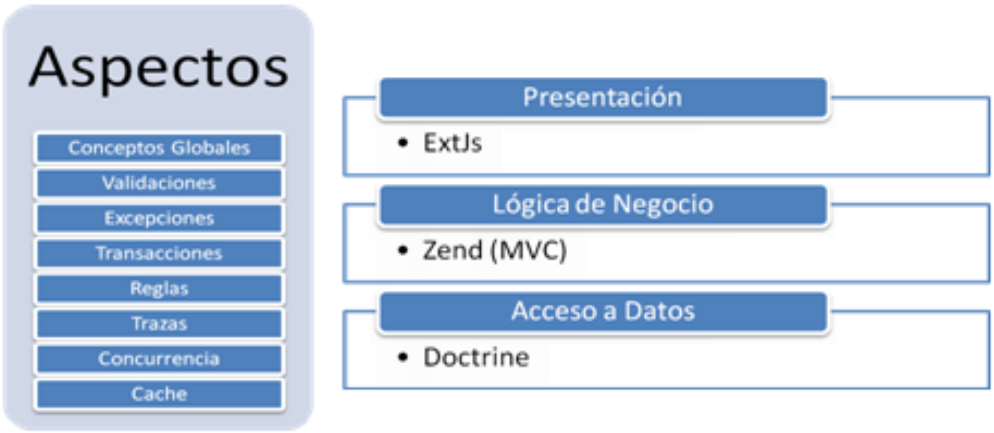


Fig. 5: Estructura arquitectónica del marco de trabajo Sauxe.

- Fuente: (12)

3.2.1. Patrón arquitectónico aplicado

Modelo Vista Controlador

Este se encarga de separar los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres elementos distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, el controlador es el responsable de recibir los eventos de entrada desde la vista.

Modelo

- Accede a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- Define las reglas de negocio (la funcionalidad del sistema).
- Lleva un registro de las vistas y controladores del sistema.

Controlador

- Recibe los eventos de entrada.
- Contiene reglas de gestión de eventos.

Vistas

- Recibe datos del modelo y los muestra al usuario.
- Tiene un registro de su controlador asociado (normalmente porque además lo instancia) (37).

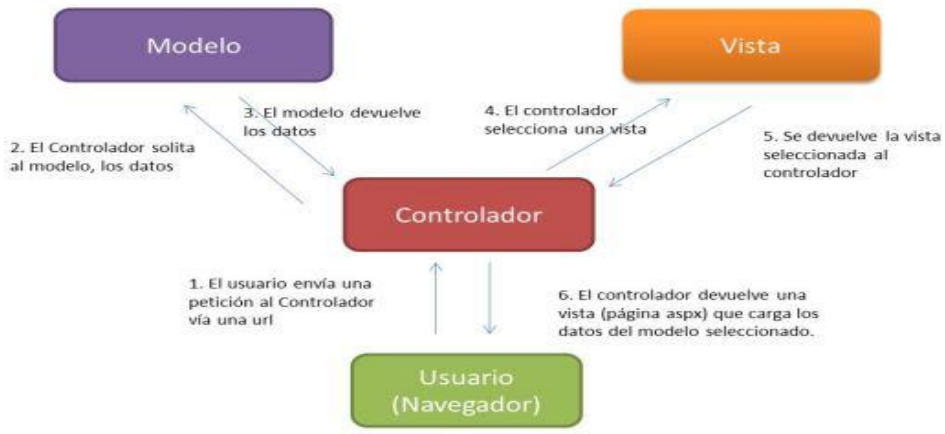


Fig. 6: Estructura de intercambio de información en el patrón MVC.
Fuente: (38)

3.2.2. Diseño de la solución en términos de componentes

Las funcionalidades capturadas y modeladas en las disciplinas de negocio y requerimientos quedan expresadas o contenidas en al menos un componente, y las distintas interacciones entre estos componentes originan funcionalmente la existencia de subsistemas. De esta forma el sistema queda constituido por un conjunto de componentes que responden a las funcionalidades, un grupo de interacciones entre estos componentes respondiendo a las distintas integraciones y dependencias originadas en el negocio, estos componentes están agrupados en una unidad mayor denominada subsistemas que responden a las áreas de procesos más generales identificadas en el negocio. Cada funcionalidad por su parte, puede abarcar o no, un conjunto de requisitos funcionales, los cuales convergen en un solo controlador de eventos. A continuación se presenta el Mapa de componentes que responde al funcionamiento del Sistema de reservación de visitas a trabajadores internos, el cual está estructurado de acuerdo a los diferentes niveles de empaquetamiento: subsistema, componente y funcionalidad.

Especificación de los niveles de empaquetamiento subsistema y componente:

Subsistema

Sistema de Reservación: Permite el registro, seguimiento y control de las reservaciones de visitas a trabajadores para todos los niveles organizacionales. Dicho subsistema contiene el componente:

- Reservación: Permite la gestión de los elementos de los servicios de reservación de visitas y de acceso de vehículos. Que contiene la funcionalidad

- Reservación de entrada: la cual responde a los procesos de gestión de las reservaciones de entradas de visitas y vehículos.

Este componente se relaciona mediante una interfaz de comunicación denominada SeguridadProxyService con el componente de seguridad, del cual serán consumidos los servicios de usuarios y de gestión de roles. A partir de su relación con el servicio de estructura mediante la interfaz de comunicación EstructuraService permitirá que sea cargada la aplicación con la estructura definida.

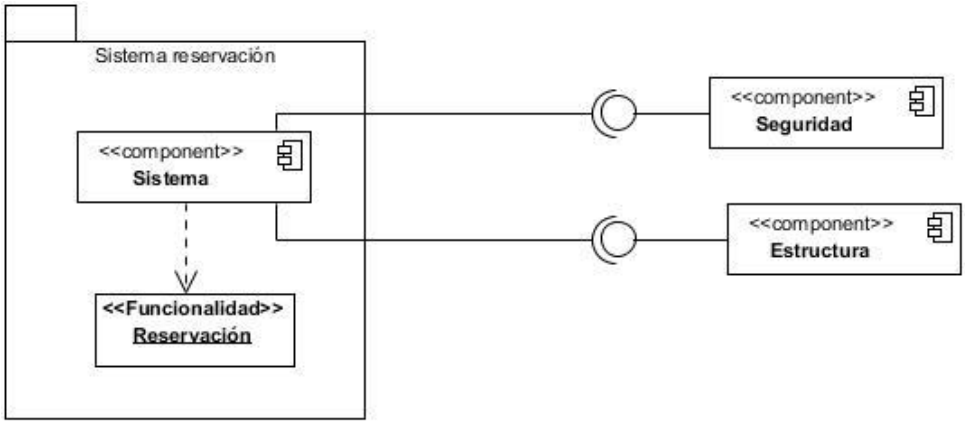


Fig. 7: Diagrama de componentes.

3.3. Patrones de diseño

GRASP: (General Responsability Assignment Software Patterns) permiten la asignación responsabilidades a los objetos en sentido general (39).

Experto

Lo que plantea este patrón es que se debe asignar una responsabilidad al experto en información, es decir, la clase que tiene la información necesaria para cumplir con dicha responsabilidad. El experto se emplea más que cualquier otro patrón en la asignación de responsabilidades, es un principio usado continuamente en el diseño.

Dicho patrón es evidenciado en la definición de las clases de acuerdo a la información manejada dentro del sistema, como por ejemplo las clases que hacen referencia a las bases de datos. Ejemplo de ello es la clase Datreservacion, la cual contiene toda la información referente a las reservaciones de visitas. Dicha clase será la responsable de efectuar las operaciones que respectan a las funciones para la labor de cada una de las funcionalidades.

Creador

El creador guía la asignación de responsabilidades relacionadas a la creación de objetos, una tarea muy común en sistemas orientados a objetos. El intento básico del patrón creador es encontrar un creador que necesite estar conectado al objeto creado en un evento en particular (39).

Este patrón es adaptable a la clase GestreservacionController, quienes son las encargadas de crear los objetos de tipo Datreservacion, para permitir el acceso a la información almacenada a nivel de datos.

Controlador

Asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. Un error muy común es asignarle demasiada responsabilidad y alto nivel de acoplamiento con el resto de los componentes del sistema (40).

Las clases controladoras definidas (GestreservacionControllers) son un ejemplo de la aplicación de este patrón, las mismas tendrán a su cargo la responsabilidad de manejar los eventos dentro del sistema.

Bajo acoplamiento

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases. Este patrón da soporte a una mínima dependencia y a un aumento de la reutilización; una clase con bajo acoplamiento no depende de muchas otras clases para realizar sus tareas.

Alta cohesión

La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realizan un trabajo enorme. Fomenta la reutilización, mejorando la claridad y facilidad del diseño.

3.3.1. Patrones GoF

GangofFour (“pandilla de los cuatro”) estos pueden ser clasificados según su propósito de aplicación en el sistema entre estas se encuentra:

- **Creación:** tratan la creación de instancias.
- **Estructura:** tratan la relación entre clases, la combinación de clases y la formación de estructuras de mayor complejidad.
- **Comportamiento:** tratan la interacción y cooperación entre clases (41).

Fachada

Es un patrón estructural que provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un sistema. Desde el punto de vista de los usuarios que van a consumir el sistemas estos no

necesitan tener conocimiento del negocio interno del componente porque de este modo es más sencillo de utilizar. En cuanto a lo que del sistema se refiere, ayuda a organizar un sistema en capas, a controlar o eliminar las dependencias complejas o circulares entre objetos y permite hacer cambios en los componentes sin afectar a los clientes.

La clase Zend_View, encargada de asignarle responsabilidades a objetos de manera dinámica y configurarlos con nuevos atributos implementa dicho patrón el cual proporciona una interfaz unificada para un conjunto de interfaces de un sistema. Define una interfaz de alto nivel que hace que el sistema sea más fácil de utilizar.

Cadena de responsabilidad

Es un patrón de comportamiento que evita acoplar el emisor de una petición a su receptor, al dar a más de un objeto la posibilidad de responder a la petición. Crea una cadena con los objetos receptores y pasa la petición a través de la cadena hasta que esta sea tratada por algún objeto. Su uso en el diseño es enfocado principalmente para validar la producción de errores al insertar elementos en la base de datos, el cual es captado por las capas superiores, reenviando la excepción hasta la capa de aplicación donde dicho error se traduce al lenguaje del usuario.

Este patrón se refleja en las clases controladoras y las del modelo donde se establece la cadena a seguir para las peticiones de la clase controladora al modelo y del modelo a la entidad. Ejemplos: GestreservacionControllers y GestreservacionModel.

Instancia única

Es un patrón creacional que garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia (41).

Este proporciona una instancia única del controlador frontal disponible para lograr una vía de entrada única a las solicitudes.

3.3.2. Diagrama de clases

El diagrama de clases del diseño está compuesto por métodos, atributos y clases donde se definen los tipos de relación que existirá entre ellas que pueden ser de uso, agregación, composición y herencia. El diagrama de clases para la gestión de reservaciones se realizó teniendo en cuenta las características del marco de trabajo Sauxe, el cuál presenta un estilo arquitectónico n-capas asociado al patrón arquitectónico MVC. A continuación se presenta el diagrama de clases de diseño correspondiente a gestionar solicitud de visitas, el resto de los mismos pueden ser consultados en expediente de la aplicación.

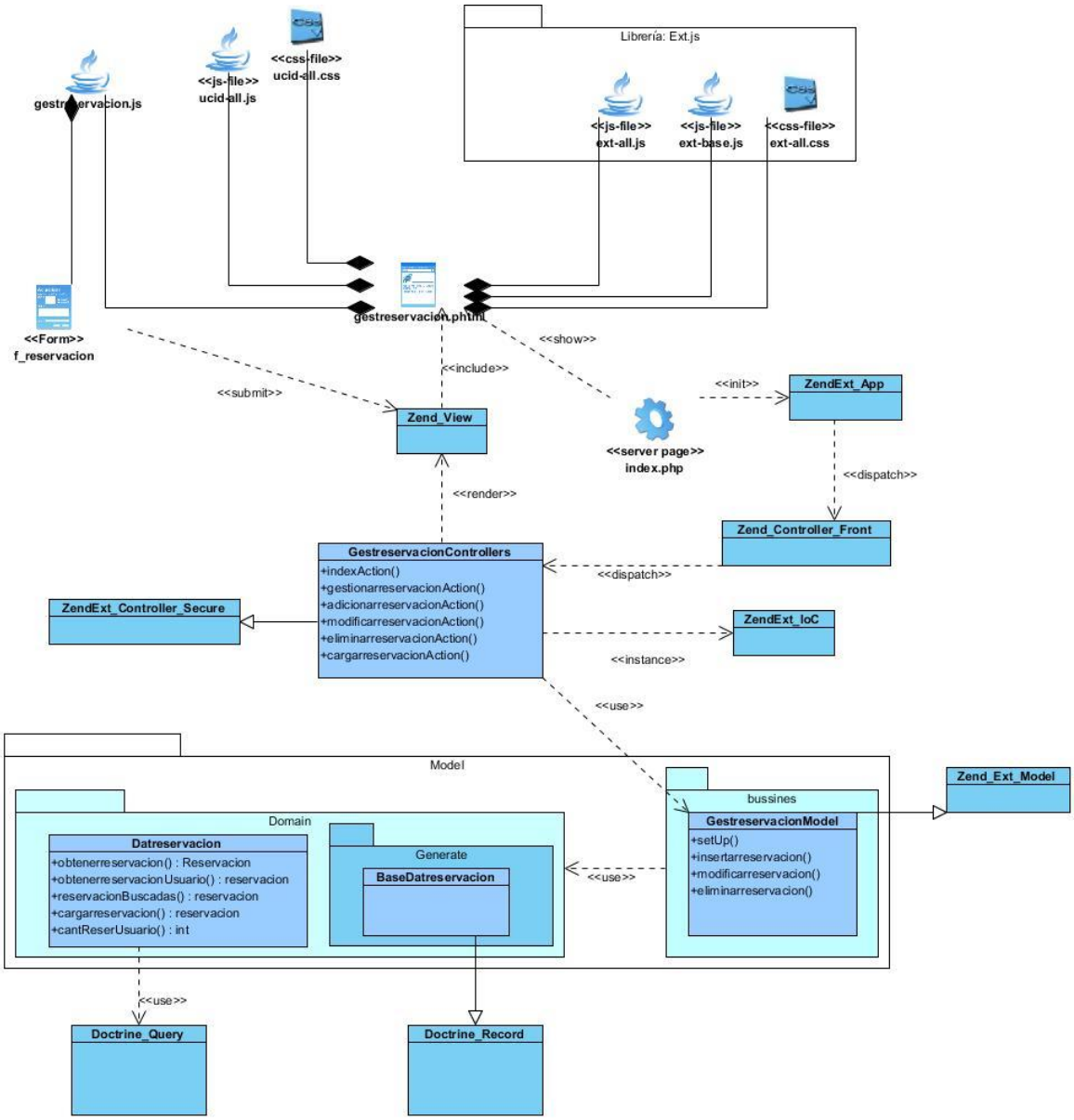


Fig. 8: Diagrama de clases del diseño. Gestionar reservación de visitas.

Tabla. 3: Descripción de las clases del diseño.

Clase	Descripción
gestreservacion.js	Clase donde se determinan los elementos que van a ser registrados de las solicitudes.

Ext-base.js	Librería JavaScript utilizada para el desarrollo de las interfaces de usuario.
Ext-all.js	Librería JavaScript utilizada para el desarrollo de las interfaces de usuario.
Ext-all.css	Librería CSS utilizada para el desarrollo de las interfaces de usuario.
Ucid-all.js	Clases utilizadas para la configuración de los mensajes.
Ucid-all.css	Clases utilizadas para la configuración de los mensajes
<<Form>> f_reservacion	Formulario utilizado para el registro de la solicitud.
gestreservacion.phtml	Es el HTML donde se incluyen todos los java script para la representación del componente.
Index.php	Es la clase servidora que se encarga de toda la configuración de la aplicación, re direccionar y dar seguridad.
Zend_View	Provee una plantilla basada en PHP, proporciona un ayudante que permite la creación de códigos de visualización.
ZendExt_Controller_Secure	Clase de donde heredan todas las clases controladoras.
Zend_Controller_Front	Inicializa el entorno de la solicitud, enruta la solicitud entrante, y luego envía las acciones descubiertas; le agrega las respuestas y los devuelve cuando el proceso se haya completado.
ZendExt_App	Clase que proporciona ayuda en la gestión del Modelo-Vista-Controlador.

Zend_Ext_Model	Obtiene las conexiones activas en el sistema cuando se trata de acceder a una clase determinada del modelo.
GestreservacionController.php	Es la clase que gestiona la lógica de negocio y se encarga de capturar los parámetros que le son enviados desde la interfaz.
GestreservacionModel.php	Es la clase encargada de guardar los elementos registrados.
Datreservacion.php	Es la clase encargada de generar las solicitudes para la realización de determinada funcionalidad.
BaseDatreservacion.php	Es la clase utilizada para la definición de los elementos de la base de datos.
ZendExt_IoC	Clase que contiene el servicio web utilizado para buscar usuarios.

3.3.3. Diagramas de secuencia

Los diagramas de secuencia permiten mostrar la interacción existente entre un conjunto de objetos en una aplicación a través del tiempo. Los cuales contiene detalles de implementación del escenario analizado, son modelados para cada requisito funcional e incluyendo los objetos y clases que se usan su implementar, además de los mensajes intercambiados entre los objetos. A continuación se muestra un diagrama de secuencia perteneciente al requisito funcional Adicionar reservación de visitas. Para consultar el resto de los diagramas remitirse al expediente asociado ha dicho sistema.

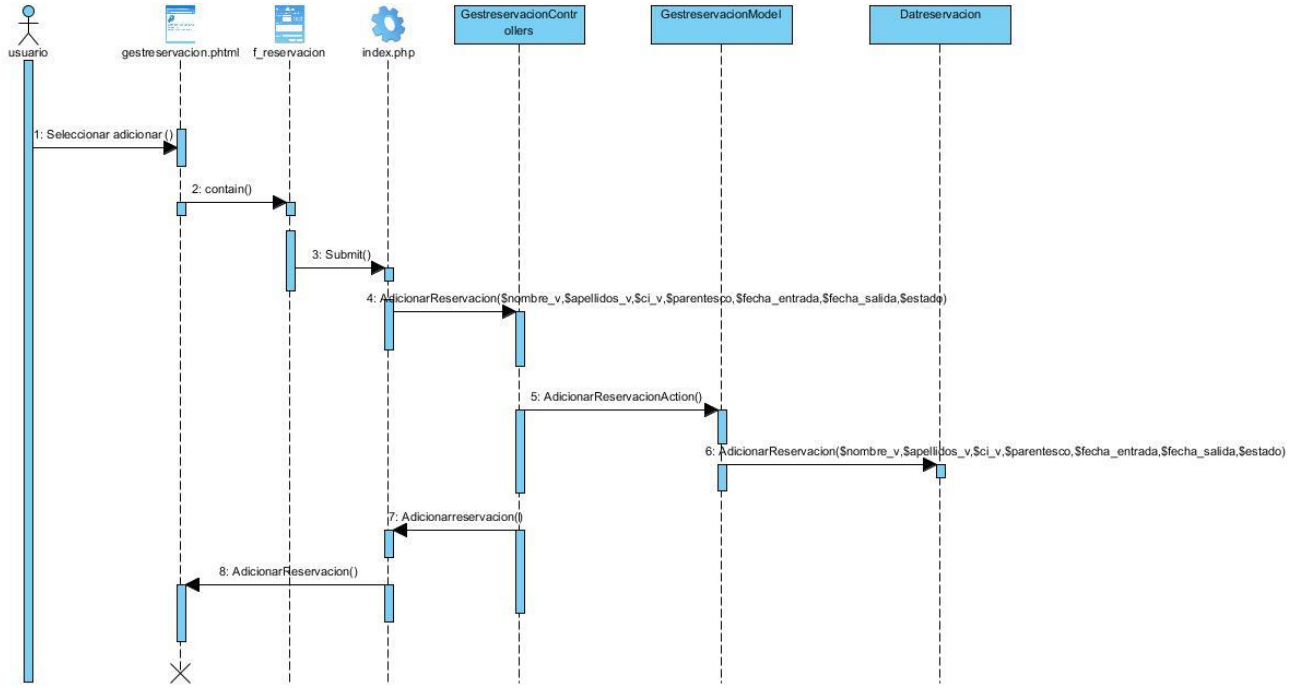


Fig. 9: Diagrama de secuencia. Adicionar reservación de visitas.

3.3.1. Modelo de datos

El modelo de datos permite describir las estructuras de los datos de la base y la forma en que se relacionan, incluye además las restricciones de integridad lo cual permite definir los elementos de la realidad que intervienen en un problema dado y la forma en que se relacionan esos elementos entre sí (42). La estructura física de la información que será registrada en el sistema quedara estructurada de la siguiente manera.

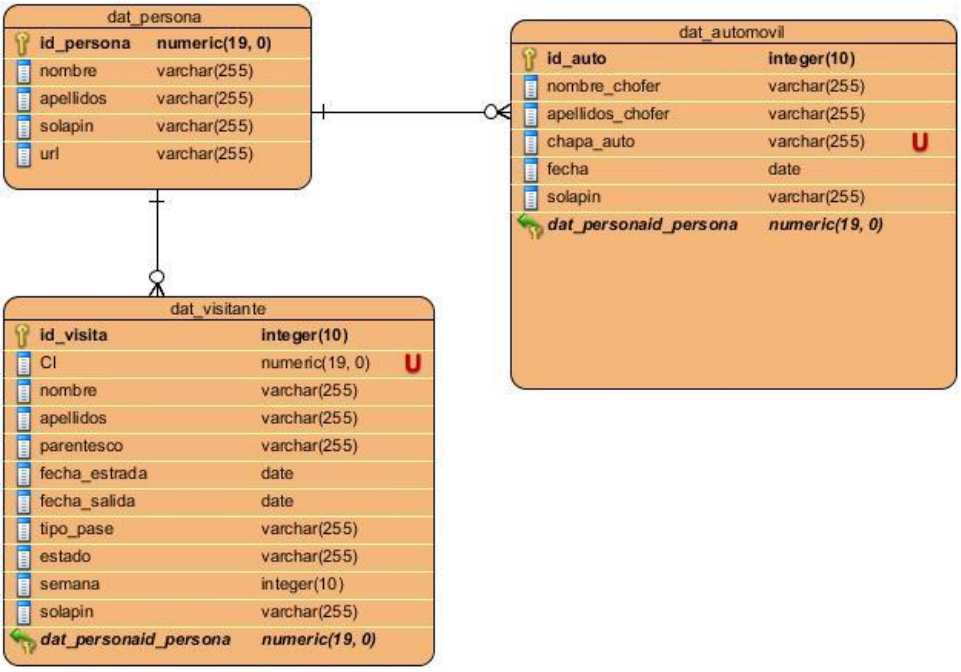


Fig. 10: Modelo físico de datos.

3.3.2. Descripción de las tablas utilizadas

Se muestra la descripción de las tablas utilizadas. Estas se crearon debido a la importancia de almacenar la información para su posterior análisis.

Tabla 4: Diccionario de datos tabla: dat_persona.

Nombre: dat_persona		
Atributo	**Tipo**	**Descripción**
ld_persona	integer	Llave primaria de la tabla.
nombre	varchar	Elemento que muestra el nombre de la persona que realizo la reservación.
apellidos	varchar	Elemento que muestra el apellido de la persona que realizo la reservación.
solapín	varchar	Elemento que muestra que pertenece a la institución como trabajador interno.
Tablas relacionadas	**Descripción de la relación**	
dat_automovil	Uno a muchos	
dat_visitante	Uno a muchos	

Descripción de la tabla: almacena los datos correspondientes a los roles de los usuarios.

Tabla 5: Diccionario de datos tabla: dat_visitante.

Nombre: dat_visitante		
Atributo	**Tipo**	**Descripción**
Id_visita	integer	Llave primaria de la tabla.
dat_personaid_persona	integer	Llave foránea de la tabla que indica la relación existente entre persona y el visitante
CI	numeric	Identificador del visitante en el sistema
nombre	varchar	Nombre de visitante
apellidos	varchar	Apellidos del visitante
Parentesco	varchar	Relación del visitante con el trabajador
fecha_entrada	date	Fecha de entrada del visitante
fecha_salida	date	Fecha de salida del visitante
tipo_pase	varchar	Forma de solicitud del pase (normal o especial)
estado	varchar	Indica el estado de la solicitud (solicitado, aceptado o rechazado)
semana	integer	Número de la semana en que se realizó la solicitud en el año.
usuario	varchar	Usuario del trabajador que realizó la solicitud de pase.
Tablas relacionadas	**Descripción de la relación**	
dat_persona	Uno a muchos	
Descripción de la tabla: almacena los datos correspondientes de los visitantes.		

Tabla 6:Diccionario de datos tabla: dat_automovil.

Nombre: dat_automovil		
Atributo	**Tipo**	**Descripción**
Id_auto	integer	Identificador de la tabla

nombre_chofer	varchar	Nombre del chofer del automóvil
apellidos_chofer	varchar	Apellidos del chofer del automóvil
chapa_auto	varchar	Indica la chapa del automóvil
estado	varchar	Indica el estado de la solicitud (registrada, aceptada o rechazada)
fecha	date	Indica la fecha en que debe entrar y salir el automóvil
usuario	varchar	Usuario del trabajador que realizo la solicitud
dat_personaid_persona	Integer	Identificador de la relación con la tabla dat_persona.
Tablas relacionadas	**Descripción de la relación**	
dat_rol	uno a muchos	
Descripción de la tabla: almacena los datos correspondientes de los autos de los visitantes.		

3.4. Conclusiones parciales.

En este capítulo fueron expuestos los artefactos generados durante el análisis y diseño de la solución propuesta para el sistema de reservación de visitas a trabajadores internos en la UCI.

- Mediante el uso correcto de los patrones de diseño en la generación de los artefactos necesarios para el desarrollo, se obtuvo un diseño que posibilita la obtención de un punto de partida a las actividades de implementación, con el fin de satisfacer las necesidades del cliente.
- El diseño de la solución en cuanto a la distribución por componentes posibilitó establecer las bases para el desarrollo de la solución. Por consiguiente, una vez precisadas las funcionalidades y la estructura de la solución es posible el comienzo a la implementación de la solución.

CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA

4.1. Introducción

En este capítulo a partir del entendimiento de las diferentes funcionalidades a desarrollar en dicho sistema se procede a la implementación de las mismas. Por otra parte se mostrará el diagrama de despliegue, además de la puesta en práctica del proceso de validación y prueba mediante las técnicas de validación de requisitos, la métrica de diseño y las pruebas de caja negra a las interfaces del sistema.

4.2. Modelo de implementación

El modelo de implementación comprende un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos (43).

4.2.1. Estructura física del sistema

El marco de trabajo Sauxe contiene una estructura física la cual debe ser representada por sus elementos asociados de forma tal que permite la organización y estandarización de manera que facilita la claridad durante el desarrollo.

La carpeta denominada *apps* es la que contiene la lógica del negocio. En esta serán almacenados los controladores y el modelo de cada uno de los componentes correspondientes a los subsistemas.

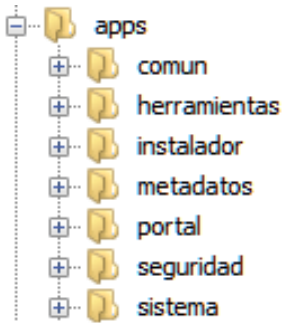


Fig. 1: Contenido de la carpeta de aplicación apps.

comun: la cual contiene la carpeta recurso quien tiene asociado consigo los ficheros validation, ioc, exception. Dichos ficheros contiene cada uno de ellos su papel determinante en cuanto al funcionamiento del sistema.

- validation**:** chequea las precondiciones antes de ejecutar una determinada función en el servidor según el tipo de parámetros, la acción y el usuario que la realice.
- ioc**:** publica los servicios que brindan cada uno de los componentes en cuanto a nombre de clases, funciones y tipo de resultado.
- exception**:** define el tipo, idioma y la descripción del mensaje a mostrar cuando se lance una excepción en el servidor.

A partir del uso del marco de trabajo Sauxe el componente asociado al sistema contendrá la estructura que se muestra a continuación:

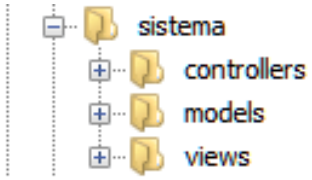


Fig. 12: Contenido del componente sistema de reservación dentro de la carpeta apps.

Donde en la carpeta *controllers* estarán agrupadas cada una de las clases controladoras responsables de la gestión de las funcionalidades del sistema.

La carpeta *models* representa la agrupación de las carpetas mostradas a continuación:

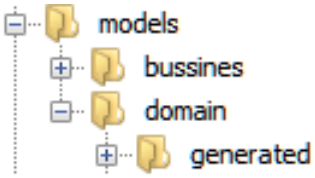


Fig.13: Contenido de la carpeta models.

- bussines: contiene las clases necesarias para acceder a los datos que persisten en la base de datos.
- domain: contiene las clases generadas por el ORM Doctrine Generator a partir de cada una de las tablas existentes en la base de datos.

Las clases pertenecientes a la carpeta domain estarán relacionadas con las clases contenidas en la carpeta generated las cuales han sido resultado igualmente del uso de Doctrine Generator.

La carpeta *views* contiene las carpetas *idioma* y *scripts*, que se encargan de contener el idioma en que se va a mostrar la aplicación y las páginas clientes respectivamente.

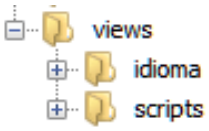


Fig. 14: Contenido de la carpeta views.

En el mismo nivel en que se encuentra contenida la carpeta *apps* se encuentra asociada también la carpeta *web* la cual contiene las vistas de los subsistemas y componentes.

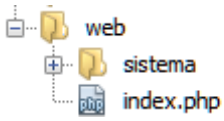


Fig. 14: Carpeta de diseño correspondiente al sistema.

- index: fichero que incluye la dirección del archivo de configuración y a través de este inicializa la aplicación para que sea posible abrir en esta un conjunto de componentes necesarios para su funcionamiento.

Dentro de la carpeta perteneciente al sistema se encontrara asociada la carpeta *views* la cual trae asociada consigo un grupo de carpetas empleadas en las vista que se le muestra al usuario del sistema.

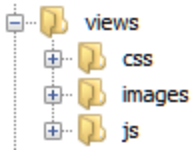


Fig. 15: Contenido dentro de la carpeta views.

- js: contiene las clases Java Script que posibilitan que el usuario interactúe con el sistema y obtenga los resultados necesarios. Está compuesta por carpetas con los nombres de las funcionalidades del componente asociadas a las clases controladoras.

4.2.1. Estándares de codificación

Los estándares de codificación son reglas que se siguen para la escritura del código fuente para lograr un estilo de programación homogéneo en un proyecto. Permite que todos los participantes lo puedan entender en menos tiempo y que el código en consecuencia sea guiado por una misma norma. De tal manera es más

sencillo el proceso de comprensión por parte de otros programadores del código que posee determinado programa (como identificar las variables, las funciones o métodos, etc.) (44).

Notación Húngara: Esta convención se basa en definir prefijos para cada tipo de datos y según el ámbito de las variables. También es conocida como notación REDDICK (por el nombre de su creador).La idea de esta notación es la de dar mayor información al nombre de la variable, método o función definiendo en ella un prefijo que identifique su tipo de dato y ámbito (44).

Notación Pascal Casing: Es como la notación húngara pero sin prefijos. En este caso, los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula.

Notación Camel Casing: Es parecido al Pascal-Casing con la excepción que la letra inicial del identificador debe estar en minúscula (45).

4.2.2. Nomenclatura de las clases

Los nombres de las clases comenzarán con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación Pascal Casing. El mismo estará dado según la función que cumplan, es decir, se le agregará al nombre original un prefijo o sufijo que describa el tipo de clase que es la misma. A continuación se presentan los prefijos y sufijos determinados para cada tipo de clase.

```php
<?php

class GestreservacionController extends ZendExt_Controller_Secure {

    function init() {
        parent::init();
    }

    function gestreservacionAction() {
        $this->render();
    }
```

Fig.16: Ejemplo de la aplicación de la notación Pascal Casing.

4.2.3. Nomenclatura de las funciones

Los nombres a emplear para las funciones se escribirán con minúscula, en caso de que sea un nombre compuesto se empleará notación Camel Casing.

En el caso particular de las funciones de las clases controladoras se les adiciona el sufijo “Action”.

```
function adicionarreservacionAction() {

    $funcion = new Visita();
     $funcion->nombre = $this->_request->getPost('nombre');
    $funcion->apellido = $this->_request->getPost('apellidos');
    $funcion->ci = $this->_request->getPost('ci');
```

Fig. 17: Ejemplo de la aplicación de la notación Camel Casing.

Nomenclatura de las variables

Los nombres a emplear para las variables se escribirán con la notación húngara, en caso de que sea un nombre compuesto se empleará notación Camel Casing.

El nombre de las variables estará dado según la notación Húngara, es decir se le agregará al nombre original un prefijo que describa el tipo de dato. A continuación se listan los prefijos a utilizar para cada tipo de dato.

- Arreglos: “arr”
- Objetos: “obj”
- Enteros: “int”
- Cadena: “str”
- Float: “flt”
- Boolean: “bool”

```
$arrayreservaciones = array();
$semana = array();
$arrayreservaciones = Visita::obtenerreservacion($funcion->ci)->toArray();

foreach ($arrayreservaciones as $reservaciones) {
    //print_r($reservaciones);die;
    if ($reservaciones['semana'] == $funcion->semana) {
        if ($reservaciones['ci'] == $funcion->ci) {
            $no=TRUE;
```

Fig. 11: Ejemplo de la aplicación de la notación Húngara.

Nomenclatura de los comentarios

Los comentarios deben ser lo bastante claros y precisos de forma tal que se entienda el propósito de lo que se está desarrollando. Antes de la declaración de una clase o método se escribirá una breve descripción donde se explique el propósito del mismo. Este comentario tendrá la siguiente estructura:

- En las clases

```
/*
 *Sistema de reservación de visitas.
  * @package
 * @copyright ERP Cuba
 * @author Jenny Martínez
 * @version 1.0-0
 */
```

Fig. 18: Nomenclatura de comentario en las clases.

- En los métodos

```
    /**
* Nombre de la función *
* Descripción *
* @author * (en caso de que no sea el autor de la clase)
* @param *(los parámetros que se le pasan a la función con su descripción)
* @throws *(en caso de que dispare una excepción)
* @return *(se pone lo que devuelve la función y un comentario)
*/
```

Fig. 19: Nomenclatura de comentario en los métodos.

4.3. Métricas de diseño

Las métricas del software permiten medir de forma cuantitativa la calidad de los atributos internos del producto, esto permite al ingeniero evaluar la calidad durante el desarrollo del sistema. (46)

Es de suma importancia la aplicación de las métricas básicas a tener en cuenta para el estudio de la calidad del diseño, que representan el impacto que tendrá la solución propuesta en cuanto al mantenimiento y comprensión de la misma, lo cual permitirá su mejoramiento con éxito. Los atributos de calidad que se tienen en cuenta son:

- **Cantidad de pruebas:** es el grado de esfuerzo para realizar las pruebas de calidad del producto diseñado.
- **Reutilización:** se refiere a la posibilidad de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- **Complejidad del mantenimiento:** es el esfuerzo necesario a realizar para desarrollar una mejora o una rectificación de algún error de un diseño de software.

- **Acoplamiento:** es el grado de dependencia o interconexión de una clase o estructura de clase, con otras (47).

La métrica escogida para determinar la calidad del diseño del sistema de reservaciones de visitas a trabajadores internos en la UCI y su relación con los elementos de calidad es la siguiente:

Relaciones entre Clases (RC): Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

Tabla 6: Atributos de calidad evaluados por la métrica RC.

Atributo de calidad	**Modo en que lo afecta**
Cantidad de pruebas	Aumento del RC provoca aumento de la cantidad de pruebas de unidad necesarias para probar una clase.
Reutilización	Aumento del RC provoca disminución en el grado de reutilización de la clase.
Complejidad del mantenimiento	Aumento del RC provoca aumento de la complejidad del mantenimiento de la clase.
Acoplamiento	Aumento del RC provoca aumento del Acoplamiento de la clase.

Se definieron los siguientes criterios y categorías de evaluación para los atributos de calidad anteriores:

Tabla 7: Criterios de evaluación para la métrica RC.

Atributos	**Categoría**	**Criterio**
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad de Mantenimiento	Baja	<= Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	> 2* Promedio.
Reutilización	Baja	>2* Promedio.
	Media	Entre Promedio y 2* Promedio.
	Alta	<= Promedio.

Cantidad de Pruebas	Baja	<= Promedio.
	Media	Entre Promedio y 2* Promedio.
	Alta	> 2* Promedio.

4.3.1. Resultados obtenidos de la aplicación de la métrica RC.

Tabla 8: Instrumento de evaluación de la métrica RC.

Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad de Mantenimiento	Reutilización	Cantidad de Pruebas
Visitas	1	Bajo	Baja	Alta	Baja
GestprincipalControllers	1	Bajo	Baja	Alta	Baja
VisitasModel	1	Bajo	Baja	Alta	Baja
GestveiculosControllers	1	Bajo	Baja	Alta	Baja
VehiculoModel	1	Bajo	Baja	Alta	Baja
Vehiculo	1	Bajo	Baja	Alta	Baja
Gestprincipal1Controllers	1	Bajo	Baja	Alta	Baja
PersonaModel	1	Bajo	Baja	Alta	Baja
GestreservacionControllers	2	Medio	Media	Media	Media
Persona	1	Bajo	Baja	Alta	Baja

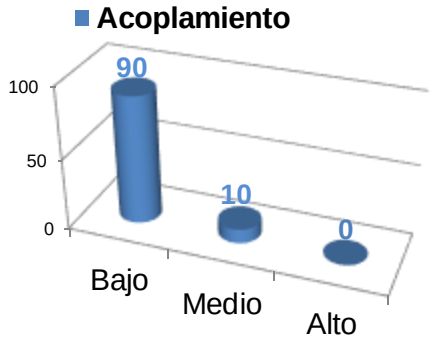


Fig. 19: Resultados de la evaluación de la métrica RC para el atributo Acoplamiento.

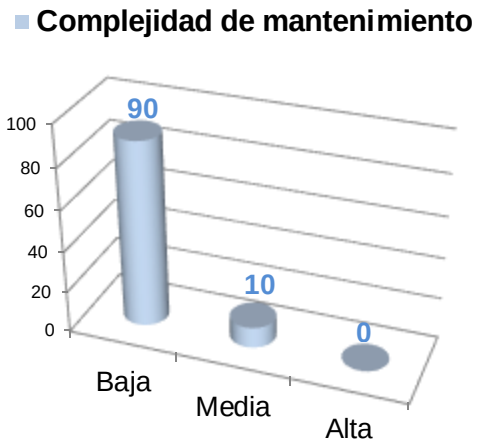


Fig. 20: Resultados de la evaluación de la métrica RC para el atributo Complejidad de mantenimiento.

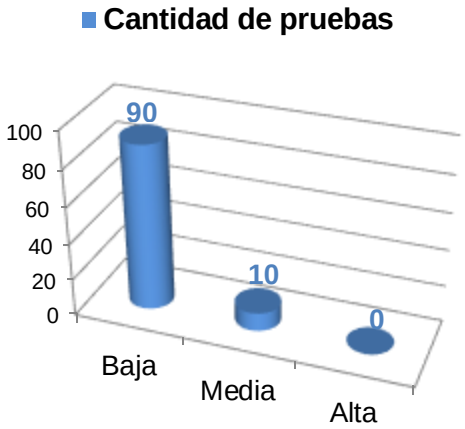


Fig. 21: Resultados de la evaluación de la métrica RC para el atributo Cantidad de pruebas.

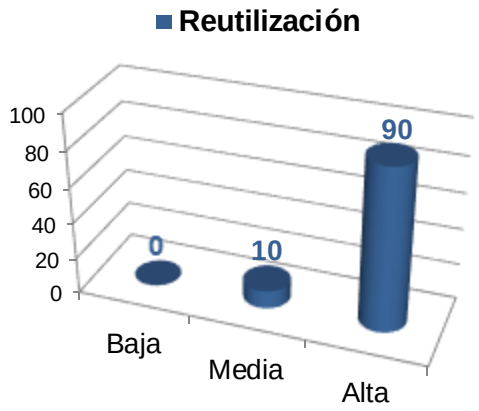


Fig. 22: Resultados de la evaluación de la métrica RC para el atributo Reutilización.

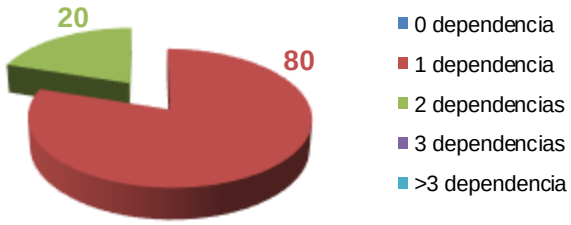


Fig. 23: Representación de las asociaciones de uso por cantidad de clases.

Luego de aplicarse la métrica de diseño RC se obtuvieron resultados que permiten evaluar el diseño propuesto de calidad aceptable teniendo en cuenta que un 80 de las clases (9 clases) utilizadas en el sistema poseen 1 dependencia y el otro 20% restante (1 clase) presenta 2 dependencias.

Los atributos de calidad se encuentran en un nivel satisfactorio; ya que los atributos Complejidad de Mantenimiento, la Cantidad de Pruebas y la Reutilización se comportan favorablemente para un 90% de las clases.

4.4. Pruebas.

Las pruebas constituyen una etapa imprescindible durante el proceso de desarrollo del sistema, estas permiten determinar y corregir al máximo los errores antes de la entrega de la aplicación. El éxito de las mismas puede mejorar la idea de calidad del usuario final y lograr su satisfacción. El objetivo principal de las pruebas es asegurar que el software cumpla con las especificaciones requeridas y eliminar los posibles defectos que este pudiera tener. Las pruebas persiguen como propósitos realizar la validación del sistema desarrollado.

4.4.1. Métodos de Prueba

Existen dos enfoques alternativos descritos como: Caja Negra las cuales comprueban las funcionalidades sin tener en cuenta la estructura interna y de Caja Blanca que comprueban los componentes internos (48).

1. Pruebas de caja negra o funcional

Esta prueba se centra en el estudio de la reacción del sistema con respecto a la entrada de datos y la respuesta brindada a partir de esta información. Para la realización de las mismas son aplicados un conjunto de casos de prueba los cuales señalan los posibles valores de entrada así como de salida para la aplicación de esta prueba. La misma se limita a brindar solo datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el sistema internamente, es decir, solo trabaja sobre su interfaz externa (49).

Este tipo de pruebas permite:

- Verificar que todos los requisitos se han implementado correctamente.
- Funciones incorrectas o ausentes.
- Errores de interfaz.

Entre las técnicas que incluye se encuentran:

- **Partición de equivalencia:** divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- **Análisis de valores límites:** prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- **Grafos de causa-efecto:** permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones. (49)

4.4.2. Pruebas realizadas a la solución

Para verificar el funcionamiento del sistema se realizaron pruebas funcionales aplicando el método de caja negra y específicamente la técnica de partición de equivalencia. Mediante el uso de esta técnica son

determinados un conjunto de datos válidos y no válidos para ser ejecutados como condiciones de entradas al sistema. Son definidas dos tipos fundamentales de variables de equivalencia: las validas que representan las posibles entradas con reacción favorable a partir su entrada en el sistema y las no validas representan las entradas de datos erróneos.

A partir de cada requisito funcional fueron elaborados los diseños de casos de prueba de los cuales se muestra una representación a continuación:

Casos de pruebas de caja negra generados al aplicar dicha técnica.

Requisito a probar: ¨Adicionar reservación de visitas¨.

Tabla 9. Caso de prueba: ¨Adicionar reservación de visitas¨.

Nombre del requisito	**Descripción general**	**Escenarios de pruebas**	**Flujo del escenario**
1: Adicionar reservación de visitas.	El sistema permite realizar registro de personas externas.	EP 1.1: Adicionar visita introduciendo datos válidos.	➢ Se presiona el botón **Adicionar**. ➢ Se introducen los datos de la persona correctamente. ➢ Se presiona el botón **Aceptar**. ➢ Se muestra un mensaje de información. Se presiona el botón **Aceptar**
		EP 1.2: Adicionar visita introduciendo datos inválidos.	➢ Se presiona el botón **Adicionar.** ➢ Se introducen los datos inválidos de la persona. ➢ Se presiona el botón **Aceptar**. Se muestra un mensaje informando del error.
		EP 1.3: Adicionar visita dejando	➢ Se presiona el botón **Adicionar.**

		campos vacíos.	➢ Se introducen los datos dejando algún campo en blanco. ➢ Se presiona el botón **Aceptar**. Se muestra un mensaje de error.
		EP 1.4: Cancelar.	➢ Se presiona el botón **Adicionar.** ➢ Se introducen o no los datos de la persona. Se presiona el botón **Cancelar**.

Luego de aplicado el método de prueba al sistema, se obtuvieron resultados satisfactorios desde el punto de vista funcional de la solución. Las no conformidades descubiertas fueron atendidas logrando un correcto comportamiento de la solución desarrollada ante diferentes situaciones. Luego de la segunda interacción la solución estaba libre de no conformidades.

A continuación se presenta el número de no conformidades detectadas tras cada iteración de pruebas realizadas donde 9 de estas son de documentación y 1 de validación:

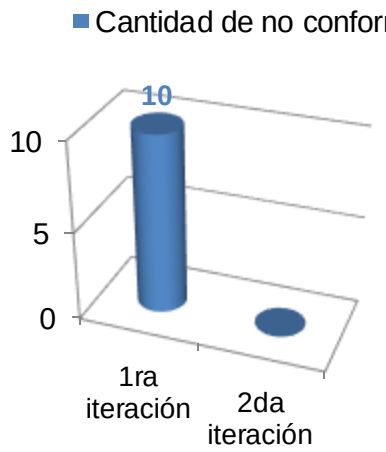


Fig. 23: Resultado de la aplicación de las pruebas de caja negra.

4.5. Conclusiones parciales.

Al finalizar el presente capítulo vale destacar que:

- El actual capítulo se presentó la estructura física de la solución y los estándares de código para su implementación. Se aplicó del mismo modo la métrica: Relaciones entre clases que permitió evaluar especialmente la complejidad en la implementación de la solución en la actual investigación.
- Se realizaron las pruebas de caja negra para comprobar el correcto funcionamiento de la solución propuesta. Luego de realizar la primera iteración de pruebas se detectaron un total de 9 no conformidades las cuales fueron resueltas en la segunda iteración permitiendo cumplir con los requisitos capturados en la primera fase del desarrollo.

CONCLUSIONES

Luego de la realización de este trabajo de diploma se puede determinar que se desarrollaron las tareas definidas a fin de dar cumplimiento al objetivo propuesto, lo cual conllevó a obtener como resultado:

- El estudio de soluciones análogas facilitó el análisis de soluciones que implementan variantes para la gestión de reservaciones en determinadas áreas de servicios, proporcionando un marco de referencia para conformar una solución fundamentada.
- Desarrollar el modelado del negocio asociado al proceso de reservación de visitas a trabajadores internos en la UCI, permitió comprender los elementos significativos del negocio y fueron identificadas las posibles funcionalidades de la solución a implementar.
- El análisis y el diseño del sistema de gestión para la reservación de visitas a trabajadores internos en la UCI, permitió recoger todas las necesidades del cliente, además de lograr un mejor entendimiento de los procesos que fueron informatizados, garantizando una comprensión exacta sobre las funcionalidades a informatizar.
- Las pruebas de caja negra permitieron encontrar y corregir los errores no detectados durante la implementación posibilitando cumplir con las especificaciones requeridas y la validación del sistema implementado, las mismas arrojaron resultados satisfactorios demostrando la calidad del sistema desarrollado.

RECOMENDACIONES

Una vez finalizado el presente trabajo de diploma se recomienda:

- Integrar el componente de notificaciones al sistema.
- Realizar mejoras al consumir servicios del LDAP, de forma tal que la base de datos del sistema siempre esté actualizada con los trabajadores efectuarán las solicitudes de visitas.

BIBLIOGRAFÍA REFERENCIADA

1. **Coca, Juan R.** Tendencias de las Telecomunicaciones. [En línea] Las TICs crecen en el mundo al 30% anual , 14 de abril de 2009. [Citado el: 4 de febrero de 2013.] http://www.tendencias21.net/Las-TICs-crecen-en-el-mundo-al-30-anual_a3164.html.

2. **RAE.** Diccionario de la Lengua Española. [En línea] 22 edición. [Citado el: 20 de febrero de 2013.] http://lema.rae.es/drae/?val=servicio.

3. Definicion.DE. [En línea] [Citado el: 30 de 11 de 2012.] http://definicion.de/gestion/.

4. sensagent. [En línea] http://diccionario.sensagent.com/reservaci%C3%B3n/es-es/.

5. software sabre. [En línea] 2006. [Citado el: 1 de diciembre de 2012.] http://softwaresabre.blogspot.com/.

6. Playo. [En línea] [Citado el: 1 de diciembre de 2012.] http://es.planyo.com/.

7. planyo. [En línea] [Citado el: 1 de diciembre de 2012.] http://es.planyo.com/what-is-planyo-booking-system.php.

8. **Residencia, Analista del Centro de Informatización Universitaria (CENIA) para el módulo de.** Gestion Universitaria. *Sistema para la reservación de visitas.* La Habana, 2 de marzo de 2013.

9. **CEIGFE.** *Proceso de desarrollo y gestión de proyectos de software.* La Habana : s.n., 2009.

10. **Milestone Consulting.** *Curso Modelado de Negocio con BPMN 2.0 y UML.* [En línea] [Citado el: 05 de diciembre de 2012.] http://www.milestone.com.mx/CursoModeladoNegociosBPMN.htm..

11. **Ferré, Grau Xavier.** *"Tutorial UML, Desarrollo Orientado a Objetos con UML.".* [En línea] 2010. [Citado el: 20 de enero de 2011.] http://www.clikear.com/manuales/uml/index.aspx..

12. **Gómez Baryolo Ing. Oiner, Morejón Borbón Ing. Yoandry, García Tejo Ing. Darien.** *SOFTWARE, ARQUITECTURA TECNOLÓGICA PARA EL DESARROLLO DE.*

13. **Babin, Lee.** *Introducción a Ajax con PHP.* España : ANAYA MULTIMEDIA, 2007.

14. **'Cutter' Blades, Steve y Frederick, Shea.** *Learning Ext JS.* 2008.

15. tecnoretales. [En línea] [Citado el: 6 de diciembre de 2012.] http://www.tecnoretales.com/programacion/que-es-doctrine-orm/..

16. Zend Framework » ¿Qué es Zend Framework? [online]. [En línea] 2009. [Citado el: 10 de diciembre de 2012.] http://spanish.zendfw.com/que-es-zend-framework/.

17. Ciberaula. [En línea] 2010. [Citado el: 4 de febrero de 2013.] http://php.ciberaula.com/articulo/introduccion_php/.

18. ¿Qué es JavaScript? - Definición de Javascript [online]. . [En línea] [Citado el: 10 de diciembre de 2012.] http://www.masadelante.com/faqs/javascript..

19. **Pérez, Iván Nieto.** Diccionario de siglas CSS (Cascading Style Sheets). [En línea] 2008. [Citado el: 15 de enero de 2013.] http://www.elcodigo.net/tutoriales/diccionario.html.

20. Extensible Markup Language (XML). [En línea] 2003. [Citado el: 15 de enero de 2013.] http://www.w3.org/XML/.

21. **RENDÓN ARTOLA ARIADNA, CASTELLANOS PÉREZ MANUEL ALEJANDRO.** *Modelación y construcción de los componentes Gestor de actividades y Calendario para el Subsistema Planificación por Objetivos del Sistema Integral de Gestión de Entidades Cedrux.* La Habana : Universidad de las Ciencias Informáticas, 2010.

22. Documentación del Servidor HTTP Apache 2.0. [En línea] . [En línea] 2009. [Citado el: 4 de febrero de 2013.] http://httpd.apache.org/docs/2.0/es/.

23. Visual-paradigm. [En línea]. [En línea] 2005. [Citado el: 4 de febrero de 2013.] http://www.visual-paradigm.com..

24. Observatorio Tecnológico. [En línea] Ministerio de Educación , Cultura y Deporte , España, 17 de enero de 2008. [Citado el: 4 de febrero de 2013.] http://recursostic.educacion.es/observatorio/web/es/software/software-general/548-luis-garcia.

25. PostgreSQL. [En línea] [Citado el: 5 de diciembre de 2012.] http://www.postgresql.org/about/press/presskit91/es/.

26. *¿Qué es NetBeans? [online].* [En línea] [Citado el: 3 de diciembre de 2012.] http://ayuda-java.blogspot.com/2007/07/qu-es-netbeans.html.

27. Caracteristicas NetBeans 6.9. [En línea] [Citado el: 3 de diciembre de 2012.] http://www.jujuyjug.com.ar/portal/index.php?option=com_content&view=article&id=74:netbeans-69-final&catid=3:newsflash&Itemid=50..

28. Tera.loc. [En línea] [Citado el: 6 de febrero de 2013.] http://www.teraloc.com/portal/index.php?option=com_content&view=article&id=51&Itemid=92.

29. Process library. [En línea] 2013. [Citado el: 7 de febrero de 2013.] http://www.processlibrary.biz/index.php?main_page=index&cPath=69.

30. Tecnología y Synergix. [En línea] 10 de julio de 2008. [Citado el: 5 de febrero de 2013.] http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/.

31. **Serrano, Dr. Gonzalo León.** *Ingeniería de sistemas de software.* Madrid , España : Isdefe, 1996.

32. **José Escalona María, Koch Nora.** *Ingeniería de Requisitos en Aplicaciones para la Web – Un estudio comparativo .* Sevilla,España : Departamento de Lenguajes y Sistemas Informáticos Escuela Técnica Superior de Ingeniería Informática Universidad de Sevilla, 2002.

33. **Pressman, Roger S.** *Ingeniería de Software, un enfoque práctico.* 2002.

34. **Cristóbal, González Almirón.** AdictosAlTrabajo. [En línea] 05 de mayo de 2010. [Citado el: 10 de junio de 2013.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=RM.

35. **Armas, Dayamy Linares.** Biblioteca Virtual de la Universidad de Cienfuegos. [En línea] 2010. [Citado el: 7 de febrero de 2013.] http://www.eumed.net/libros-gratis/2010b/698/Requisitos%20funcionales.htm.

36. Metodología de Gestión de Requisitos. [En línea] [Citado el: 7 de febrero de 2013.] https://sites.google.com/site/metodologiareq/capitulo-ii/tecnicas-para-identificar-requisitos-funcionales-y-no-funcionales.

37. Seguridad de la Informacion, Software y Tecnología. [En línea] 4 de febrero de 2013. http://www.comusoft.com/modelo-vista-controlador-definicion-y-caracteristicas.

38. **Prestashop 5 Estrellas.** Desarrollo de aplicaciones con el framework MVC de .NET (I). [En línea] 16 de junio de 2012. [Citado el: 10 de junio de 2013.] http://miblogtecnico.wordpress.com/2012/07/16/desarrollo-de-aplicaciones-con-el-framework-mvc-de-net-i/.

39. inf.utfsm. [En línea] [Citado el: 20 de marzo de 2013.] http://www.inf.utfsm.cl/%7Evisconti/ili236/Docu.

40. ldc.usb. [En línea] [Citado el: 20 de marzo de 2013.] http://www.ldc.usb.ve/~teruel/ci3711/patron3a/index.html..

41. **Olivares, Mc Juan Carlos.** *Patrones de diseño.* Mexico : Dirección General de Educación Superior Tecnológica.

42. Chronotech, el futuro está en nuestras manos. Modelos De Datos. [En línea] [Citado el: 20 de marzo de 2013.] http://sites.google.com/site/jalexiscv/modelosdedatos..

43. Universidad del Valle. [En línea] California, 20 de marzo de 2013. [Citado el: 18 de abril de 2013.] http://eisc.univalle.edu.co/cursos/web/material/750091M/1/Modelo-de-implementacion.pdf.

44. Universidad Nacional Autónoma de México. [En línea] 22 de abril de 2013. [Citado el: 22 de abril de 2013.] http://recursosweb.unam.mx/recursos-web/creacion-de-paginas-web/estandares-de-codificacion/.

45. **Mon Alicia, Estayno Marcelo G. , Scalzone Patricia.** *Definición de un marco de trabajo para la implementación inicial de un Modelo de Proceso Software.* Argentina : Universidad Nacional de La Plata, 2006.

46. **IEEE.** *Standard Glossary of Software Engineering Terms .* 93.

47. desarrolloweb.com. [En línea] http://www.desarrolloweb.com/articulos-copyleft/articulo-metricas-de-software.html.

48. **Ramírez, Jaime.** Métodos de Prueba del Software. Unidad de Programación. [En línea] 2012.

49. **S. Pressman, Roger.** *Ingeniería de Software un enfoque práctico.* Madrid : s.n., 2001.

BIBLIOGRAFÍA CONSULTADA

epsilon eridani. [En línea] [Citado el: 1 de diciembre de 2012.] http://www.epsilon-eridani.com/cubic/ap/cubic.php/doc/ALO---central-de-reservas-online-58.html.

Fernández González, Mairelys y Zorrilla Rivera, Osley. *Diseño e implementación del componente Ajuste al Costo del Subsistema Costos y Procesos del Sistema Integral de Gestión de Entidades CEDRUX.* La Habana : UCI, 2010.

Pressman, Roger. *Ingenieria de Software.* EE.UU : Higner Educa.

Sparx Systems. [En línea]. [En línea] 2007. [Citado el: 20 de marzo de 2013.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html..

Microsoft. [En línea] [Citado el: 17 de abril de 2013.] http://support.microsoft.com/.

Cubana de Aviacion. [En línea] [Citado el: 1 de diciembre de 2012.] https//www.cubana.cu/home/.

E.U. Ingeniería Técnica Informática de Oviedo. Sitio web de la E.U.I.T.I.O. [En línea] [Citado el: 5 de Noviembre de 2011.] http://www.petra.euitio.uniovi.es%2F~i1667065%2FHD%2Fdocumentos%2FEntornos%2520de%2520Desarrollo%2520Integrado.pdf&ei=GannTrWxL8Pqgge-2.

Fernández González, Mairelys y Zorrilla Rivera, Osley. Diseño e implementación del componente Ajuste al Costo del Subsistema Costos y Procesos del Sistema Integral de Gestión de Entidades CEDRUX. *Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.* Ciudad de la Habana : UCI, 2010.

S. Pressman, Roger. *Ingeniería de Software un enfoque práctico.* Madrid : s.n., 2001.

Unidad de Compatibilización Integración y Desarrollo de Soluciones Informáticas para la defensa. *Proceso de Desarrollo y Gestión de Proyectos de Software.* La Habana : s.n., 2009.

Garcia Calderón, JoséAlejandro y Rodríguez Moreno,Iran Roberto. *Sistema Integrado de Transportación (SIT). Implementación de los Módulos Administración - Configuración y Seguridad.* La Habana : UCI, 2009.

CEIGE. *Proceso de desarrollo y gestión de proyectos de software.* La Habana : s.n., 2009.

Morejón Borbón, Yoandry y García Tejo, Darien. *ARQUITECTURA TECNOLÓGICA PARA EL DESARROLLO DE SOFTWARE.* La Habana : UCI.

Ministerio de la Informática y las Comunicaciones. Agencia de Control y Supervisión. [En línea] 2011. [Citado el: 26 de 11 de 2012.] http://www.acs-mic.cu/sw_certificados.htm.

Directorio de Servicios Web UCI. *Directorio de Servicios Web UCI.* [En línea] [Citado el: 6 de Diciembre de 2011.] http://uddi.uci.cu/.

Doctrine-Project.org. Doctrine. [En línea] 2010. [Citado el: 23 de 11 de 2012.] http://www.doctrine-project.org.

Doctrine. [En línea] 2010. [Citado el: 30 de Noviembre de 2011.] http://www.doctrine-project.org.

Visual Paradim. freedownloadmanager.org. [En línea] freedownloadmanager.org. [Citado el: 30 de Noviembre de 2011.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

freedownloadmanager.org. [En línea] 5 de 3 de 2007. [Citado el: 20 de 11 de 2012.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

Fundamentos Informaticos. [En línea] 2 de febrero de 2008. [Citado el: 3 de diciembre de 2012.] http://fundamentosinformaticosjl.wordpress.com/category/base-de-datos/.

Hoteles now. [En línea] [Citado el: 1 de diciembre de 2012.] http://www.hotelesnow.com/red/reservas/sistema-de-reservas-una-pequena-revolucion.

Grupo de autores. HTML.net. [En línea] 2012. [Citado el: 16 de Enero de 2012.] http://es.html.net/tutorials/html/lesson2.php.

HTML.net. HTML.net. [En línea] 2012. [Citado el: 19 de 11 de 2012.] http://es.html.net/tutorials/html/lesson2.php.

Informática Hoy. Informática Hoy. [En línea] [Citado el: 27 de 11 de 2012.] http://www.informatica-hoy.com.ar/sap/Que-es-SAP.php.

Colectivo de Json. JSON. [En línea] JSON, 2011. [Citado el: 30 de Noviembre de 2011.] http://www.json.org/json-es.html.

Metodologías de Sistemas. Metodologías de Sistemas. [En línea] 2007. [Citado el: 25 de 11 de 2012.] http://metodologiasdesistemas.blogspot.com/2007/10/que-es-un-orm-object-relational-mapping.html..

Universidad Politécnica de Madrid. *Patrones del "Gang of Four".* Facultad de informática. Madrid : s.n.

Patrones del "Gang of Four". Facultad de informática. Madrid : s.n., 2010.

The PHP Group. PHP: Hipertext Preprocesor. [En línea] php.net, 4 de 2 de 2011. [Citado el: 19 de 11 de 2012.] http://php.net/manual/es/intro-whatis.php.

PHP: Hipertext Preprocesor. [En línea] php.net, 4 de Febrero de 2011. [Citado el: 25 de Noviembre de 2011.] http://php.net/manual/es/intro-whatis.php.

Sabre. [En línea] [Citado el: 4 de febrero de 2013.] http://softwaresabre.blogspot.com/.

Scribd. [En línea] http://www.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.

Carnegie Mellon University. Software Engineering Institute. [En línea] 2012. [Citado el: 17 de Enero de 2012.] http://www.sei.cmu.edu/architecture/start/glossary/community.cfm.

Universidad de San Martin. [En línea] [Citado el: 3 de diciembre de 2012.] http://www.usmp.edu.pe/publicaciones/boletin/fia/info49/articulos/RUP%20vs.%20XP.pdf.

Española, Real Academia de la Lengua. *Diccionario de la Lengua Española.* 2013.

Javier Nieto Diego, Pablo Ramos Fernández. *El Patrón Fachada.* s.l. : Departamento de Informática y Automática.

Navarro, Javier Mendoza. *Diseño del Sistema de Tarjeta de Crédito Con UML.* s.l. : Tesis Digitales UNMSM.

Análisis conceptual de las principales interacciones entre la gestión de información, la gestión documental y la gestión del conocimiento. **Ponjuán Dante, Gloria.** La Habana ,Cuba : http://bvs.sld.cu/revistas/aci/vol18_1_08/aci07708.htm, 2008, Vol. 18_1_08.

Presman, Roger. *Ingeniería de Software, un enfoque práctico.* La Habana : Editoral Félix varela, 2012.

RAE. *Diccionario de la Lengua Española 22 Edicion.*

eco-finanzas.com. eco-finanzas.com. *ESTADO DE RESULTADOS.* [En línea] [Citado el: 18 de 11 de 2012.] http://www.eco-finanzas.com/diccionario/E/ESTADO_DE_RESULTADOS.htm.

ERP Access. ERP Access. [En línea] 2009. [Citado el: 5 de 2 de 2013.] http://erpaccess.blogspot.com/2012/11/analisis-financiero-con-microsoft-access.html.

Unidad de Compatibilización Integración y Desarrollo de Soluciones Informáticas para la defensa. *Proceso de Desarrollo y Gestión de Proyectos de Software.* Ciudad de la Habana : s.n., 2009.

EVA. Entorno virtual de aprendizaje. *El diseño de software.* [En línea] 2003. http://eva.uci.cu/file.php/158/Documentos/Recursos_bibliograficos/Libros_y_articulos_UD_1/Diseno_de_software/Diseno_del_Software.pdf.