Universidad de las Ciencias Informáticas

Centro de Informatización Universitaria

Facultad 1

Título: Módulo Complejo Residencial v2.0 para el Sistema de Gestión de Residencia.

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

Autores:

Grettel Barrio Marshall.

Osvaldo José Aguilera Velázquez.

Tutores:

Ing. Adriana Alfonso Luis.

Ing. Dasiel Alberto Pérez Suárez.

Ing. Yurisleis Alonso Beatón.

La Habana. Junio, 2013.

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Autorizamos a dicho centro para que haga el uso que estime pertinente con este trabajo. Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Grettel Barrio Marshall

_____

Firma del Autor

Osvaldo José Aguilera Velázquez

_____

Firma del Autor

Adriana Alfonso Luis

_____

Firma del Tutor

Dasiel Alberto Pérez Suárez

_____

Firma del Tutor

Yurisleis Alonso Beatón

_____

Firma del Tutor

DATOS DE CONTACTO

Nombre y Apellidos: Grettel Barrio Marshall

Correo electrónico: gbarrio@estudiantes.uci.cu

Universidad de las Ciencias Informáticas. La Habana, Cuba.

Nombre y Apellidos: Osvaldo José Aguilera Velázquez

Correo electrónico: ojaguilera@estudiantes.uci.cu

Universidad de las Ciencias Informáticas. La Habana, Cuba.

Ing. Adriana Alfonso Luis

Correo electrónico: aluis@uci.cu

Graduada en la Universidad de las Ciencias Informáticas del curso 2009-2010. Pertenece a la Facultad 1 al Centro de Informatización Universitaria (CENIA) donde se ha desempeñado como analista de diferentes proyectos, actualmente pertenece al proyecto Residencia del Departamento Gestión Universitaria.

Ing. Dasiel Alberto Pérez Suárez

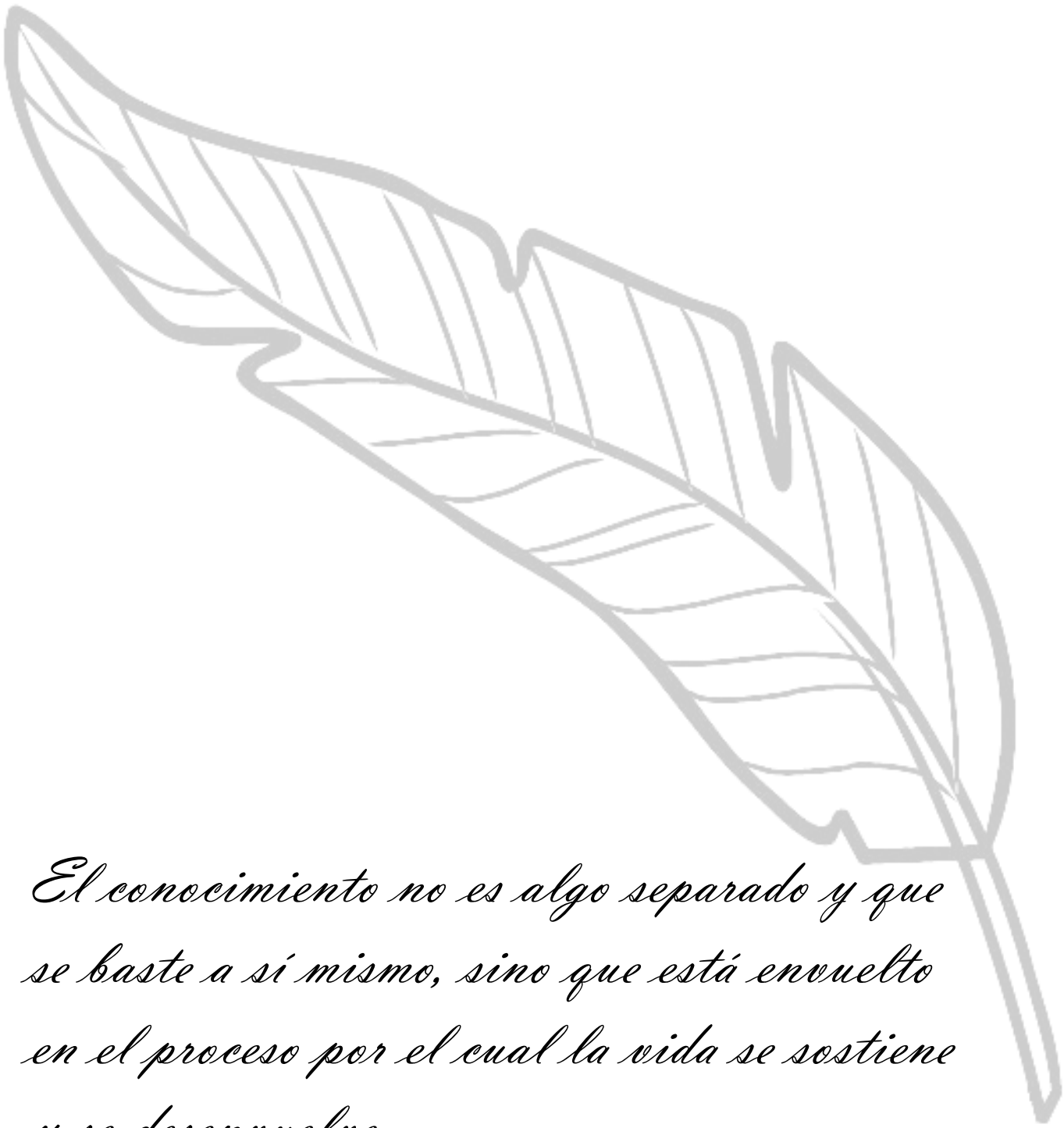
Correo electrónico: daperez@uci.cu

Graduado en Ingeniería en Ciencias Informáticas en el curso 2007-2008 en la Universidad de las Ciencias Informáticas. Cuenta con cuatro años de experiencia de trabajo en la universidad. En los años que lleva como docente ha sido profesor de asignaturas como Sistema Operativo, Ética Informática e Inglés y ha tutorado siete trabajos de diploma. Además ha estado vinculado a proyectos del Centro de Informatización Universitaria.

Ing. Yurisleis Alonso Beatón

Correo electrónico: ybeaton@uci.cu

Graduado en Ingeniería en Ciencias Informáticas en el curso 2008 - 2009 en la Universidad de las Ciencias Informáticas. Cuenta con tres años de experiencia de trabajo en la universidad. En los años que lleva como docente ha sido profesor de asignaturas de la disciplina de Programación y ha tutorado tres trabajos de diploma. Se ha desempeñado como desarrollador en varios proyectos productivos del Centro de Informatización Universitaria (CENIA) y como tutor de Práctica Profesional IV y V.



El conocimiento no es algo separado y que se baste a sí mismo, sino que está envuelto en el proceso por el cual la vida se sostiene y se desenvuelve.

John Dewey

Dedicatoria

Grettel

A mis padres
Por todo el apoyo, la entrega y la dedicación que me han brindado.
Por ser mí guía y ejemplo y ser tan maravillosos conmigo.
A mis maestros por jugar un papel importante en mi educación y formación.

Osvaldo

A mis padres
Por siempre estar a mi lado. Porque sus consejos siempre me guían y me hacen ser una mejor persona. Hoy estoy más convencido que nunca que mis hermanos y yo somos su razón de ser.
A ustedes les dedico este trabajo, nadie lo merece más.
A mi abuela Gladys:
Sé que donde estés siempre estás cuidándonos. Me arrepiento de no haber sido un mejor nieto.
Me viste crecer, pero desafortunadamente no me viste hacerme ingeniero.

Agradecimientos

Grettel

A mis padres
Por la confianza que han depositado en mí y por la formación que me han inculcado durante toda mi crianza. Por apoyarme en las decisiones que he tomado y alentarme a seguir adelante. A ustedes que son mi razón de ser.
A mi familia
Por el cariño y la atención que siempre me han brindado. Por ser una parte importante de vida y enseñarme lo importante que es mantener la familia unida.
A mis tutores Adriana, Dasiel y Yuri
Por la constancia, la ayuda, la orientación y la exigencia durante la etapa de desarrollo de este trabajo. Sin ustedes llegar hasta aquí habría sido todo un desafío.
A Osvaldo
Por compartir este momento tan importante conmigo y soportarme durante estos 5 años con la paciencia y el cariño del primer día. Por tenerme tan presente y considerarme una personita importante durante su estancia en la universidad.
A todas las personas que colaboraron en la confección de este trabajo: Yasmany, Yoan Carlos, Raidel, JD, Nano. A Julio y Ari que forman parte indispensable de la solución.
A los que han compartido conmigo buenos y malos momentos durante el transcurso por la universidad y hemos vivido juntos experiencias inimaginables: Greisy, Iliada, Kenia, Daimel, Lilianne, Adrián, Yare, Rafa.
A todos los que se han apoyado en mí, gracias por la confianza. A los que con tanto cariño me quieren, gracias por aguantarme.

Osvaldo

A mis padres

Por haberme dado la vida. Por apoyarme y estar presentes siempre que los necesito. Nunca lo demuestro lo suficiente, pero son mi mayor orgullo. Sin ustedes nunca hubiese llegado a este momento.

A mi familia

La familia lo es todo. Sin su amor y apoyo mi vida sería más que difícil, imposible.

A mis amigos

He hecho muy buenos amigos aquí en universidad y he mantenido los de mi provincia. A todos les quiero agradecer por estar conmigo en los buenos y en los malos momentos; porque cuando los necesito acuden sin pedir nada a cambio.

A mis tutores:

Por siempre estar peleando para que se hiciera este trabajo de la mejor forma posible. Porque cuando nos encontramos con Yuri como único tutor, fue solo decirles a Dasiel y a Adriana que necesitábamos ayuda. Por siempre buscar la manera de hacer que esta tesis avanzara. Por las críticas, los regaños y por el ánimo.

A Arianna y Julio:

Por haber hecho posible que hiciéramos esta tesis. Porque siempre buscaban la forma de que su trabajo hiciera más fácil el de Grettel y mío. Por tantos dolores de cabeza que nos dimos mutuamente.

A Grettel:

No sé qué hubiese pasado de no haber sido tú mi compañera de tesis. Este trabajo es gracias a ti, de no ser porque siempre me estabas regañando y diciendo que me pusiera a trabajar, la tesis estaría estancada. Gracias por contarte entre mis mejores amigos. Todos los que te conocemos sabemos que siempre tenemos a alguien con quien contar a tu lado. Por haberme soportado por 5 años.

A todos los que nos ayudaron a hacer posible esta tesis. Pido perdón por los que me falten, pero quiero agradecer especialmente a Yoan Carlos, Yasmany, Raidel, Víctor, Juan Daniel, Adrián, Sael, Manuel, Israel, Lilianne, Yarelis, Greisy, Daimel, Iliada.

RESUMEN

La Universidad de las Ciencias Informáticas (UCI) cuenta con un conjunto de edificios, dentro de su residencia, que son destinados al alojamiento de visitantes, y que forman el denominado Complejo Residencial; diseñado para brindarle hospedaje y avituallamiento al personal nacional o extranjero autorizado. Se propone el desarrollo web de la segunda versión (v2.0) del módulo Complejo Residencial, como parte del Sistema de Gestión de Residencia. Dicho sistema permite que se automatice dentro del proceso de hospedaje, la ubicación de los huéspedes en los apartamentos, el registro de reservaciones sin una previa solicitud, la permuta entre huéspedes, notificaciones de entrada y salida de las personas alojadas, así como la posibilidad de acceder a datos históricos almacenados en el sistema. Para conseguir este objetivo se hace uso de un proceso de desarrollo de *software* con enfoque ágil y orientado al nivel dos de CMMI, de los lenguajes PHP, JavaScript, HTML, CSS; del marco de trabajo GUUD (basado en CodeIgniter con jQuery), servidor web Apache y PostgreSQL como sistema gestor de base de datos. Una vez implementado el módulo se realizan pruebas funcionales y se corrigen los errores detectados, garantizando de esta manera que todas las funcionalidades se comporten de la forma correcta.

Palabras clave: Complejo Residencial, hospedaje, residencia, sistema de gestión.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Con el transcurso de los años, las tecnologías han ido evolucionando a gran velocidad, lo que ha traído consigo que estas jueguen un papel fundamental en el desarrollo de las diferentes esferas de la sociedad. El hombre se ha visto en la obligación de aprender cada vez más rápido, volviéndose a su vez más eficiente en su trabajo, trayendo como consecuencia una mayor competencia entre las empresas y mayor calidad de los productos. Cuba no queda exenta de la importancia de este desarrollo tecnológico, por lo que en el año 2002 crea la Universidad de las Ciencias Informáticas (UCI). Esta tiene como objetivo producir aplicaciones y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación; y servir de soporte a la industria cubana de la informática. Todo esto ha posibilitado el desarrollo de varios proyectos, a través de los cuales, se han informatizado una parte de los principales procesos que se llevan a cabo tanto del centro educacional como en la sociedad cubana e incluso a nivel internacional.

Dentro de la residencia de la Universidad de las Ciencias Informáticas existe un conjunto de edificios destinados al alojamiento de visitantes, que es denominado Complejo Residencial. El objeto social del mismo es brindarle hospedaje y avituallamiento a especialistas nacionales o extranjeros, así como a los directivos y trabajadores que colaboren con la superación de profesores, especialistas y técnicos, en la confección de herramientas informáticas que se utilizan en Cuba o en el exterior y que participen en eventos investigativos, deportivos o culturales, tanto internos como externos. Este procedimiento parte de la aprobación de las solicitudes de hospedaje por parte de la Rectora o por el Vicerrector de Extensión Universitaria y Residencia, y posteriormente se hace a la reservación y ubicación dentro del Complejo Residencial.

Actualmente existe un sistema web (módulo Complejo Residencial v1.0) desarrollado en el Centro de Informatización Universitaria (CENIA) que gestiona parcialmente el proceso de hospedaje. Contiene funcionalidades como: la gestión del proceso de solicitudes, el control de entrada y salida de huéspedes, así como la entrega y recepción de la llave, la ubicación de las reservaciones y la generación de reportes de capacidad. Pero la aplicación no permite registrar el hospedaje de una persona sin solicitud previa, lo que puede provocar un descontrol en la planificación y distribución de las habitaciones. No está preparada para controlar los incidentes que pueden ocurrir durante la estancia de un huésped, no es capaz de proporcionar notificaciones sobre las próximas entradas al Complejo Residencial ni las salidas de las personas alojadas, lo que trae como consecuencia que haya que consultar cada una de las reservaciones para conocer la fecha de estancia. Además no está apta para responder a la solicitud de permuta por parte de algún huésped, no está habilitada para gestionar diferentes servicios de costo que brinda el Complejo Residencial como son: el de habitaciones y lavandería o cualquier otro que pueda surgir y puedan ser solicitados por los huéspedes. Lo antes

planteado demuestra que este sistema no cumple con las necesidades actuales del Complejo Residencial ni puede enfrentarse al hecho de cobrar los servicios que ofrece, lo que trajo consigo que el módulo Complejo Residencial v1.0 nunca llegara a desplegarse ya que no facilita la gestión del proceso de alojamiento que tiene lugar en el mismo.

Debido a la situación existente los autores proponen como **problema de investigación**: ¿Cómo organizar y agilizar el control del proceso de hospedaje en el Complejo Residencial de la Universidad de las Ciencias Informáticas?

El **objeto de estudio** definido es el proceso de gestión de hospedaje, enmarcando como **campo de acción** el proceso de gestión de hospedaje en el Complejo Residencial de la Universidad de las Ciencias Informáticas.

Los autores proponen como **objetivo general** desarrollar la versión 2.0 del módulo Complejo Residencial del Sistema de Gestión de Residencia de la Universidad de las Ciencias Informáticas para facilitar el control y organización de los procesos asociados al alojamiento de visitantes en la instalación.

Se determinaron los siguientes **objetivos específicos**:

- Analizar los principios teóricos de la investigación enfocados a los procesos que se llevan a cabo en el Complejo Residencial.
- Describir la propuesta de solución de la versión 2.0 del módulo Complejo Residencial para el Sistema de Gestión de Residencia.
- Implementar las nuevas funcionalidades para la versión 2.0 del módulo Complejo Residencial, como son:
    - Gestión de hospedaje sin solicitud.
    - Gestión de permutas.
    - Gestión de servicios de costo.
    - Gestión de incidencias.
    - Gestión de medios básicos.
    - Gestión de notificaciones.
- Realizar pruebas funcionales a la solución final.

Para cumplir el objetivo planteado se establecen las siguientes **tareas de investigación**:

- Elaboración del marco teórico de la investigación.

- Revisión bibliográfica acerca de los sistemas de hospedaje en línea existentes para el desarrollo exitoso de la investigación.
- Preparación del entorno de trabajo con las herramientas a utilizar en el desarrollo del sistema en cuestión.
- Modelación de los procesos involucrados en el hospedaje en el Complejo Residencial de la Universidad de las Ciencias Informáticas.
- Identificación de los requisitos funcionales y no funcionales.
- Definición del estándar de programación que se empleará en el desarrollo del módulo.
- Definición del modelo de datos que se empleará en el desarrollo del sistema.
- Desarrollo del sistema propuesto.
- Diseño de los casos de pruebas a la solución propuesta.
- Ejecución de las pruebas y respuesta a las posibles no conformidades detectadas.

Para la realización del trabajo se tuvieron en cuenta **métodos científicos investigativos** como:

- **Método Histórico–Lógico:** Se utiliza para analizar a nivel nacional e internacional el uso de los sistemas informáticos que tengan similitud con el que se va a implementar.
- **Método Analítico-Sintético:** Análisis de la bibliografía disponible para realizar un estudio sobre el estado de los sistemas de hospedaje. Posibilita definir los conceptos principales y analizar otras soluciones existentes. Se sintetizan las principales características de las herramientas para el desarrollo del sistema y las ventajas del uso de las mismas.

Con este trabajo los autores pretenden crear una aplicación que satisfaga las necesidades actuales del Complejo Residencial de la Universidad de las Ciencias Informáticas, ya que las funcionalidades con las que cuenta la versión actual no cumplen con todas las expectativas del cliente. Se logrará así brindar mejores servicios, además de un mayor control y organización en sus procesos.
Con la culminación de la investigación presentada mediante el análisis e implementación de la versión 2.0 del módulo Complejo Residencial para el Sistema de Gestión de Residencia se esperan obtener resultados satisfactorios, entre los que se encuentran:

- Control sistemático de las reservaciones realizadas por los directivos autorizados.
- Alta confidencialidad y seguridad de la información en la realización de las reservaciones sin solicitud y permuta.
- Generación de información histórica y estadística con gran rapidez simplificando el contenido de trabajo del personal de la instalación.

Estructuración del contenido

El trabajo está conformado en tres capítulos que se describen brevemente a continuación:

Capítulo 1 Fundamentación teórica: Se presentan los elementos teóricos que sirven de base a la investigación del problema planteado, analizando los principales conceptos relacionados con el objeto de estudio. Se realiza un estudio del estado del arte de las herramientas y tecnologías que se utilizan en el desarrollo del módulo Complejo Residencial v2.0 para el Sistema de Gestión de Residencia de la Universidad de las Ciencias Informáticas.

Capítulo 2 Descripción de la solución propuesta: Se analiza la solución propuesta, realizando una descripción del flujo actual del proceso de hospedaje, se presenta una descripción de los procesos que serán gestionados por el módulo, se identifican las funcionalidades del módulo a través de la especificación de requisitos, se muestran diferentes sistemas de hospedaje con utilización nacional e internacional.

Capítulo 3 Construcción y validación de la solución propuesta: Se describe la arquitectura de desarrollo y el estándar de código a seguir para la implementación del módulo. Se diseñan y ejecutan los casos de prueba basados en los requisitos. Además se presenta la implantación del módulo, a través de la distribución física del Sistema de Gestión Universitaria (SGU).

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

Introducción

En el presente capítulo se describen de forma detallada los conceptos relacionados con el dominio del problema. Se realiza un estudio a nivel nacional e internacional de los sistemas de hospedaje que presenten similitud con el sistema propuesto. Se describen las tecnologías, herramientas, lenguajes y metodología que permiten dar solución al problema. Todo esto basado en las características y necesidades del Complejo Residencial de la Universidad de las Ciencias Informáticas.

1.1 Conceptos asociados al problema

Cuando está en desarrollo un *software*, tienden a existir problemas en la comunicación y entendimiento entre los clientes, desarrolladores y usuarios finales. Esto se debe al manejo de diferentes términos y conceptos que pueden ser desconocidos para el personal externo, de ahí lo importante que es definir y argumentar los que pueden ser los principales conceptos que se utilizan a lo largo del presente trabajo.

Complejo Residencial

Es un espacio diseñado para ofrecer una atención integral a sus residentes. El “Hotelito de la UCI”, se estableció a principios del año 2004 y desde sus inicios pertenecía a la Dirección de Atención a Profesores y Especialistas Invitados. En mayo del 2009, se constituye como dirección independiente, con el nombre de Complejo Residencial. Tiene como objeto social brindar alojamiento y avituallamiento a especialistas cubanos y extranjeros a partir de la recepción de solicitudes de hospedaje que tengan la aprobación de la Rectora o del Vicerrector de Residencia.

Sistema de Gestión de Residencia

Es un subsistema de gestión de la residencia universitaria como espacio de alojamiento tanto de estudiantes, como profesores, especialistas, personal vinculado a la actividad productiva en el desarrollo de soluciones informáticas y otros. La solución comprende los módulos de Alojamiento, Avituallamiento y lavandería, Trabajo educativo, Complejo Residencial y Reportes. Tiene como objetivo general desarrollar un sistema informático que permita automatizar y optimizar la gestión de los procesos asociados a la residencia de la Universidad de las Ciencias Informáticas.

Módulo Complejo Residencial

Es uno de los módulos del Sistema de Gestión de Residencia, que informatiza los procesos de hospedaje en el Complejo Residencial. Brinda numerosas funcionalidades que facilitan el desarrollo de estos procesos, que en ocasiones resulta engorroso controlar de forma manual por el gran volumen de información, entre otros factores. Este módulo surge a partir de la necesidad de informatizar el proceso de hospedaje que se realiza en la instalación y que se implementan en la versión 2.0 del mismo.

1.2 Análisis de sistemas de hospedaje

1.2.1 Estado de sistemas de hospedaje a nivel internacional

***Software* HotelOnePlace**

Es un *software* para pequeños y medianos hoteles de menos de cincuenta habitaciones. Administra las habitaciones, temporadas y tarifas, reservaciones, huéspedes, la cuenta y facturación del alojado. Permite llevar un control de las reservaciones y un ágil manejo de la recepción. Admite visualizar gráficamente la ocupación del hotel y obtener reportes y estadísticas para la toma de decisiones. El sistema se encuentra vinculado a su sitio de internet para llevar un buen control de las reservaciones desde un solo punto. Permite a sus visitantes realizar reservaciones y pagar en línea. (Logismic, 2012)

Para su desarrollo se trabajó con las principales tecnologías propiedad de empresas líderes y tecnologías *Open Source* del mercado como:

- **Tecnologías de Microsoft:** .NET, ASP/ASP.NET, C/C++/C#
- **Tecnologías de Base de Datos:** MySQL, MS SQL Server, MS Access, PostgreSQL
- **Tecnologías *Open Source*:** Apache, PHP, Ruby on Rails
- **Tecnologías Varias:** AJAX, Flash, HTML, Java/J2EE, JavaScript, XML, Velneo

Este *software* fue desarrollado con tecnología privativa que va en contra de las políticas del centro, que aboga por el uso del *software* libre y además para su uso es necesario comprarlo.

Sistema de gestión Hotelera Express

El sistema cuenta con varios módulos entre los que se encuentran Reserva, Recepción, Caja y facturación que se describen a continuación. (CQR Sistemas S.R.L, 2012)

Reserva. Permite llevar el control de la disponibilidad de habitaciones en tiempo real, consultar reservas por nombre, número o empresa. Dar de alta, modificar y cancelar reservas. Consulta de disponibilidad por pantalla o listado. Emisión de reportes y consultas.

Recepción. Permite realizar el *check-in* con reserva o sin reserva. Completar la ficha del huésped tomando los datos de la base de pasajeros históricos, consulta del plano del hotel por pantalla, asignándole colores a las habitaciones según el tipo o el estado de las mismas. Consulta de pasajeros alojados, e históricos, habitaciones ocupadas, estado de cuenta de pasajeros, alojados por empresa, posibles salidas, todo por pantalla o impresora. Modificación de datos de alojados, cambio de habitación, tarifas, estado de habitaciones y prolongación de estadía.

Caja y facturación. Permite cargar en forma manual los consumos de alojados, que no son cargados en forma automática, ej.: gastos de lavandería. Además permite consultar el detalle de la cuenta de huéspedes o grupos. Emitir facturas parciales de cuenta, dividir la cuenta del cliente, emitir factura final

para cerrar la habitación y facturas de pagos adelantados. Genera una caja que podrá ser consultada por pantalla o por impresora, y emite un cierre por cada turno.

Requerimientos mínimos para sistemas monousuario / terminal de red:

- Pentium II 400 con 128 MB de RAM (Recomendado: Pentium III 500 con 256 MB de RAM).
- Microsoft Windows 98 (Segunda edición), Windows 2000, Windows NT 4.0.
- Monitor VGA
- 200 MB de espacio en disco disponible.
- Lectora de CD-ROM (para la instalación), recomendado Grabadora de CD para hacer *Backup*.

Requerimientos mínimos para server / red:

- Recomendado: Pentium III 500 con 256 MB de RAM.
- Microsoft Windows 98 (Segunda edición), Windows 2000, Windows NT 4.0, Novell Netware, Linux.
- Monitor VGA
- 500 MB de espacio en disco disponible.
- Lectora de CD-ROM (para la instalación).
- Cableado estructurado Nivel 5 (UTP).
- *Switch*.

Su mayor inconveniente es que la adquisición del *software* depende de la compra de la licencia para su uso.

1.2.2 Estado de sistemas de hospedaje a nivel nacional

eHotel

La Suite eHotel, de la empresa Datys, es una aplicación PMS (*Property Management System*), que ofrece una solución integral para la administración de un hotel o una multipropiedad de hoteles y *resorts*, en un ambiente amigable y flexible, englobando las alternativas operacionales más utilizadas, con una interfaz gráfica, sencilla e intuitiva. Cuenta con una gran variedad de soluciones para cadenas hoteleras, que incluye central de reservaciones, *datawarehouse* y gestión centralizada del historial de huéspedes y empresas. (Datys, 2012)

eHotel permite:

- Procesos de chequeo de entrada y salida de huéspedes, y gestión de habitaciones con gran flexibilidad.

- Reservaciones individuales y de grupo.
- Clasificación de los huéspedes totalmente configurable.
- Consulta en línea de la disponibilidad.
- Generación automática de tarifas.
- Contratación y tarifas multimoneda.
- Estimación del importe total de cada estancia.
- Facturación y generación de información histórica y estadística.
- Manejo centralizado de la información de huéspedes y compañías.
- Acceso remoto vía web para consulta y monitoreo de las actividades del hotel.
- Garantiza que las reservaciones vía web quedarán registradas en tiempo real afectando simultáneamente su inventario de habitaciones.

eHotel tiene entre sus características:

- Está implementada sobre una plataforma eficaz y con la potencia de las bases de datos de Oracle.
- Diseñada sobre una arquitectura multicapas y modular que prevé su constante y necesaria evolución.

Este sistema presenta el inconveniente de que utiliza la tecnología de Oracle por lo que no cumple con las políticas del Centro de Informatización Universitaria que están encaminadas al uso de tecnologías libres.

Suite Zun

Sistema de gestión hotelera integrado por módulos fácilmente adaptables a cualquier hotel, independientemente de la cantidad de habitaciones que posea o la complejidad de su operación. Está bajo la autoría de GET (Grupo de Electrónica para el Turismo) que es una organización empresarial perteneciente al Ministerio del Turismo.

Entre los módulos de la Suite Zun se encuentran:

ZUNsa: Administrador del Sistema (*System Administration*). Configuración general del sistema.

ZUNpms: Carpeta (*Property Management System*). Reservaciones, alojamiento, facturación, ama de llaves, previsiones, auditoría nocturna, estadísticas e históricos.

ZUNacc: Contabilidad (*Accounting*). Configuración multicompañías, multimoneda, plan de cuentas en varios niveles, captación de presupuestos, registros contables, estados financieros, cobros y pagos, otros procesos contables.

E-Hotel

Este sistema gestiona de manera eficiente las operaciones más utilizadas en la administración de un hotel o multipropiedad de hoteles y *resorts*. Sus módulos cuentan con un ambiente amistoso y flexible que se adapta a actividades extrahoteleras. (GET, 2012)

Administrador general del sistema. Configuración general del sistema: instalación del servidor de datos Oracle, creación de la base de datos, instalación del DBsetup de eHotel en el servidor corporativo y en el hotel.

Carpeta. Reservaciones, alojamiento, habitaciones, grupos, rangos, características de habitaciones, formas de cobro, departamentos, créditos internos, permite facturación simple y compuesta.

Contabilidad. Configuración multicompañías, multimoneda, plan de cuentas en varios niveles, captación de presupuestos, registros contables, estados financieros, cobros y pagos, otros procesos contables.

Activos fijos. Permite gestionar los movimientos de un medio básico, tales como: compra, venta, traspasos recibidos y efectuados, enviados y recibidos a reparar. La depreciación se realiza de forma automática.

Estas herramientas que ofrece GET deben ser compradas para poder utilizarlas.

1.2.3 Módulo Complejo Residencial v1.0 del Sistema de Gestión de Residencia

Es un módulo de fácil configuración que forma parte del Sistema de Gestión de Residencia y está integrado al Sistema de Gestión Universitaria. Dentro de sus funcionalidades están:

- **Solicitud.** Se gestionan las solicitudes de hospedaje en el Complejo Residencial, que van desde la creación hasta la aprobación o denegación de la misma por parte de las personas autorizadas.
- **Reservación.** Se realiza la ubicación de cada una de las solicitudes aprobadas y posteriormente en la fecha establecida se le da entrada o salida a los huéspedes en la entidad y en el sistema.
- **Configuración.** Se administran los aspectos configurables del sistema como por ejemplo: los eventos y las camareras. También se brindan reportes de capacidad disponible para un intervalo de fechas dado.

El sistema permite gestionar parcialmente el proceso de hospedaje que se lleva a cabo en el Complejo Residencial, pero no se llegó a desplegar.

1.2.4 Aportes obtenidos de los sistemas de hospedaje estudiados

En el estudio realizado se pudo observar que a excepción del módulo Complejo Residencial v1.0 del Sistema de Gestión de Residencia estos sistemas son configurables y adaptables a las características de cualquier institución dedicada al alojamiento de personas, pero la adquisición de los mismos es costosa y han sido desarrollados con tecnología privativa. De igual forma no son compatibles, de manera parcial o completa con las pautas de diseño visual establecidas por el CENIA, ni es posible su integración al Sistema de Gestión Universitaria. Es válido destacar que aunque ninguno satisface en su totalidad los requerimientos necesarios para el funcionamiento eficiente del alojamiento en la UCI sí aportaron ideas novedosas y de utilidad a la presente investigación. Fueron útiles para proponer al cliente nuevas funcionalidades que enriquecieron el sistema desde el punto de vista funcional como: la generación automática de tarifas y la configuración de tarifas multimoneda, así como el control de los medios básicos de las habitaciones. En el caso de la versión 1.0 del módulo Complejo Residencial se reutilizaron las funcionalidades: “Registrar entrada” y “Registrar salida”, además del mecanismo de integración con algunos módulos del SGU. Por todo lo antes expuesto se propone la versión 2.0 del módulo que favorezca la gestión de información en el Complejo Residencial situado en la residencia universitaria de la UCI, y de esta forma brindar mejores servicios y garantizar un mayor control y organización en sus procesos.

1.3 Entorno de desarrollo

Para darle solución al problema planteado en la investigación, se hizo necesario el estudio de los elementos que van a conformar el entorno de trabajo de la misma. A la hora de realizar este estudio se tuvo en cuenta que la solución informática que se pretende construir va a formar parte del Sistema de Gestión Universitaria, por lo que se estudió el entorno de desarrollo que este sistema usa y a partir de ahí se seleccionaron todos los elementos necesarios para integrarlos a la investigación. A continuación se presentan todas las herramientas, lenguajes y tecnologías, con sus principales características, establecidas por la dirección del CENIA.

1.3.1 Lenguajes de desarrollo.

PHP 5.3.3

PHP es un lenguaje de programación interpretado que posibilita la generación dinámica de contenidos en un servidor web. Se destacan entre sus características principales su alto rendimiento, potencia, facilidad de aprendizaje y los pocos recursos que consume. Entre las características que posee este lenguaje y que lo convierten en una potente herramienta están: (Gutmans, Saetther, et al., 2005)

- Es un lenguaje multiplataforma.
- Soporte sólido para Programación Orientada a Objetos.
- Incorpora bibliotecas que contienen funciones integradas para realizar útiles tareas relacionadas con la Web.

- Es un *software* de código abierto.
- Soporta bases de datos entre las que se pueden mencionar: MySQL y PostgreSQL.

JavaScript 1.2

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.(Librosweb.es, 2013)

HTML 4

El Lenguaje de Marcado de Hipertexto trata básicamente de un conjunto de etiquetas que sirven para definir el texto y otros elementos que compondrán una página web. Es un lenguaje de programación que se utiliza para el desarrollo de páginas de Internet. EL HTML permite describir la estructura y el contenido en forma de texto, además de complementar el texto con objetos tales como imágenes.(Alvarez, 2001)

CSS 2

Es un lenguaje de hojas de estilos en cascada creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML entre otros. La separación de los contenidos y su presentación muestran numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo, también llamados “documentos semánticos”. Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes. (Librosweb.es, 2013)

1.3.2 Tecnologías de desarrollo

Las tecnologías de desarrollo han estado en un lugar primordial en la construcción de productos gracias al avance de las TIC, debido a que, posibilitan que el producto se desarrolle en un ambiente agradable y que cuente con buena calidad. Dentro de las más utilizadas están los marcos de trabajo y las librerías. En los siguientes epígrafes se explican las tecnologías que se van a utilizar en el desarrollo de la investigación.

1.3.2.1 Marco de trabajo

Un marco de trabajo o “*framework* es un conjunto de bibliotecas, herramientas y normas a seguir que ayudan a desarrollar aplicaciones. (…) Un *framework* está compuesto por varios segmentos/ componentes que interactúan los unos con los otros. Las aplicaciones pueden escribirse de manera más eficaz si utilizamos un *framework* adaptado al proyecto en lugar de tener que volver a inventar la rueda cada vez.” (Lafosse, 2010)

GUUD es un marco de trabajo propuesto por el equipo de arquitectura del CENIA. El mismo integra a su vez los marcos de trabajo CodeIgniter en su versión 1.7.3 y jQuery 1.3.2 en una sola infraestructura, razón por la cual posee las mismas características que estos. En esta integración se incluyen además un conjunto de novedades o mejoras y algunas modificaciones hechas específicamente al CodeIgniter, además creación de *plugins* y componentes de interfaz de usuario en jQuery. A continuación se muestra una relación de las principales mejoras y modificaciones que incorpora el GUUD en su infraestructura. (García Vidal, 2011)

Del lado del cliente:

1. Se implementaron una serie de *widgets* para utilizarlos de interfaz de algunos de los *widgets* base de *jquery-ui* como por ejemplo el *date*, el *tab* (ambos son interfaces de los *widgets* de mismo nombre de *jquery-ui*) y el *popup* (interfaz del *dialog* de *jquery-ui*). Además de los ya mencionados, se implementaron otros nuevos entre los que se encuentran: *attach*, menú, *message*, *tooltip*, *form* (se construyó con la unión de los *plugins form* de jQuery el cual se utiliza para el envío de formularios AJAX y el *validate* utilizado para validar formularios), *grid* (utiliza como *plugin* el *jqgrid*), *multiselect* (para hacer selecciones múltiples), *navbar* (para la creación de barras de navegación), *tree* (para la creación de árboles) y el *graph* (utiliza la librería *Highchart*).
2. Se le implementó un *plugin* a jQuery para el manejo de espacios de nombre e internacionalización.
3. Se implementaron funciones comunes para todo el sistema (contenidas en los archivos *core.js* y *common.js*) entre las que se destacan: *loadIn*, *getDataJson*, *createSelect*, *isArray*, *isFunction*, *site_url*.

Del lado del servidor (hechas a CodeIgniter):

1. Se le agregó manejo de excepciones y mensajes.

2. Se le implementó el IOC para la interacción entre módulos.

3. Se le añadió la característica de la modularidad o sea que una aplicación pueda dividirse en módulos. CodeIgniter no cuenta con esta posibilidad.

4. Se añadieron, modificaron y extendieron los *helpers* o asistentes.

A continuación se describen por separados los dos marcos de trabajos que conforman la propuesta.

CodeIgniter 1.7.3

CodeIgniter es un marco de trabajo para desarrollar aplicaciones web utilizando el lenguaje de programación PHP. Es un producto de código libre y se puede utilizar en cualquier aplicación. Como cualquier otro marco de trabajo, CodeIgniter contiene una serie de librerías que sirven para el desarrollo de aplicaciones web y además propone una manera de desarrollarlas que se debe seguir para obtener provecho de la aplicación. Marca una manera específica de codificar las páginas web y clasificar sus diferentes *scripts*, que sirve para que el código esté organizado y sea más fácil de crear y mantener. CodeIgniter implementa el proceso de desarrollo llamado Modelo Vista Controlador, que es un estándar de programación de aplicaciones, utilizado tanto para hacer sitios web como programas tradicionales. Este marco de trabajo contiene muchas ayudas para la creación de aplicaciones PHP avanzadas, que hacen que el proceso de desarrollo más rápido. A la vez, define una arquitectura de

desarrollo que hará que se programe de una manera más ordenada y contiene diversas herramientas que ayudan a hacer aplicaciones más versátiles y seguras.

Características generales de CodeIgniter

Algunos de los puntos más interesantes sobre este marco de trabajo, sobre todo en comparación con otros productos similares, son su versatilidad, compatibilidad, facilidad de instalación, flexibilidad, ligereza y que posee una gran documentación. Sin duda, lo más destacable de CodeIgniter es su accesibilidad, ya que se puede utilizar en la mayor gama de entornos. (Alvarez, 2009)

jQuery 1.3.2

jQuery es un marco de trabajo para el lenguaje JavaScript. Ofrece una infraestructura con la que se tendrá mayor facilidad para la creación de aplicaciones complejas del lado del cliente. Es un producto serio, estable, bien documentado y con un gran equipo de desarrolladores a cargo de su mejora y actualización. Entre sus funciones se encuentra la manipulación de documentos, manejo de eventos, animación y AJAX mucho más simple con una API fácil de usar que funciona con una gran cantidad de navegadores. Con una combinación de versatilidad y extensibilidad, ha cambiado la forma que millones de personas trabajan con JavaScript.(Alvarez, 2009)

Con jQuery se ahorra muchas líneas de código, se mejora el tiempo de creación y depuración, esta tecnología tiene licencia para uso en cualquier tipo de plataforma, personal o comercial.

1.3.2.2 Sistema Gestor de Bases de Datos

PostgreSQL 8.4.13

PostgreSQL es un poderoso sistema de gestión de bases de datos objeto-relacional de código abierto. Puede ser utilizado en la mayor parte de los sistemas operativos, incluyendo Linux, UNIX y Windows. Tiene soporte completo para llaves foráneas, uniones, vistas, disparadores y procedimientos almacenados. Al poseer licencia de código abierto, brinda al usuario la libertad de usarlo, modificarlo y distribuirlo en la forma en que desee. Cualquier modificación que haga es suya para hacer lo que desee con ella. Las reglas del sistema permiten al diseñador de base de datos crear reglas que identifiquen operaciones específicas para una tabla o vista dada y transformarla dinámicamente en operaciones alternas cuando sean procesadas.(Martínez, 2010)

Servidor Web Apache 2.2.2

Apache es un servidor web configurable, robusto y estable. Su licencia permite modificar su código fuente, incluyéndolo así dentro de los impresionantes productos del *software* libre que son utilizados en la actualidad. Es multiplataforma, permite la creación de ficheros *log* según la necesidad del administrador, posibilitando un mayor control de la información. (Apache Software Foundation, 2010)

1.3.3 Herramientas de desarrollo

Visual Paradigm 8.0

Herramienta para desarrollo de aplicaciones utilizando modelado UML ideal para ingenieros de *software*, analistas de sistemas y arquitectos de sistemas que están interesados en construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. Soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El *software* de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.(Visual Paradigm International Ltd 2007)

Características:

- Navegación intuitiva entre la escritura del código y su visualización,
- Potente generador de informes en formato PDF/HTML,
- Ambiente visualmente superior de modelado,
- Sincronización de código fuente en tiempo real.

Entorno Integrado de Desarrollo

El Entorno Integrado de Desarrollo (IDE), es un programa informático compuesto por un conjunto de herramientas de programación. Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Python, Java, C#, Delphi, Visual Basic y PHP. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto.

NetBeans 7.2

NetBeans IDE es un entorno de desarrollo, una herramienta para que los programadores puedan 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 extender el NetBeans IDE. Es un producto libre y gratuito sin restricciones de uso.(NetBeans, 2012)

pgAdmin 1.10.5

pgAdmin es una plataforma de desarrollo de PostgreSQL. La aplicación puede utilizarse en Linux, FreeBSD, Solaris, Mac OSX y Windows para administrar PostgreSQL 7.3 y superiores, así como las versiones comerciales y derivados de PostgreSQL como servidor de *Postgres Plus Advanced* y base de datos *Greenplum*. Diseñado para responder a las necesidades de todos los usuarios, desde escribir simples consultas SQL hasta crear bases de datos complejas. La interfaz gráfica soporta todas las características de PostgreSQL y facilita la administración. La aplicación también incluye un resaltado de sintaxis SQL editor, un editor de código del lado del servidor y mucho más. La conexión con el

servidor se puede hacer a través de TCP/IP o *Unix Domain Sockets* (en plataformas * nix), y puede ser encriptado mediante SSL para la seguridad. No se requieren controladores adicionales para comunicarse con el servidor de base de datos. Es un *software* libre publicado bajo la licencia PostgreSQL.(The pgAdmin Development Team, 2002)

Pencil Project 1.3.4

Esta herramienta es una simple extensión de Firefox que cuenta con una considerable potencia y flexibilidad. Es la alternativa libre y multiplataforma para realizar el prototipado. Permite crear proyectos con varias pantallas, en las cuales es posible añadir cualquier elemento de una GUI. Cuenta con multitud de elementos web. Una de las características más importantes es que permite alinear objetos para que el conjunto quede ordenado. (Pencil Project, 2010)

Apache Jmeter 2.6

Es una aplicación de escritorio (100 % Java) de código abierto desarrollada por Apache Jakarta que es un proyecto de la *Apache Software Foundation*. Básicamente, fue diseñada para realizar pruebas de carga y medir el rendimiento de sistemas. Esta herramienta se puede utilizar para probar el rendimiento de los recursos estáticos y dinámicos de los sistemas (archivos, *servlets*, *scripts* de Perl, objetos Java, bases de datos y consultas, servidores FTP) y para simular la carga de un servidor, red u objeto con el fin de poner a prueba su resistencia (conocidas como pruebas de estrés). (Apache Software Foundation, 2012)

1.4 Proceso de desarrollo de *software*

Un proceso de desarrollo de *software* no es único, tiene como propósito la producción eficaz y eficiente de un producto *software* que reúna los requisitos del cliente. No existe uno que sea efectivo para todos los contextos de proyectos de desarrollo, aunque existe un conjunto de actividades fundamentales que se muestran presentes en todos. (Pressman, 2010)

Dentro de estas actividades está la especificación de *software*, el diseño e implementación, la validación y la evolución. Además, Pressman menciona un conjunto de “actividades protectoras”, que se aplican a lo largo de todo el proceso de desarrollo del *software*. Dentro de estas se encuentran: el seguimiento y control de proyecto, las revisiones técnicas, la garantía de la calidad del *software*, las mediciones, la gestión de riesgo, etc.(Pressman, 2010)

Para guiar el desarrollo del módulo se utiliza el proceso de desarrollo basado en prácticas de las metodologías Programación Extrema (XP), Scrum y prácticas del nivel dos del Modelo Integrado de Capacidad y Madurez (CMMI) que CENIA utiliza en algunos de sus proyectos. CMMI le permite a una organización aproximarse a la mejora de procesos y a las evaluaciones usando dos representaciones diferentes, la representación continua y la representación por etapas o escalonada. Este último tipo de representación es el que se emplea en los proyectos productivos de la universidad.

La representación por etapas utiliza conjuntos predefinidos de áreas de proceso para definir un camino de mejora en la organización, estableciendo para ello cinco niveles de madurez (Inicial, Administrado, Definido, Cuantitativamente administrado y Optimizado). Cada nivel de madurez proporciona un

conjunto de áreas de proceso que caracterizan diferentes comportamientos organizativos a cumplir por la entidad. (Llano Castro, 2010)

El nivel dos posee siete áreas de procesos que se ilustran en la figura.

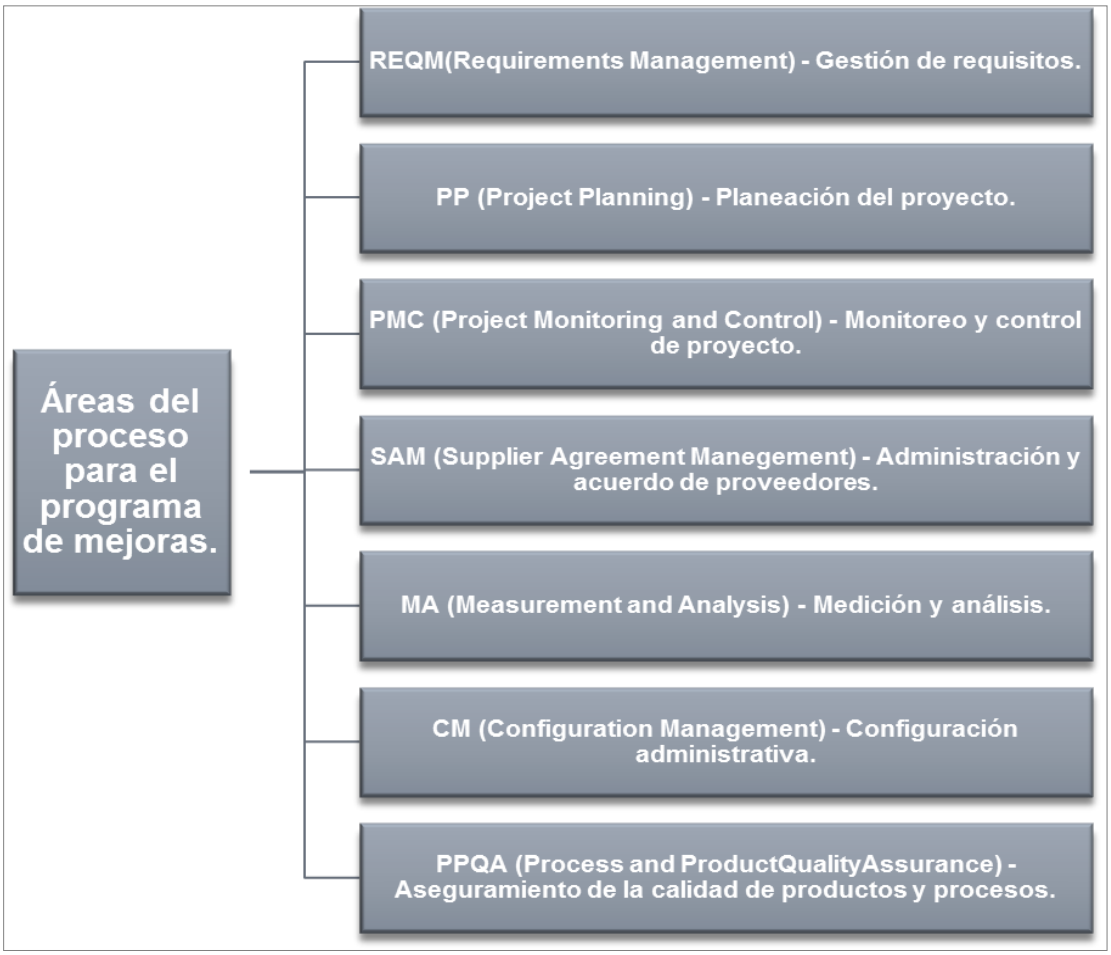


Figura 1. Áreas de proceso que intervienen en el nivel 2 de CMMI.

Programación Extrema (XP) es una metodología ágil de desarrollo de *software* que posee cuatro tareas fundamentales: planificación, diseño, desarrollo y pruebas. Esta metodología está basada en la simplicidad durante el desarrollo, la comunicación entre las partes implicadas (clientes y desarrolladores) y la retroalimentación para poder reutilizar el código desarrollado. En su concepción establece entregas frecuentes con posibilidad de refactorización continua, permitiendo mejorar el diseño cada vez que se añade una funcionalidad. Para su implementación XP establece un conjunto de prácticas que deben ser empleadas en los proyectos de desarrollo. (Llano Castro, 2010)

Scrum es una metodología ágil enfocada a la gestión de proyectos. Sus principales características se pueden resumir en dos: el desarrollo de sprint o iteraciones y reuniones a lo largo del desarrollo. Las iteraciones en Scrum tienen una duración máxima de 30 días y el resultado de cada uno de ellas define un incremento del producto a desarrollar. La evolución del proyecto por la metodología se define a través de reuniones diarias donde el trabajo del día anterior es revisado por el equipo, previendo además la labor a realizar el día siguiente. (Llano Castro, 2010)

Conclusiones

El estudio de los diferentes sistemas de alojamiento en el ámbito internacional y nacional, permitió conocer las diferentes características que presentan los sistemas que informatizan los procesos de alojamiento. Los sistemas privativos estudiados no se pueden utilizar en la propuesta de solución, debido a que el país incurriría en gastos elevados a la hora de dar soporte y estos no cumplen con las políticas de migración de Cuba al uso del *software* libre. Para la aplicación de los sistemas del ámbito nacional e internacional estudiados se requiere comprar su respectiva licencia de uso. Además el módulo Complejo Residencial v1.0 del Sistema de Gestión de Residencia no satisface las necesidades del cliente, de ahí que sea primordial el desarrollo de su segunda versión. El estudio de las herramientas y tecnologías establecidas por la dirección del CENIA, permitió profundizar los conocimientos necesarios para el desarrollo de la versión 2.0 de dicho módulo.

CAPÍTULO 2: DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA.

Introducción

Una buena comprensión de la problemática y un adecuado estudio del flujo actual de los procesos, influyen directamente en que la solución propuesta satisfaga las necesidades del cliente, pues este es el ente fundamental para los que ofrecen productos de *software*. El presente capítulo aborda las principales características del módulo, se describe el objeto de estudio en el cual se llevan a cabo los procesos involucrados en el campo de acción y el desarrollo de la propuesta de solución. Se declaran las personas involucradas en el negocio, se especifican los requisitos funcionales y no funcionales para dar cumplimiento a las actividades que serán objeto de automatización. El proceso de desarrollo de *software* que se aplica es con un enfoque ágil basado en CMMI nivel dos, se establece la arquitectura y patrones de diseño utilizados, así como la descripción del modelo de datos del sistema.

2.1 Descripción del proceso de negocio.

La identificación de las actividades que se realizan para brindar el servicio de hospedaje en el Complejo Residencial de la UCI, permite tener un mejor conocimiento del proceso, éste se resume en tres actividades fundamentales las cuales se describen a continuación:

- Realizar reservación del huésped.

A la entidad llega un listado de las solicitudes de hospedaje que han sido aprobadas por los principales responsables de la universidad en esta área, dígase Rector(a), Asesores o Vicerrector(a) de Residencia y Extensión Universitaria. El técnico de alojamiento es el responsable de ubicar las reservaciones en las habitaciones disponibles y procede a informarle a la carpetera.

- Realizar entrada o salida del huésped.

La carpetera es la encargada de realizar la entrada y la salida de los huéspedes en las respectivas fechas. Para ello consulta las reservaciones ubicadas previamente por el técnico de alojamiento. Le hace entrega de la llave al huésped y recibe del mismo un depósito por el valor de la llave. Si durante la estancia del mismo se produce alguna incidencia, se realiza un análisis y posteriormente se registra. Una vez culminado el período de estancia de la reservación el huésped entrega la llave y recibe el dinero depositado en caso de que no la haya perdido.

- Gestionar servicios.

Durante el período de la reservación el huésped puede solicitar, mediante la carpetera, diferentes servicios que brinda el Complejo Residencial y que tienen un costo incluido como el de lavandería y cambio de avituallamiento.

2.1.1 Roles asociados al módulo Complejo Residencial v2.0.

Todas aquellas personas que forman parte del proceso de hospedaje se definen como personas involucradas con el módulo. A continuación se realiza una descripción de las mismas especificando el rol que desempeñan en el sistema.

Administrador: máximo responsable del proceso de hospedaje, es el único que tiene acceso a todas las funcionalidades del sistema.

Técnico de alojamiento: persona facultada para ubicar las solicitudes aprobadas y las reservaciones sin solicitud.

Carpetera: persona encargada de darle entrada y salida a los huéspedes en el Complejo Residencial. Tiene acceso a las siguientes funcionalidades: gestionar entrada y salida de los huéspedes, servicios e incidencias y reservación.

2.1.2 Características del módulo.

El módulo Complejo Residencial v2.0 está compuesto por cuatro agrupaciones principales, donde va a contener una serie de funcionalidades. A continuación se describen cada una de ellas:

Reservación

Proporciona un listado de las solicitudes aprobadas que serán ubicadas en la entidad. Además de un registro de las personas hospedadas y de las que se les dará salida próximamente.

Huésped

Contiene servicios, que no son de hospedaje, y que se le pueden asignar a los huéspedes que los soliciten. Además permite tener un control de las incidencias que se llevan a cabo en la instalación.

Configuración

Presenta los elementos configurables del sistema como son: eventos, camareras, moneda, servicio de costo, modalidad de servicio y precio, así como el tipo de incidencia y de medio básico. Brinda la opción de crear y modificar cada uno de estos elementos.

Información

Provee información sobre las capacidades por apartamentos y un listado de las futuras entradas de huéspedes al Complejo Residencial, así como la salida de los mismos en las fechas establecidas, además de un historial de los huéspedes.

2.1.3 Integración del módulo al Sistema de Gestión Universitaria (SGU)

Como se ha mencionado con anterioridad el módulo Complejo Residencial está incluido en el SGU y es de vital importancia integrar las nuevas funcionalidades utilizando los métodos y servicios brindados por otros sistemas y módulos dentro del mismo, entre los que se encuentran:

Solicitudes: gestiona todas las solicitudes del sistema Residencia, brinda información referente al estado de las solicitudes durante su ciclo de vida. Le ofrece al módulo Complejo Residencial el listado de los solicitantes con sus solicitudes aprobadas.

Personal: gestiona todos los datos de las personas en la base de datos.

Inmueble: contiene todos los edificios de la residencia de la universidad, lo que permite obtener los inmuebles que pertenecen al Complejo Residencial.

Eventos: gestiona la información referente a los diferentes eventos que se desarrollan en la universidad.

Notificaciones y alertas: brinda avisos a roles específicos sobre las acciones que se han llevado a cabo en el sistema, estos se pueden configurar para que envíen correos.

Aunque no de forma directa también se comunica con los siguientes módulos:

Seguridad: garantiza el acceso a la información dados los niveles de privilegio de cada usuario, haciendo uso de la arquitectura sobre la cual está desarrollado el sistema. En el módulo Complejo Residencial la seguridad permitiría que el administrador del complejo y los técnicos de alojamiento sean capaces de acceder a las funcionalidades en las que tengan establecidos los permisos, evitando el mal uso de información, corrupción y modificación por personas no autorizadas a acceder a la misma.

Trazas: gestiona todo lo referente a las incidencias de un usuario sobre el SGU, registrando el usuario, la acción realizada y el momento en que se ejecutó la misma. Permite el registro de las actividades que se realicen sobre el módulo.

2.2 Requisitos.

Según Sommerville "los requerimientos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requerimientos reflejan las necesidades de los clientes de un sistema que ayude a resolver algún problema como el control de un dispositivo, hacer un pedido o encontrar información".

2.2.1 Captura de requisitos

La captura de los requisitos se refiere a de dónde vienen los requisitos del *software* y cómo el ingeniero de *software* puede recogerlos. Es la primera etapa en la construcción de una comprensión del problema que el *software* quiere solucionar. (Swebok, 2004)

Técnicas de captura de requisitos

Dentro de las técnicas de captura de requisitos que se emplearon para la investigación se encuentran:

Entrevistas: se realizaron entrevistas estructuradas con las personas que se verían afectadas por el uso del sistema propuesto, ya sean usuarios finales que interactúan con el sistema como miembros de la organización en la que se desarrolla el proceso de negocio a informatizar, con el objetivo de entender el dominio del problema y sus necesidades (ver **Anexo 1**).

Prototipos: es una herramienta valiosa para clarificar requisitos confusos. Pueden actuar de una manera similar a los escenarios. En la investigación se usan para que los usuarios entiendan mejor qué información necesitan proporcionar.

2.2.2 Requisitos funcionales

Las técnicas de captura de requisitos permiten identificar los requisitos del cliente y del producto o sistema. Los requisitos del producto o del sistema se derivaron en cincuenta y dos requisitos funcionales. A continuación se muestran los requerimientos funcionales definidos para la propuesta de solución.

Tabla 1. Listado de requisitos funcionales

Requisitos funcionales	
RFCR1_Mostrar solicitudes aprobadas	RFCR27_Ver detalles de moneda
RFCR2_Ubicar solicitud	RFCR28_Crear servicio de costo
RFCR3_Ver detalles de las solicitudes aprobadas	RFRC29_Mostrar servicio de costo
RFCR4_Crear reservación sin solicitud	RFCR30_Modificar servicio de costo
RFCR5_Crear nueva persona	RFCR31_Ver detalles de servicio de costo
RFCR6_Realizar permuta	RFCR32_Crear modalidad de servicio
RFCR7_Mostrar huéspedes con incidencia	RFCR33_Mostrar modalidad de servicio
RFCR8_Registrar incidencia	RFCR34_Modificar modalidad de servicio
RFCR9_Mostrar incidencia del huésped	RFCR35_Ver detalles de modalidad de servicio
RFCR10_Modificar incidencia	RFCR36_Registrar precio
RFCR11_Ver detalles de incidencia	RFCR37_Mostrar precio
RFCR12_Asignar servicio	RFCR38_Modificar precio
RFCR13_Mostrar servicio asignado	RFCR39_Crear tipo de incidencia
RFCR14_Ver detalles de la tarifa del huésped	RFCR40_Mostrar tipo de incidencia
RFCR15_Modificar servicio asignado	RFCR41_Modificar tipo de incidencia
RFCR16_Eliminar servicio asignado	RFCR42_Ver detalles del tipo de incidencia
RFCR17_Ver detalles de servicio asignado	RFCR43_Crear tipo de medio básico
RFCR18_Mostrar próximas entradas	RFCR44_Mostrar tipos de medios básicos
RFCR19_Mostrar próximas salidas	RFCR45_Modificar tipos de medios básicos
RFCR20_Crear evento	RFCR46_Ver detalles del tipo de medio básico
RFCR21_Mostrar evento	RFCR47_Mostrar inmuebles
RFCR22_Modificar evento	RFCR48_Registrar medio básico
RFCR23_Ver detalles del evento	RFCR49_Mostrar medios básicos del inmueble
RFRC24_Crear moneda	RFCR50_Modificar medio básico del inmueble
RFCR25_Mostrar moneda	RFCR51_Ver detalles del medio básico
RFCR26_Modificar moneda	RFRC52_ Ver detalles del inmueble

Los requisitos funcionales fueron sometidos a una evaluación para encontrar la complejidad de los mismos. En la Tabla 2 se puede encontrar el resultado y la clasificación que le corresponde, esta clasificación que puede ser: alta, media o baja.

Tabla 2. Complejidad de los requisitos funcionales

Prioridad	Cantidad
Alta	**6**
Media	**24**

Baja	**21**
Total	**51**

Especificación de requisitos

La especificación de requisitos es definida por varios autores como un documento que especifica los requisitos para un sistema o componente. En la investigación al usar un proceso de desarrollo con enfoque ágil, se definen distintos artefactos, uno de ellos es la plantilla de "Especificación de Requisitos de *Software*" en la que se describen cada uno de los identificados. Ésta brinda la posibilidad de observar las vistas del sistema con su respectiva descripción, así como un detallado catálogo con los requisitos funcionales y no funcionales.

La tabla de los requisitos funcionales definida en el catálogo, está estructurada para recoger toda la información posible del requisito. Dentro de sus campos se encuentran:

- No: contiene el código que representa al requisito. La nomenclatura está compuesta por *RFCR_número*, donde *RF* hace referencia a requisito funcional, *CR* es el nombre del módulo Complejo Residencial y el *número* corresponde a un número consecutivo que se le asigna a cada uno.
- Nombre: contiene el nombre del requisito que se está describiendo.
- Descripción: recoge la descripción del requisito.
- Complejidad: establece la complejidad que se le asigna a cada funcionalidad, dividiéndose éstas en 3 (Alta, Media y Baja).
- Prioridad para el cliente: toman los mismos valores que la complejidad.
- Prototipo: contiene el prototipo de interfaz generado en la herramienta Pencil Project.
- Campos: reúne el nombre de los campos que se muestran en el prototipo de interfaz.
- Tipos de datos: incluye el tipo de dato que le corresponde a cada campo en la base de datos.
- Reglas o restricciones: cada campo tiene reglas que se deben cumplir a la hora de implementarlos y son recogidas aquí.
- Observaciones: contiene todas las observaciones que tiene el requisito funcional como son: mensajes, precondiciones, entre otras.

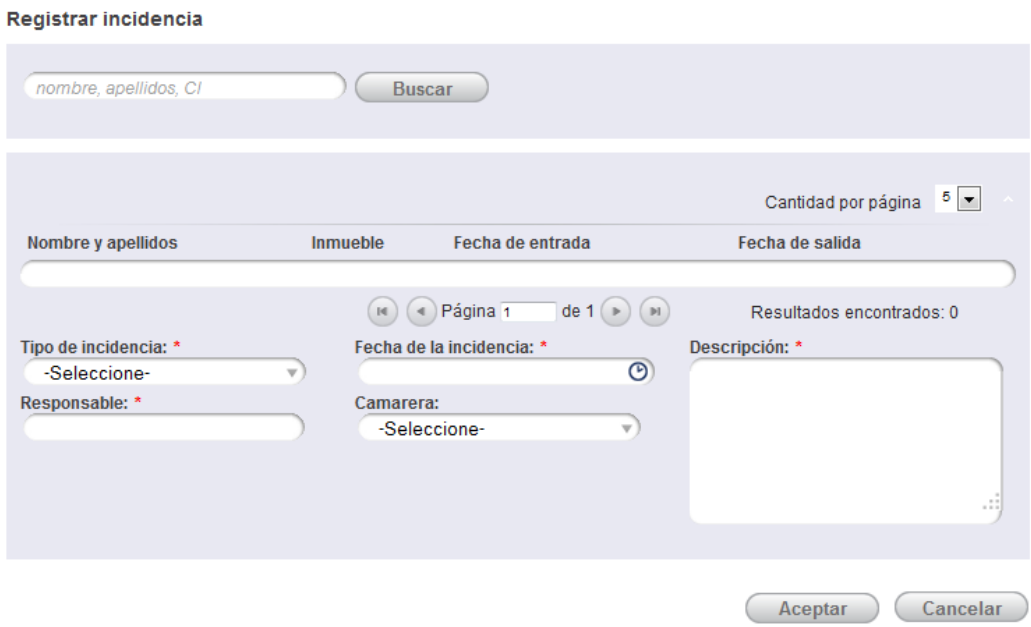
En la Tabla 3 se muestra un ejemplo de la especificación del requisito "Registrar incidencia" que corresponde a una de las funcionalidades de la versión del módulo. En el **Anexo 2** se encuentra la descripción de otros requisitos.

Tabla 3. Especificación de requisitos: RFCR26_Registrar incidencia.

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RFCR7	Registrar incidencia	En caso de que algún huésped tenga un incidente se crea un registro del mismo. Se busca al huésped, se selecciona el tipo de incidencia, la fecha en la que ocurrió y la camarera que la reporta. En el campo "Responsable" se especifica el	Media	Media

	nombre de la persona que se responsabiliza por el huésped. En el campo “Descripción” se establecen los detalles de la incidencia. Se puede escoger la cantidad de elementos a mostrar. Se selecciona el botón “Aceptar” y el sistema muestra el mensaje: “Se ha creado el elemento correctamente”.		

Prototipo



Campos	Tipos de Datos	Reglas o Restricciones
Buscar	Varchar	Los nombres y apellidos no deben contener números. En el caso del carnet de identidad no debe exceder los 11 dígitos.
Mostrar resultado	Campo de selección única	No procede
Tipo de incidencia	Lista desplegable	Campo obligatorio. Deben existir tipos de incidencias creados previamente en el sistema
Fecha de incidencia	Fecha	Campo obligatorio. Fecha con formato válido DD/MM/YYYY.
Camarera	Lista desplegable	Campo obligatorio. Deben existir camareras creadas previamente en el sistema.
Responsable	Vachar	Campo obligatorio. Admite un rango de caracteres válidos de 1 a 50 caracteres.
Descripción	Varchar	Campo obligatorio. Admite un rango de caracteres válidos de 0 a 250.
Observaciones	Si selecciona el botón “Cancelar” el sistema muestra un mensaje de confirmación: “¿Está seguro de realizar la acción?”.	

2.2.3 Requisitos no funcionales

Son restricciones de los servicios no funciones ofrecidos por el sistema. Incluye restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema (Sommerville, 2005).

Los requisitos no funcionales también fueron clasificados, en la Tabla 4 se puede encontrar el resultado de esa clasificación.

Tabla 4. Listado de requisitos no funcionales.

Requisitos no funcionales	Cantidad
Usabilidad	**13**
Confiabilidad	**2**
Eficiencia	**2**
Soporte	**2**
Restricciones de diseño	**2**
Requisitos para la documentación de usuarios en línea y ayuda del sistema	**3**
Interfaz	**4**
Requisitos Legales, de Derecho de Autor y otros	**2**
Estándares aplicables	**1**
Total	**31**

Los requisitos no funcionales dentro de la plantilla de requisitos son descritos según el tipo que corresponde. Son identificados por un código donde la nomenclatura está compuesta por *RnF_número*, donde: *RnF* significa que es un requisito no funcional y el *número* corresponde a un número consecutivo que se le asigna a cada uno.

Para la construcción del módulo se han especificado varios requisitos no funcionales, a continuación se describen los mismos:

Usabilidad

RnF_1 Desarrollar una solución web integrada al Sistema de Gestión Universitaria.

RnF_2 Permitir el acceso rápido a la información por parte de los usuarios y la fácil adaptación de usuarios sin experiencia.

RnF_3 Garantizar la confiabilidad de la información que se registra.

RnF_4 Adaptar el sistema al lenguaje y términos utilizados por los usuarios en la rama abordada con vista a una mayor comprensión por parte del cliente de la herramienta de trabajo.

RnF_5 Brindar potencialidades de capacitación orientadas a interfaces intuitivas, lo que enaltece la posibilidad de que el usuario aprenda mediante el uso y explotación de la herramienta.

RnF_6 El sistema debe tener un tiempo de respuesta corto.

RnF_7 El sistema debe soportar la conexión simultánea de todos los posibles usuarios.

RnF_8 Para el despliegue del sistema se debe contar en el servidor de bases de datos con PostgreSQL 8.4 bajo el sistema operativo Ubuntu Server 10.4.

RnF_9 Para el despliegue del sistema se debe contar en el servidor de aplicaciones web con: PHP v5.3 con las librerías php5-ldap, php5-gd, php5-mcrypt, php5-pgsql, php5-xsl, php5-openssl, Apache 2.2 con el módulo *rewrite* activado.

RnF_10 Para el uso del sistema se debe contar en la PC cliente con el navegador web Mozilla Firefox 3.6 o superior.

RnF_11 Para el uso del sistema se debe contar en la PC cliente con los siguientes sistemas operativos o versiones superiores: Windows XP y GNU/Linux.

RnF_12 El servidor web y el de base de datos deben contar con: Microprocesador Dual Core, 8GB de memoria RAM y 4 GB disco duro como mínimo para ambos.

RnF_13 El sistema debe ejecutarse en una PC cliente que tenga: 1GB RAM y 1 GB disco duro disponible como mínimo.

Confiabilidad

RnF_14 El sistema puede permanecer inactivo durante 10 minutos. Al cumplirse este término se cerrará la sesión teniendo que autenticarse el usuario nuevamente.

RnF_15 Cuando ocurre una excepción el sistema mostrará un mensaje explicativo del error ocurrido y permanecerá en el mismo estado sin realizar ninguna otra operación.

Eficiencia

RnF_16. El sistema debe tener un tiempo de respuesta promedio por operación de 3 a 7 segundos.

RnF_17 El sistema soportará una conexión simultánea de al menos 5 usuarios.

Soporte

RnF_18 Se proporcionará soporte al sistema siempre y cuando este lo necesite ya que el mismo se desplegará en la sede central.

RnF_19 El sistema debe cumplir con las normas de codificación, conversiones para nomenclatura, bibliotecas de clases, normas de acceso y utilidades de mantenimiento definidas en el documento CENIA_PRE_ADASP-v1.0 disponible en el expediente de proyecto.

Restricciones de diseño

RnF_20 El sistema debe cumplir con la arquitectura de información definida para el Sistema de Gestión Universitaria.

RnF_21 El sistema debe ser desarrollado con las herramientas definidas por el Centro de Informatización Universitaria.

Requisitos para la documentación de usuarios en línea y ayuda del sistema.

RNF_22 El sistema brinda como apoyo una ayuda funcional en la cual se refleja detalladamente la explicación de cada una de las pantallas con sus respectivas funcionalidades.

RnF_23 El sistema debe brindar como apoyo un Manual de Usuario en el cual se refleje detalladamente la explicación de cada una de las pantallas con sus respectivas funcionalidades.

RnF_24 Documentación actualizada del grupo de desarrollo: se precisa que la documentación del sistema esté actualizada en todos los aspectos, fases de trabajo y ciclos de desarrollo del mismo, permitiendo con ello un respaldo tanto ingenieril como legal del desarrollo de dicho sistema.

Interfaz

RNF_25 La interfaz web es sencilla con colores suaves a la vista y sin cúmulo de imágenes u objetos que distraigan al cliente del objetivo.

RnF_26 La comunicación entre el servidor de aplicaciones y la base de datos se lleva a través del protocolo TCP/IP.

RnF_27 La comunicación entre el cliente y el servidor de aplicaciones se lleva a través del protocolo HTTPS.

RnF_28 El sistema debe comunicarse con los servicios web de pasarela y ldap de la Universidad de las Ciencias Informáticas mediante el protocolo SOAP durante la autenticación y para operaciones de seguridad.

Requisitos Legales, de Derecho de Autor y otros.

RNF_29 El sistema será sometido a un análisis legal por parte de los abogados y personal autorizado con vistas a declarar su autenticidad y evitar restricciones legales para su uso y comercialización documentando el resultado en el expediente del proyecto.

RNF_30 Se procederá a una evaluación y certificación por parte del cliente del producto documentando el resultado en el expediente del proyecto.

Estándares aplicables

RNF_31 Referirse al documento de arquitectura: CENIA_PRE_ADASP-v1.0 (en el mismo se especifica los requisitos de estándares aplicables) disponible en el expediente de proyecto.

2.3 Arquitectura propuesta

La arquitectura del *software* es el diseño de más alto nivel de la estructura de un sistema, programa o aplicación y tiene la responsabilidad de:

- Definir los módulos principales.
- Definir las responsabilidades que tendrá cada uno de estos módulos.
- Definir la interacción que existirá entre dichos módulos:
    - Control y flujo de datos.
    - Secuenciación de la información.
    - Protocolos de interacción y comunicación.
    - Ubicación en el hardware.

La arquitectura del *software* aporta una visión abstracta de alto nivel, proponiendo el detalle de cada uno de los módulos definidos a pasos posteriores del diseño. La definición oficial de arquitectura del *software* es la IEEE Std 1471-2000 que señala: “La arquitectura del *software* es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución.” (Casanovas, 2004)

2.3.1 Modelo arquitectónico Cliente/servidor

Es un modelo de sistema, que se organiza como un conjunto de servicios y servidores asociados, más unos clientes que acceden y usan los servicios. Los principales componentes de este modelo son un conjunto de servidores que ofrecen servicios a otros subsistemas, un conjunto de clientes que llaman a los servicios ofrecidos por los servidores y una red que permite a los clientes acceder a estos servidores. (Sommerville, 2005)

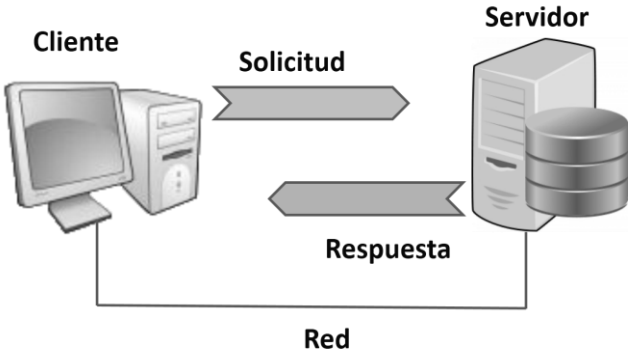


Figura 2. Estilo arquitectónico cliente/servidor

2.3.2 Patrón Modelo Vista Controlador

Como patrón arquitectónico se utilizó el Modelo-Vista-Controlador (MVC). Este separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y el controlador representa la lógica del negocio. (García Vidal, 2011)

El marco de trabajo GUUD es el utilizado actualmente en el centro, debido a que utiliza como patrón de arquitectura MVC, propone un modelo de diseño a través del cual deben regirse las aplicaciones que se desarrollen en el mismo.

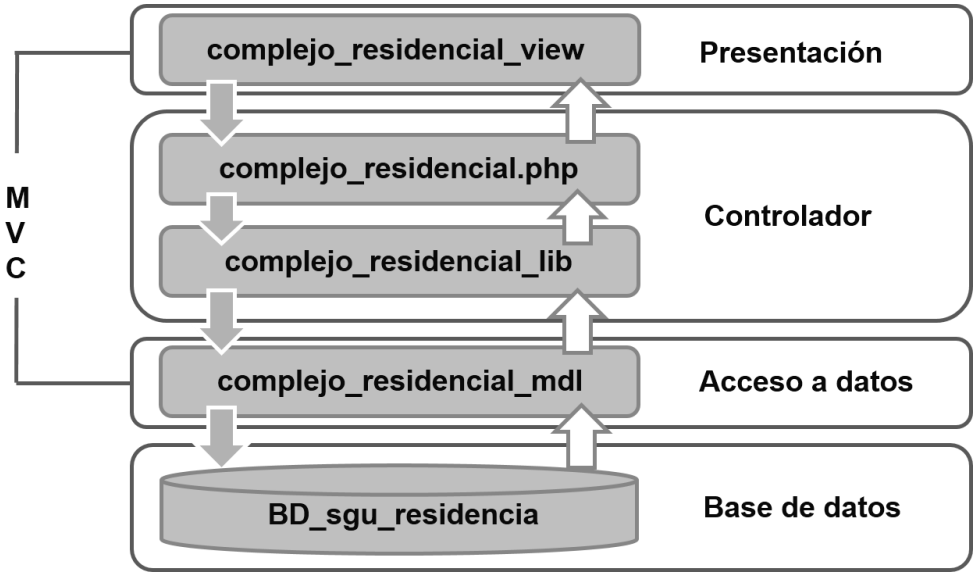


Figura 3. Patrón arquitectónico Modelo-Vista-Controlador implementado por GUUD

La estructura del MVC se pone de manifiesto en el núcleo, allí existe una carpeta base y dentro de esta la carpeta aplicaciones, donde se encuentran todos los subsistemas, separados cada uno por carpetas. Cada subsistema a su vez contiene carpetas con los módulos que conforman el subsistema. Todos los módulos están compuestos por 5 carpetas, con funciones diferentes:

- La carpeta *config* incluye la clase box donde se define las agrupaciones y funcionalidades que contendrá el módulo.
- La carpeta *controllers* incluye las clases que contienen las llamadas a los métodos que se encuentran en las librerías obteniendo a través de estas los datos, los organiza y luego son mostrados al usuario.
- En la carpeta *libraries* se encuentran todas las clases librerías por cada una de las controladoras, estas son las encargadas de la lógica del negocio, sirven de intermediarias entre la modelo y la controladora y obtienen información de otros módulos.
- La carpeta *models* es la encargada de guardar los datos en un medio persistente. Contiene las consultas de acceso a datos que encierran una serie de clases con métodos básicos predefinidos para el acceso a datos.
- La carpeta *views* es la encargada de la presentación de los datos. Contiene subcarpetas, cada una nombrada con distintas funcionalidades. Incluye a su vez las páginas HTML con las que interactúa el cliente, llamadas interfaces.

2.4 Patrones de diseño

Según Craig Larman, en su libro “UML y Patrones: Introducción al análisis y diseño orientado a objetos” un patrón es una pareja de problema/solución con un nombre y que es aplicable a otros contextos, con una sugerencia sobre la manera de usarlo en situaciones nuevas.(Larman, 1999)
Como se menciona anteriormente a un patrón se le asigna un nombre, teniendo esto como ventaja que apoya el agrupamiento y la incorporación del concepto al sistema cognitivo y a la memoria, además facilita la comunicación. Dentro de los patrones que se utilizan en el diseño están dos grupos fundamentales, los patrones conocidos como GRASP y GOF, teniendo cada una de estas agrupaciones varios patrones que permiten que se apliquen en la solución de un problema.

2.4.1 Patrones GRASP

Para la realización del sistema se utilizaron dentro de los patrones GRASP los patrones: Experto, Creador, Alta Cohesión, Bajo Acoplamiento y Controlador. A continuación se dará una breve descripción de estos patrones:
Experto: con este patrón se le asigna una responsabilidad a la clase que cuenta con la información necesaria para cumplirla. Aplicando este patrón, las librerías serán los expertos, debido a que, son las que cuentan con la información necesaria para cumplir responsabilidades sobre los elementos de negocio. Este patrón brinda beneficios como la conservación del encapsulamiento y el soporte de un bajo acoplamiento y una baja cohesión.

Bajo acoplamiento: este patrón estimula asignar una responsabilidad a una clase, de modo que su colaboración no incremente tanto el acoplamiento con otras clases, al nivel que produzca los resultados negativos propios de un alto acoplamiento. CodeIgniter tiene este patrón nivelado, pues permite el uso de los componentes de forma individual.

Alta cohesión: este patrón mejora la calidad y facilidad del diseño, genera un bajo acoplamiento y promueve la reutilización. En la propuesta de solución se evidencia cuando entre los componentes hay dependencia. La propia implementación de CodeIgniter contiene este patrón nivelado pues permite el uso de los componentes de forma individual, evidenciando la dependencia entre ellos o alta cohesión.

Controlador: este patrón ofrece una guía para tomar decisiones apropiadas en la elección de los controladores de eventos. Su utilización propicia que las operaciones del sistema se manejen en la capa de dominio de los objetos, y no en la de presentación. En el módulo Complejo Residencial las clases controladoras serán las que se encargan de obtener los datos, enviarlos a las librerías y a las vistas actuando así como el patrón controlador.

2.4.2 Patrones GOF

Los patrones GOF, describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases, la combinación de clases y la formación de estructuras de mayor complejidad. Permiten crear grupos de objetos para ayudar a realizar tareas complejas.

Los patrones GOF constituyen un catálogo de veintitrés patrones de diseño básicos publicados en el libro *Design Patterns: Elements of Reusable Object-Oriented Software*.(Gamma, Helm, et al., 2009)

Algunos de los patrones GOF empleados en la implementación del marco de CodeIgniter son *Singleton* (Instancia Única), *Observer* (Observador) y *Mediator* (Mediador) los que se describen a continuación:

Singleton: El empleo de este patrón permite asegurar que una clase posee solamente una instancia, proporcionando un punto de acceso global al objeto instanciado. En la investigación se evidencia en todas las clases controladoras que son instancias únicas.

Mediator: Como su nombre lo dice el uso de este patrón permite encontrar un mediador, en la propuesta de solución aplicando este patrón las librerías funcionan como mediadoras entre las clases controladoras y las modelos o acceso a datos.

Observer: En la clase *loader* que es el objeto *load* de las clases controladoras es donde se evidencia este patrón, esta clase es la que se encarga de cargar los elementos del marco de trabajo dígase, librerías, modelos y se encarga de actualizar la controladora instanciada.

2.5 Pautas de diseño visual

El módulo Complejo Residencial v2.0 al integrarse al Sistema de Gestión Universitaria es necesario que cumpla con las mismas pautas, estándares y diseño que posee el sistema. Trayendo como principal ventaja la uniformidad en la estructura de todas las páginas web, definiéndose un mapa de navegación y una taxonomía general para todo el módulo.

Vista de presentación: es la primera vista que se le muestra a cualquier usuario, mediante la que podrá autenticarse y acceder al sistema.

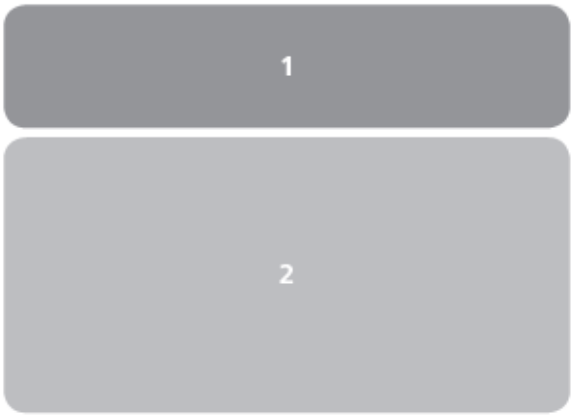


Figura 4. Vista de la presentación.

1 - Cabezal o área de identificación.

2 - Área de entrada de datos.

Vista de escritorio: se muestra luego de la autenticación del usuario, en la que podrá seleccionar el subproceso horizontal, módulo o servicio al que desee acceder y tenga los permisos requeridos.

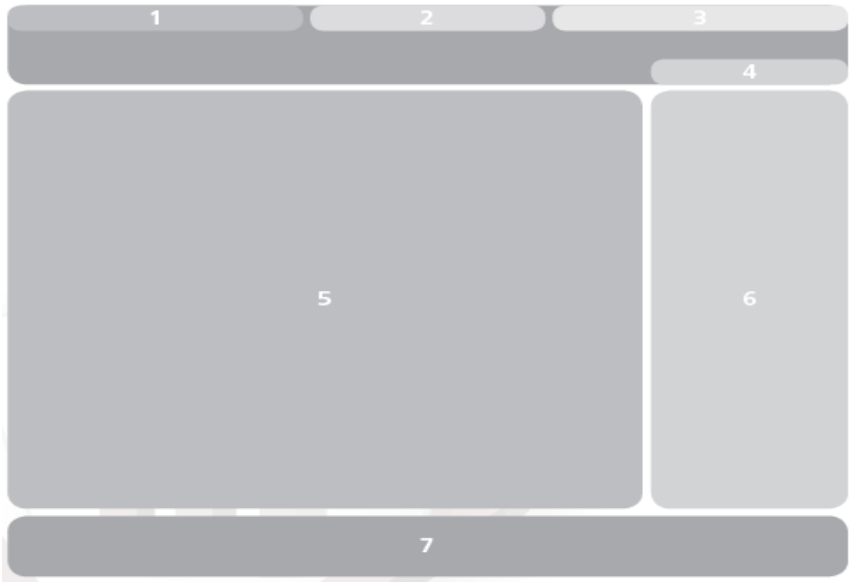


Figura 5. Vista de escritorio.

1- Área de subprocesos horizontales.
2- Área del nombre de la aplicación.
3- Área del buscador.
4- Área de nombre de usuario.
5- Área de líneas de procesos.
6- Área de servicios horizontales.
7- Área de pie de página.

Vista de gestión de procesos: permite el acceso a los módulos del subsistema seleccionado por el usuario y las funcionalidades que poseen.

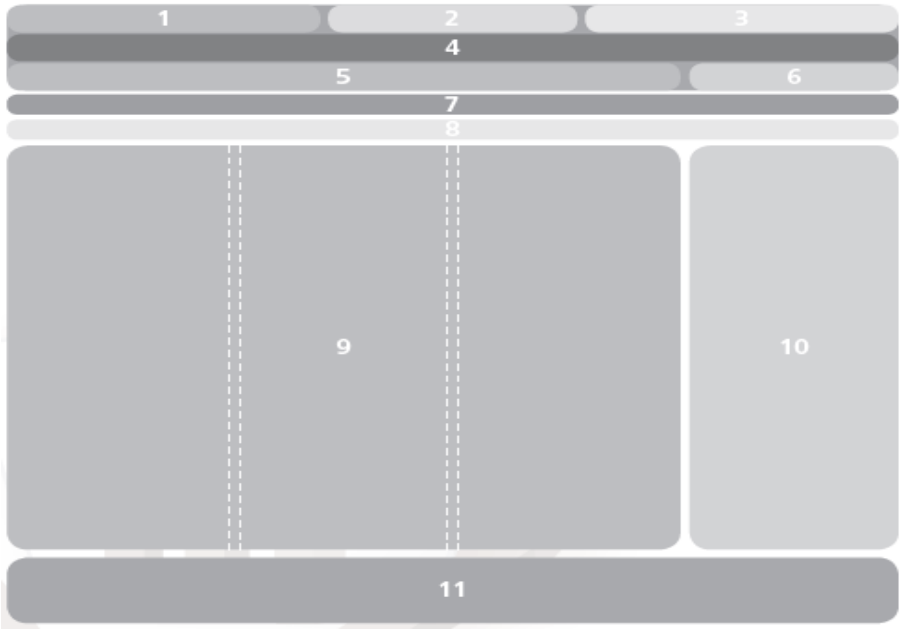


Figura 6. Vista de gestión de procesos

1- Área de subprocesos horizontales.
2- Área del nombre de la aplicación.
3- Área del buscador.
4- Área de líneas de procesos.
5- Área de módulos.
6- Área de nombre de usuario.
7- Área de traza.
8- Área de línea de progreso.
9- Área de contexto.
10- Área de menú de módulos.
11- Área de pie de página.

La selección de todas estas áreas va a validar los requisitos no funcionales de usabilidad. Debido a que, el usuario que interactúe con el sistema no tiene que acceder a tantos niveles para realizar la acción deseada.

2.5.1 Pauta cromática

Otro aspecto a tener en cuenta para el diseño son las pautas cromáticas a utilizar. Dentro de los requisitos no funcionales que se identificaron están los de interfaz, donde se especificó que los colores que se usarían para realizar el producto serían colores suaves a la vista del usuario. El Sistema de Gestión Universitaria propone un conjunto de colores que cumplen con estas condiciones, por lo que, para la realización de la propuesta de solución se adoptarán las pautas cromáticas definidas en la Figura 7.

Figura 7. Pautas cromáticas.

2.5.2 Componentes

Cumpliendo con los estándares definidos, se han seleccionado para la construcción del módulo varios componentes que van a facilitar su implementación y la uniformidad de todas las funcionalidades. Dentro de los componentes se encuentran:

- *Text Box*: componente que crea un control de entrada de una sola línea.
- *Text Area*: componente que crea un control de entrada de varias líneas.
- Combo Box Simple: componente que contiene una lista de opciones con menús desplegables que permiten elegir una de las múltiples opciones que posee.
- *Check Box*: componente para listar opciones, permitiendo la elección de más de un componente.
- *Picture Grid*: componente que permite la creación desde columnas con simples celdas de texto, hasta columnas con texto con formato, imágenes y dato específico. La diferencia radica en que en sus filas el primer elemento es una foto.

2.5.3 Mensajes

Además se definen los tipos de mensajes con los que contará el módulo. Dentro de los 3 tipos de mensajes definidos se encuentran: de información, de error y de confirmación. A continuación se describen cada uno de ellos:

Los mensajes de información para el cliente se utilizan cuando se crea un elemento o se modifica, dentro de estos mensajes están:

- El elemento ha sido creado satisfactoriamente.
- El elemento ha sido modificado satisfactoriamente.

Los mensajes de error se utilizan para mostrar al usuario cuando ha realizado una acción incorrecta. Estos mensajes pueden salir encima del componente o en forma de ventana.

En forma de ventana:

- El elemento ya existe.

Encima del componente:

- Campo requerido.
- Debe seleccionar una persona.
- Entre al menos n caracteres.

Los mensajes de confirmación son los que se utilizan cuando es necesario asegurarse que el usuario desea realizar una acción deseada, por ejemplo: cuando se va a cancelar una acción, es necesario asegurarse que eso es lo que desea. Dentro de los mensajes están:

- ¿Está seguro de realizar la acción?

2.6 Modelo de datos

La base de datos necesita una definición de su estructura que le permita almacenar datos, reconocer el contenido y recuperar la información. La estructura tiene que ser desarrollada de una manera eficiente para las aplicaciones que la usan. La puesta en práctica de la base de datos es el paso final

en el desarrollo de aplicaciones de soporte del negocio. Se conforman con los requisitos de los procesos del negocio, que es la primera abstracción de la vista de la base de datos.

La base de datos del SGU se encuentra dividida por esquemas para lograr una mayor integridad y seguridad de los datos. Uno de estos es el perteneciente al módulo Complejo Residencial v2.0, el cual cuenta con veintiuna tablas; once de datos, siete nomencladoras y tres de relaciones. La nomenclatura de dichas entidades se realiza de la siguiente forma: el modelo, el esquema está compuesto por el nombre sq_complejo_residencial donde sq representa el tipo. Las tablas de entidades o datos están compuestos por la nomenclatura tb_d+<texto>. Las tablas de los nomencladores se pueden identificar por la nomenclatura tb_n+<texto>. Las tablas de relación se identifican por la nomenclatura tb_r+<texto>. Las llaves primarias y foráneas tienen la nomenclatura id_+<texto>.

Con la inclusión de las nuevas funcionalidades la base de datos queda como se muestra en el modelo físico de datos en el **Anexo 3**.

Para la construcción de la misma, se aplicaron los siguientes patrones de base de datos con el objetivo de facilitar el trabajo y establecer un mayor control de los datos.

Modelo entidad-atributo-valor: es la representación de un modelo flexible donde se pueden representar objetos con sus atributos, es un acercamiento al modelo orientado a objeto representado en el modelo relacional, donde la entidad *Class* representa las clases, la entidad *Attribute* representa los atributos de las clases, por su parte la entidad *Object* representa las instancias de las clases, mientras que la entidad *Value* representa los valores de cada atributo para cada objeto dado. (Blaha, 2010)

Llaves subrogadas: Con este patrón se decide generar una llave primara única para cada entidad en vez de usar un atributo identificador en el contexto dado. Permite que las tablas sean más fáciles de consultar por el identificador, dado que se conoce el de cada una. (Blaha, 2010)

2.7 Modelo de despliegue

El modelo de despliegue provee un modelo detallado de la forma en la que los componentes se desplegarán a lo largo de la infraestructura del sistema. Detalla las capacidades de red, las especificaciones del servidor, los requisitos de hardware y otra información relacionada al despliegue del sistema propuesto. El diagrama de despliegue del módulo Complejo Residencial está compuesto por una computadora (PC cliente) que se conecta al servidor web a través de los protocolos HTTP/HTTPS y el servidor web a su vez se conecta al servidor de bases de datos mediante el protocolo TCP/IP. Sobre el modelo de despliegue deben hacerse las siguientes observaciones:

- Los nodos poseen relaciones que son medios de comunicación entre ellos, tales como TCP/IP, HTTP/HTTPS.

- HTTPS (Protocolo Seguro de Transferencia de Hipertexto) es un protocolo de red basado en HTTP por lo que está orientado a transacciones, sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores y sigue el esquema petición-respuesta entre un cliente

y un servidor. La principal diferencia entre ellos es que este está destinado a la transferencia segura de datos de hipertexto, en otras palabras, es la versión segura de HTTP.

- Cada nodo representa un recurso de cómputo, puede ser un procesador o un dispositivo hardware.

- PC cliente: computadora que cuenta con un navegador actualizado y que siga los estándares web (*Mozilla Firefox* v 2.x o superior). Se recomiendan estaciones de trabajo con sistema operativo GNU/Linux.

- Servidor web: representa una estación donde estará montado el Servidor *Apache* sobre el cual correrá la aplicación.

- Servidor de base de datos: representa el servidor donde estará el Sistema Gestor de Bases de Datos Postgres que dará respuesta a las peticiones hechas por la aplicación.

En la siguiente figura, se modela la distribución física de los nodos necesarios para la implantación de dicho sistema y por ende del módulo Complejo Residencial.

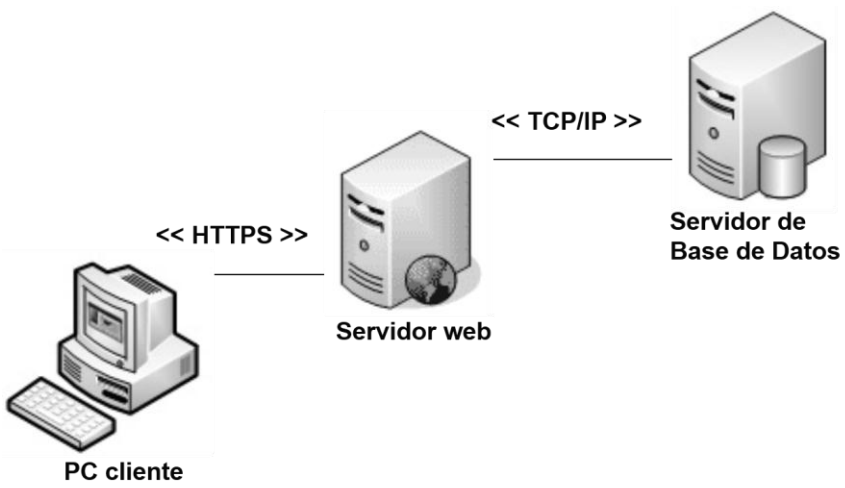


Figura 8. Modelo de despliegue.

Conclusiones

En este capítulo se realizó un profundo análisis del proceso de negocio que interviene en el Complejo Residencial, permitiendo obtener el conjunto de funcionalidades que debe cumplir el módulo. Se propuso la arquitectura a utilizar, los patrones de diseño y el modelo de datos. Estos facilitarán que se cuente con los componentes y elementos necesarios para realizar la implementación. Además quedaron descritos los requisitos no funcionales que apoyarán para el desarrollo del sistema. Las funcionalidades descritas en la propuesta de solución posibilitarán que al implementarse las mismas los elementos de control del Complejo Residencial puedan ser configurables y se logre una mejor administración.

CAPÍTULO 3: CONSTRUCCIÓN Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA.

Introducción

En el proceso de desarrollo de *software* luego de especificar y diseñar el producto se pasa a la implementación. Se procede a realizar el proceso de pruebas que no deja de ser importante en el desarrollo de un sistema, debido a que permite conocer la calidad del mismo. En el presente capítulo se incluyen los estándares y técnicas de codificación y el tratamiento de errores. Además, contiene la estrategia de prueba seleccionada para probar la propuesta de solución.

3.1 Técnicas de programación

Cuando se crea un producto el mismo debe tener la flexibilidad suficiente para ser modificable en el momento que se requiera. Estos deben ser claros, simples, con el fin de poder ser leídos e interpretados de forma fácil. Para lograr este objetivo se deben asumir en la programación técnicas que permitan la estandarización (Pressman, 2010).

En las últimas décadas con el desarrollo de la Informática es muy común escuchar que los diversos lenguajes y programas que se crean están orientados a objetos y que soportan la programación modular (Pressman, 2010).En los siguientes epígrafes se explicarán estas dos técnicas que son adoptadas para la creación del módulo.

3.1.1 Programación modular

La programación modular es uno de los métodos para el diseño más flexible y de mayor rendimiento para la productividad de un programa. En este tipo de programación el programa es dividido en módulos, cada uno de los cuales realiza una tarea específica, codificándose independientemente de otros módulos. Cada uno de estos es analizado, codificados y puestos a punto por separado (Bonanata, 2003).

Se propone utilizar esta técnica de programación debido a la estructura que tiene el Sistema de Gestión Universitaria, que se divide en módulos, posibilitando que la seguridad de los mismos sea independiente. Además, esta técnica posibilita que haya diferentes programadores trabajando a la vez sobre el producto. También permite que un módulo pueda ser codificado sin afectar a los demás, incluso sin alterar su función principal.

3.1.2 Programación orientada a objetos (POO)

En los últimos años la frase “orientado a objetos”, se ha vuelto muy popular, escuchándose a cada momento frases como: “sistemas operativos orientados a objetos”, “lenguajes orientado a objetos” y “programación orientado a objetos”. Dentro de los conceptos generales más utilizados en el modelo orientado a objeto se encuentran: la abstracción, la encapsulación y modularidad. Y con respecto a la

programación son: los objetos, las clases, los métodos, el envío y recepción de mensajes, la herencia y el polimorfismo (Bonanata, 2003).

En la POO encapsular significa que se reúne y controla todo el grupo resultante en un conjunto y no de forma individual. La abstracción es un término externo al objeto, que controla la forma en que es visto por los demás. La herencia se define como una jerarquía de clases derivadas y la relación entre estas, donde se comparte la estructura y el comportamiento de una o más clases consideradas como clases padres. El polimorfismo constituye la definición de múltiples clases con funcionalidades diferentes, pero con métodos o propiedades denominados de forma idéntica.

3.2 Estándares de codificación

Un estándar según la Real Academia Española es: un modelo, norma, patrón o referencia. Especificación que regula la realización de ciertos procesos o la fabricación de componentes para garantizar la interoperabilidad. Para el SGU se estandarizó al igual que el diseño, el código, permitiendo de esa forma que exista legibilidad y organización para su mantenimiento. A continuación se especifican los estándares de código para la construcción de la segunda versión del módulo.

3.2.1 Conversión de nomenclaturas

Variables locales:

- Los nombres de las variables se rigen por la nomenclatura *CamelCase*, específicamente por el tipo *lowerCamelCase* que es cuando la primera letra de cada palabra se escribe con mayúscula a excepción de la primera palabra que se escribe con minúscula.

Ejemplo:

```
$variable;
$variableNombreCompuesto;
```

Los nombres de algunas variables locales, como los iteradores o los contadores, pueden especificarse en minúscula y de forma abreviada, siempre que su contexto sea específicamente local y su lectura sea intuitiva.

Ejemplos:

```
$cont, $i, $j.
```

Variables globales (constantes): los nombres de variables globales deben ser siempre en mayúsculas, separando las palabras con guiones bajos (“_”).

Ejemplo:

```
define(CONSTANTE,valor);
define(CONSTANTE_COMPUESTO,valor);
```

Clases: El nombre debe ser descriptivo, evitando abreviaciones y siempre comienzan con mayúscula. En caso de nombres compuestos, se usará el caracter subrayado “_” para separar las palabras.

Ejemplo:

```
class Clase {
        //Bloque de instrucción
}
```

```
class Clase_nombre_compuesto {
        //Bloque de instrucciones
}
```

Funciones: se rigen por la nomenclatura *camelCase*. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula. Los parámetros son separados por espacio luego de la coma que los separa.

```
function funcion($param1, param2 = array())
{
        // Bloque de instrucciones
}
function funcionNombreCompuesto($param1, param2 = array())
{
        // Bloque de instrucciones
}
```

Ficheros: todo siempre en minúscula y en caso de nombres compuestos se usa el carácter subrayado "_".

- **Vistas:** intuitivo y relacionado con el formulario y/o vista que representa, contiene en el nombre el sufijo _view.
- **Modelos:** con el mismo nombre de la clase que representa, contiene en el nombre el sufijo _mdl.
- **Librerías:** con el mismo nombre de la clase que representa, contiene en el nombre el sufijo _lib.
- **Controladoras:** con el mismo nombre de la clase que representa .

3.2.2 Etiquetas de bloque PHP

Siempre utilizar **<?php** y **?>** para iniciar y terminar un bloque de código PHP, no las variantes **<?** y **?>** o **<%** y **%>**. Esto asegura compatibilidad entre diversas configuraciones de equipos.

3.2.3 Estructuras de control

- Incluye *if, for, while, switch, foreach*. Deben tener un espacio entre la palabra clave y el paréntesis de apertura, para diferenciarlos de las llamadas a funciones. Se recomienda utilizar

siempre llaves de apertura y cierre, incluso en situaciones en las que técnicamente son opcionales.

- Si las condiciones son muy largas que sobrepasan el tamaño de la línea, éstas se dividen en varias líneas. En el mejor de los casos cuando la condición es muy extensa, se puede dividir en variables y compararlas dentro de la estructura de control.

Ejemplo:

```
if ($condicion)
{
        // Bloque de instrucciones
}
else
{
        // Bloque de instrucciones
}
```

3.2.4 Documentación

Todos los archivos deben de tener la documentación asociada al mismo. Para esto debe de cumplir con el siguiente bloque al principio de cada clase.

Ejemplo:

```
/**
* Breve descripción de la clase
* @package nombre del paquete o módulo al que pertenece
* @subpackage nombre del subpaquete (si es el caso)
* @category categoría de la clase (controladora, librería, modelo)
* @author nombre y apellidos del autor (se puede agregar el e-mail)
*/
```

De la misma forma las funciones de cada clase deberán ser documentadas como se muestra a continuación.

```
/**
* Breve descripción de la función
* @param tipo de dato y nombre del parámetro (por cada uno)
* @return tipo de dato que retorna
* @author nombre y apellidos del autor (se puede agregar el e-mail)
*/
```

3.2.5 Comentarios

Se aconseja el uso de comentarios en línea para facilitar la comprensión del código, sobre todo en procedimientos complejos. Los comentarios pueden ser con fin documental o bien como “ayuda-memoria”. Para ello se recomienda utilizar los estilos de C (/* */) y C++ (//).

3.3 Tratamiento de errores

En la construcción de un *software* se debe tener en cuenta el tratamiento de errores que no es más que la acción de identificar, localizar, analizar y eliminar los errores que se producen en este proceso. A menudo, cuando se termina un producto, este no tiene un correcto funcionamiento, debido a que, a la hora de entrar los datos en el sistema no hay estabilidad, produciendo errores que atentan contra la calidad.

El tratamiento de errores que deben poseer los sistemas automatizados deben estar encaminados tanto a los errores que se producen a la hora de los usuarios introducir datos, como a los errores que puedan ser generados por el comportamiento incorrecto de los componentes internos, por ejemplo: las consultas de base de datos.

En el módulo Complejo Residencial, los datos que introduce el usuario son validados en los formularios del jQuery mediante JavaScript, mostrando mensajes de error encima de los componentes que indican al usuario cuáles datos están incorrectos o cuáles datos son obligatorios. En cuanto al tratamiento de errores en los componentes internos, se implementan en las controladoras las validaciones y cuando se detecta una posible excepción se usan las definidas en un archivo XML.

3.4 Validación de la solución propuesta

Durante el desarrollo de un sistema se pueden presentar errores en el mismo, por lo que la validación de la propuesta de solución es esencial para el desarrollo saludable de una futura aplicación. La estrategia de prueba del *software* debe ser lo suficientemente flexible como para promover un enfoque personalizado; y al mismo tiempo, debe ser adecuadamente rígido como para promover una planeación razonable y un seguimiento administrativo del avance del proyecto. La recomendación de pruebas a realizarle al sistema luego de su implementación e integración al Sistema de Gestión de Residencia está encaminada a minimizar la cantidad de posibles errores a cometer. En los siguientes epígrafes se podrán encontrar las validaciones que se le hicieron a los requisitos y la estrategia de prueba a utilizar.

3.4.1 Validación de los requisitos

La validación de requisitos se define como: un proceso o una reunión durante la cual los requisitos de *software* son presentados a clientes, gerentes u otras partes interesadas para hacer comentarios o aprobación (IEEE, 2012). Dentro de los parámetros a comprobar en la validación están:

- La validez: donde no basta con preguntar a un solo usuario, todos los potenciales usuarios deben tener puntos de vista distintos y necesitar otros requisitos.

- La consistencia: no debe haber contradicciones entre unos requisitos y otros.
- La completitud: deben estar todos los requisitos.
- El realismo: se pueden implementar con la tecnología actual.
- La verificabilidad: tiene que existir alguna forma de comprobar que cada requisito se cumple.

En el proceso de desarrollo se definen varios documentos para la validación de requisitos que permiten identificar los criterios para validar los requisitos del cliente y del producto. En la Tabla 5 se pueden encontrar las preguntas que se realizan para validar los requisitos en el artefacto de "Criterios para validar requisitos del cliente".

Tabla 5. Criterios para validar requisitos del cliente

Criterios
¿El proveedor del requisito es un proveedor válido?
¿El requisito está identificado como único?
¿El requisito es modificable?
¿El requisito no es ambiguo?
¿El requisito está completo?
¿El requisito es congruente con otros requisitos relacionados?
¿El requisito puede ser implementado?
¿El requisito puede ser probado?
¿El resultado de la evaluación de impacto es positivo?
¿El requisito está correcto?
¿El requisito es traceable?

Además de validar los requisitos, se realizaron revisiones a los requisitos funcionales y no funcionales mediante la técnica de revisión de requisitos donde se llevaron a cabo reuniones entre varios miembros del proyecto con el fin de revisar el documento de "Especificación de requisitos". Este proceso abarcó los aspectos que se mencionan a continuación:

- Verificaciones de validez: los requisitos deben cumplir con las necesidades del cliente.
- Verificaciones de consistencia: los requisitos no deben contradecirse en las especificaciones escritas, no deben existir restricciones o descripciones opuestas a las reglas definidas.
- Verificaciones de completitud: los requisitos deben incluir todas las funcionalidades propuestas por el cliente, satisfacer de manera general todas las necesidades acordadas.

3.5 Pruebas

La construcción de un *software* tiene como objetivo satisfacer una necesidad planteada por un cliente. Pero ¿cómo se puede saber si el producto construido se corresponde exactamente con lo que el cliente les pidió y funciona correctamente? Por ese motivo es que se hace necesario llevar a cabo, en paralelo al proceso de desarrollo, un proceso de evaluación o comprobación de los distintos productos o modelos que se van generando. (Juristo y Vegas, 2005)

Para dicho proceso se pueden usar distintos tipos de técnicas. En general, éstas se agrupan en dos categorías: Técnicas de Evaluación Estáticas y de Evaluación Dinámicas. Las pruebas de *software* son

los procesos que permiten verificar y revelar la calidad de un producto. Son utilizadas para identificar posibles fallos de implementación, calidad o usabilidad de un programa. Básicamente es una fase en el desarrollo de *software* consistente en probar las aplicaciones construidas. Las pruebas de *software* se integran dentro de las diferentes fases del ciclo de vida del proyecto dentro de la Ingeniería de *software*. Para comprobar que la aplicación cumple con los requisitos funcionales definidos, es necesario realizarle pruebas antes de dar por terminado su proceso de implementación. El propósito de éstas es simular una carga de producción real y observar cómo se comporta el sistema bajo cargas intensivas. Esto permite solucionar los problemas de rendimiento, antes de poner la aplicación en marcha. (Juristo y Vegas, 2005)

3.5.1 Niveles y técnicas de prueba

El proceso de evaluación de un *software* debe permitir comenzar por los componentes más simples y más pequeños e ir avanzando progresivamente hasta probar todo el *software* en su conjunto. (Juristo y Vegas, 2005)

Para ello se cuenta con tres pasos fundamentales que se implementan de manera secuencial, los mismos son representados en la figura.

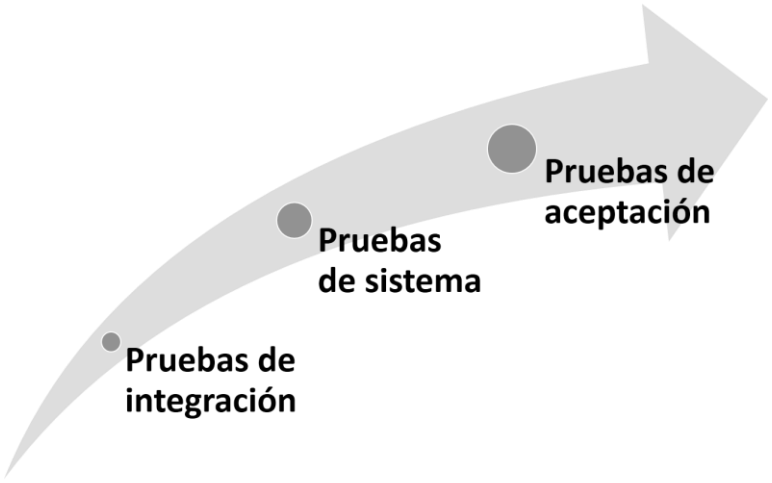


Figura 9. Niveles de prueba

Pruebas de integración

Las pruebas de integración se centran en probar la coherencia semántica entre los diferentes módulos, tanto de semántica estática (se importan los módulos adecuados; se llama correctamente a los procedimientos proporcionados por cada módulo), como de semántica dinámica (un módulo recibe de otro lo que esperaba). Normalmente estas pruebas se van realizando por etapas, englobando progresivamente más y más módulos en cada prueba. Las pruebas de integración se pueden empezar en cuanto tenemos unos pocos módulos, aunque no terminarán hasta disponer de la totalidad. En un diseño descendente (*top-down*) se empieza a probar por los módulos más generales; mientras que en un diseño ascendente se empieza a probar por los módulos de base. (Pressman, 2010)

Pruebas del sistema

Este tipo de pruebas tiene como propósito ejercitar profundamente el sistema para verificar que se han integrado adecuadamente todos los elementos del sistema (hardware, otro *software*, etc.) y que se realizan las funciones adecuadas. Concretamente se debe comprobar que:

- Se cumplen los requisitos funcionales establecidos.
- El funcionamiento y rendimiento de las interfaces de hardware, *software* y de usuario.
- La adecuación de la documentación de usuario. (Juristo y Vegas, 2005)

Pruebas de aceptación

En las pruebas de aceptación su objetivo es la evaluación del producto y la realización de una revisión de la documentación final. Estas pruebas las realiza el cliente. Son básicamente pruebas funcionales, sobre el sistema completo, y buscan una cobertura de la especificación de requisitos y del manual del usuario. Estas pruebas no se realizan durante el desarrollo, pues sería impresentable de cara al cliente; sino una vez pasadas todas las pruebas de integración por parte del desarrollador.

La experiencia muestra que aún después del más cuidadoso proceso de pruebas por parte del desarrollador, quedan una serie de errores que sólo aparecen cuando el cliente se pone a usarlo. Muchos desarrolladores ejercitan unas técnicas denominadas "pruebas alfa" y "pruebas beta". Las pruebas alfa consisten en invitar al cliente a que venga al entorno de desarrollo a probar el sistema. Se trabaja en un entorno controlado y el cliente siempre tiene un experto a mano para ayudarle a usar el sistema y para analizar los resultados. Las pruebas beta vienen después de las pruebas alfa, y se desarrollan en el entorno del cliente, un entorno que está fuera de control. Aquí el cliente se queda a solas con el producto y trata de encontrarle fallos (reales o imaginarios) de los que informa al desarrollador.

3.5.2 Métodos de prueba

Prueba de caja blanca

Método de diseño que usa la estructura de control descrita como parte del diseño al nivel de componentes para derivar los casos de prueba. Al emplear los métodos de prueba de caja blanca, el ingeniero del *software* podrá derivar casos de prueba que:

- Garanticen que todas las rutas independientes dentro del módulo se han ejercitado por lo menos una vez.
- Ejerciten los lados verdadero y falso de todas las decisiones lógicas.
- Ejecuten todos los bucles en sus límites y dentro de sus límites operacionales.
- Ejerciten estructuras de datos internos para asegurar su validez.

Prueba de caja negra

Se centra principalmente en los requisitos funcionales del *software*. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

La prueba de caja negra no es una alternativa a las técnicas de prueba de la caja blanca, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados en los métodos de la caja blanca. Muchos autores consideran que estas pruebas permiten encontrar:

- ✓ Funciones incorrectas o ausentes.
- ✓ Errores de interfaz.
- ✓ Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- ✓ Errores de rendimiento.
- ✓ Errores de inicialización y terminación.

Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están:

1. *Técnica de la Partición de Equivalencia:* esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del *software*.
2. *Técnica del Análisis de Valores Límites:* esta Técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
3. *Técnica de Grafos de Causa-Efecto:* es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Dentro del método de Caja Negra la técnica de la *Partición de Equivalencia* es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el *software*, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así en número de clases de prueba que hay que desarrollar. (Pressman, 2010)

3.5.3 Estrategia de prueba

Según Pressman una estrategia de prueba del *software*: integra los métodos de diseño de caso de pruebas en una serie bien planeada de pasos que desembocará en la eficaz construcción del *software*. La estrategia proporciona un mapa que describe los pasos, además de cuánto esfuerzo, tiempo y recursos consumirán. (Pressman, 2010)

Durante el proceso de desarrollo del módulo se llevaron a cabo diferentes pruebas mediante el uso del método de caja negra, proyectando resultados visibles. Se emplearon además, pruebas de estrés y de aceptación, las que se describen más adelante. Para los efectos de esta investigación se propone la siguiente estrategia de prueba:

Tabla 6. Estrategia de prueba aplicada a la solución

Nivel de prueba	Tipo de prueba	Método de prueba	Técnica de prueba
Integración	Estructura, Seguridad	Caja negra	Incremental

Sistema	Resistencia, Rendimiento	Caja negra	Automática
	Funcional	Caja negra	Particiones de equivalencia
Aceptación	Funcional	Caja negra	Alfa, Beta

3.5.4 Aplicación y resultado de las pruebas

Pruebas de integración

Como en la investigación la solución informática que se propone es un módulo que forma parte del Sistema de Gestión de Residencia y a la vez del Sistema de Gestión Universitaria, es necesario probar la integración entre las agrupaciones funcionales, además su integración con los módulos de los que se obtiene información.

Técnica incremental

Los autores utilizan los casos de prueba para probar los elementos mencionados anteriormente. Para cada elemento se creó un caso de prueba donde se recogen todos los datos necesarios. Dentro de estos datos se encuentran:

- *Número de caso de prueba**:*** corresponde al nombre que se le da al caso de prueba está identificado por las siglas ***Int_n***, donde ***Int*** corresponde a la identificación de las pruebas de integración y *n* es un número consecutivo que se le asigna.
- *Funcionalidad o módulo a integrar:* corresponde al nombre del elemento a probar.
- *Condiciones de ejecución:* hace referencia a los pasos a seguir antes de realizar la prueba.
- *Descripción de la prueba:* hace referencia a la descripción de la prueba.
- *Entradas/Pasos de ejecución:* hace referencia a los pasos que se deben realizar para ejecutar la prueba.
- *Resultado esperado:* es el resultado que se espera cuando se ejecutan los pasos.
- *Evaluación:* hace referencia al resultado de la prueba luego de ser ejecutada, donde el resultado puede ser satisfactorio o insatisfactorio.

En las tablas 7 y 8 se muestran ejemplos de casos de prueba de integración a funcionalidades y las tablas 9, 10 y 11 a módulos con los que interactúa. Los demás casos de prueba generados se pueden encontrar en el **Anexo 6**.

Tabla 7. Caso de prueba de integración Int_1

Número de caso de prueba: Int_1	
Funcionalidad a integrar:	Moneda
Condiciones de ejecución:	Se debe crear la moneda.
Descripción de la prueba:	Comprobar que cuando se le asigne el precio a un servicio y se llenen los datos, se liste la moneda.
Entrada/Pasos de ejecución:	El usuario se dirige a la agrupación funcional Configuración, accede a la funcionalidad Moneda y escoge la opción de crear en el área de íconos flotantes. Llena los

	datos del servicio y escoge la moneda en la que será cobrado
Resultado esperado:	Se lista la moneda
Evaluación:	Satisfactoria

Tabla 8. Caso de prueba de integración Int_2

Número de caso de prueba: Int_2	
Funcionalidad a integrar:	Tipo de incidencia
Condiciones de ejecución:	Se debe crear el tipo de incidencia: pérdida de llave.
Descripción de la prueba:	Comprobar que cuando aparezca la interfaz para crear una incidencia se pueda escoger el tipo de incidencia: perdida de llave.
Entrada/Pasos de ejecución:	El usuario se dirige a la agrupación funcional Huésped, accede a la funcionalidad Incidencia, escoge la opción de crear en el área de íconos flotantes. Cuando aparezca la interfaz se debe seleccionar el tipo de incidencia pérdida de llave.
Resultado esperado:	Se lista el tipo de incidencia: pérdida de llave
Evaluación:	Satisfactoria

Tabla 9. Caso de prueba de integración Int_3

Número de caso de prueba: Int_3	
Módulo a integrar:	Inmuebles
Condiciones de ejecución:	Deben existir inmuebles designados al Complejo Residencial creados en el módulo.
Descripción de la prueba:	Comprobar que cuando aparezca la interfaz donde se asignan medios básicos, se listen los inmuebles.
Entrada/Pasos de ejecución:	El usuario se dirige a la agrupación funcional Configuración y accede a la funcionalidad Asignar medios básicos. Cuando aparezca la interfaz se deben listar los apartamentos que corresponden al Complejo Residencial.
Resultado esperado:	Se listaron los inmuebles pertenecientes al Complejo Residencial.
Evaluación:	Satisfactoria

Tabla 10. Caso de prueba de integración Int_4

Número de caso de prueba: Int_4	
Módulo a integrar:	Solicitudes
Condiciones de ejecución:	Deben existir solicitudes de alojamiento para el Complejo Residencial aprobadas.
Descripción de la prueba:	Comprobar que cuando aparezca la interfaz donde se listan las solicitudes se pueda seleccionar una para ubicar.

Entrada/Pasos de ejecución:	El usuario se dirige a la agrupación funcional Reservación y accede a la funcionalidad Solicitudes. Cuando aparezca la interfaz se deben listar las solicitudes aprobadas que corresponden al Complejo Residencial.
Resultado esperado:	Se listaron las solicitudes aprobadas pertenecientes al Complejo Residencial.
Evaluación:	Satisfactoria

Tabla 11. Caso de prueba de integración Int_3

Número de caso de prueba: Int_3	
Módulo a integrar:	Personal
Condiciones de ejecución:	Se realiza la petición de los datos.
Descripción de la prueba:	Comprobar que el módulo Personal proporcione la información de las personas de la universidad.
Entrada/Pasos de ejecución:	El usuario se dirige a la agrupación funcional Reservación, accede a la funcionalidad Entrada y escoge la opción de crear en el área de íconos flotantes. Cuando aparezca la interfaz para buscar una persona se listan los datos que ofrece dicho módulo.
Resultado esperado:	Se listaron las personas que están registradas en la base de datos.
Evaluación:	Satisfactoria

Se comprobó la correcta distribución de responsabilidades del módulo Seguridad, del módulo Personal se obtuvo satisfactoriamente la información de las personas registradas en la base de datos. Así mismo se demostró la correcta integración con los módulos Inmuebles, Eventos y Solicitudes y las funcionalidades propias del sistema.

Prueba funcional

En la propuesta de solución se aplicó el método de caja negra con la técnica de casos de prueba en donde se intenta arruinar el sistema con todas las posibles combinaciones de datos y su resultado radica en tener una alta probabilidad de mostrar un error no descubierto hasta el momento.

Para la aplicación de esta técnica se generaron los artefactos de “Diseño de caso de prueba basado en requisitos”. Por cada requisito funcional se creó un documento en donde se recogen todos los datos necesarios para probar la interfaz (ver **Anexo 5**). Dentro de esos datos se encuentran:

- *Condiciones de ejecución*: contiene como su nombre lo indica todas las condiciones que deben existir para que el caso de prueba sea ejecutado sin dificultad.
- *SC_ Nombre del requisito a probar*: corresponde al nombre que se le da al requisito en el documento de “Especificación de requisitos”.

- *Escenario*: en este campo se nombran los diferentes escenarios a probar donde cada uno estará compuesto por las siglas ***EC*** y una numeración consecutiva 1.1… 1.n + el nombre del escenario.
- *Descripción*: corresponde como dice su nombre a la descripción del escenario a probar.
- *Variable 1…n*: este campo va a estar compuesto por dos partes: la primera podrá tomar los valores V, I y NA (válido, inválido y no aplica respectivamente), la segunda estará conformada por un ejemplo según el escenario que se pretende realizar.
- *Respuesta del sistema*: se escribe el resultado que se espera al realizar la prueba.
- *Flujo central*: corresponde a los pasos a desarrollar para probar la funcionalidad que se indicó.

Como resultado de las pruebas realizadas en todo el ciclo de vida de la solución se ha logrado obtener una aplicación segura y confiable. Para ejecutar los diseños de casos de prueba generados se definió un plan de iteraciones donde se obtuvieron un grupo de no conformidades, las que se establecen en la gráfica a continuación. En la primera iteración con veintinueve requisitos implementados se encontraron veinticinco no conformidades. Para la segunda iteración con el total de los requisitos implementados se hallaron treinta y seis no conformidades. Y en una tercera y última iteración se obtuvieron doce no conformidades.

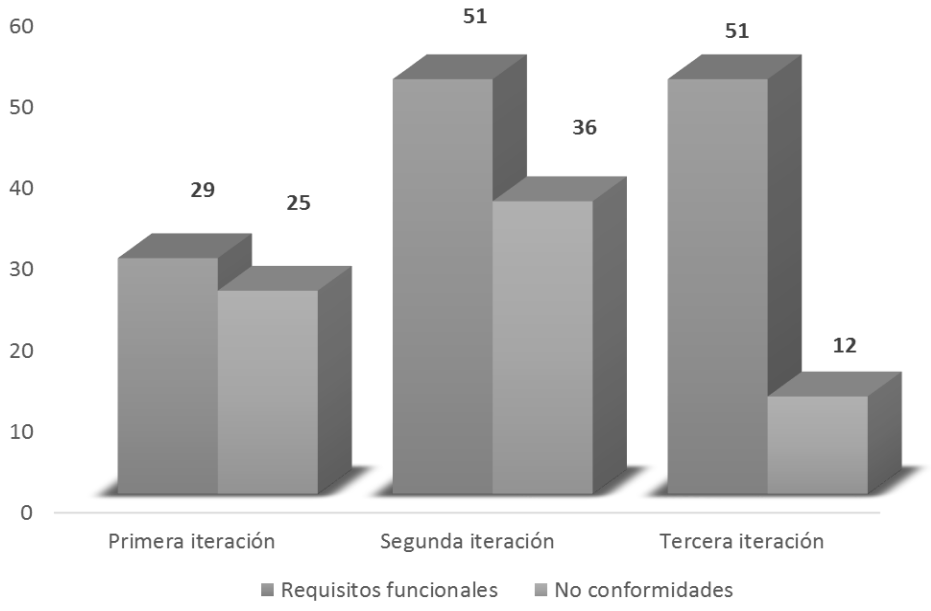


Figura 10. No conformidades encontradas por iteraciones.

Mediante estas pruebas se detectaron algunos errores funcionales y de validación, asociados a las respuestas del sistema. Todos estos casos de pruebas fallidos fueron mitigados en cada una de las iteraciones. Finalmente no se detectaron no conformidades dando paso a una probada solución.

Prueba de carga

Mediante esta prueba se ve la reacción del sistema con picos de mayor volumen en un tiempo limitado, puede ser en situaciones que nunca ocurran o que puedan eventualmente ocurrir. Para realizar las

pruebas de carga se utilizó la herramienta *Jmeter* que arrojó valiosos datos sobre el rendimiento del sistema.

En la siguiente figura se hace referencia al tiempo que demora el sistema en responder a la simulación de cinco peticiones realizadas a la funcionalidad de registrar una solicitud sin reservación.

# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento
5	1261	1384	1462	693	1741	0,00%	2,0/sec
5	1520	1620	1623	1253	1692	0,00%	1,4/sec
5	1684	1702	1708	1624	1748	0,00%	1,3/sec
5	1836	1839	1967	1588	1978	0,00%	1,3/sec
5	1211	1219	1229	1094	1356	0,00%	1,5/sec
5	16	13	16	11	29	0,00%	2,4/sec
5	18	19	21	9	30	0,00%	2,4/sec
5	7	6	8	2	13	0,00%	2,4/sec
5	1208	1144	1277	1094	1399	0,00%	1,6/sec
5	4693	4780	4864	4396	4931	0,00%	45,9/min
50	1345	1253	1978	2	4931	0,00%	3,5/sec

Figura 11. Tabla de datos ofrecida por la herramienta Jmeter.

Análisis de resultados

Los resultados expuestos en la Figura 11 se pueden interpretar de la siguiente forma:

- *#Muestras:* cantidad de hilos utilizados para la URL.
- *Media:* tiempo promedio en milisegundos para un conjunto de resultados.
- *Mediana:* valor en tiempo del percentil 50.
- *Línea de 90%:* máximo tiempo utilizado por el 90% de la muestra.
- *Min:* tiempo mínimo de la muestra de una determinada URL.
- *Max:* tiempo máximo de la muestra de una determinada URL.
- *%Error:* porcentaje de requerimientos con errores.
- *Rendimiento:* rendimiento medido en los requerimiento por segundo / minuto / hora.

Se obtuvo el tiempo promedio para acceder a una página mediante el siguiente cálculo:

Tiempo Total= #Muestra * Media.

Dio como resultado 1,345 segundos, realizándose un total de cincuenta requerimientos al servidor. El tiempo promedio total requerido por cada hilo se alcanzó mediante la fórmula:

((Tiempo Total/1000)/60)/cantidad de hilos.

El mismo fue de 0,22416 minutos. Se puede asumir que el resultado obtenido es satisfactorio, lo que significa que el sistema es capaz de soportar mucho más carga de la que se deberá enfrentar en el peor de los casos, ya que será usado por un pequeño número de usuarios.

Pruebas de aceptación

Estas pruebas las realiza el cliente. Su objetivo es la evaluación del producto y la realización de una revisión de la documentación final, se realiza en presencia del cliente final. Son básicamente pruebas funcionales, sobre el sistema completo, y buscan una cobertura de la especificación de requisitos. Se invita al cliente al entorno de desarrollo a probar el sistema. Se trabaja en un entorno controlado y el

cliente siempre tiene un experto a mano para ayudarle a usar el sistema y para analizar los resultados. Estas pruebas sirven para ir entrenando con antelación al personal que va a trabajar con el sistema.

Para las pruebas de aceptación se realizaron dos iteraciones como se muestra en la Figura 11, en la primera iteración de pruebas de veintinueve requisitos funcionales implementados se identificaron quince no conformidades a las cuales se le dio solución a su totalidad. En la segunda iteración de pruebas de cincuenta y una funcionalidades se registraron seis no conformidades, a las cuales fueron mitigadas en su totalidad.

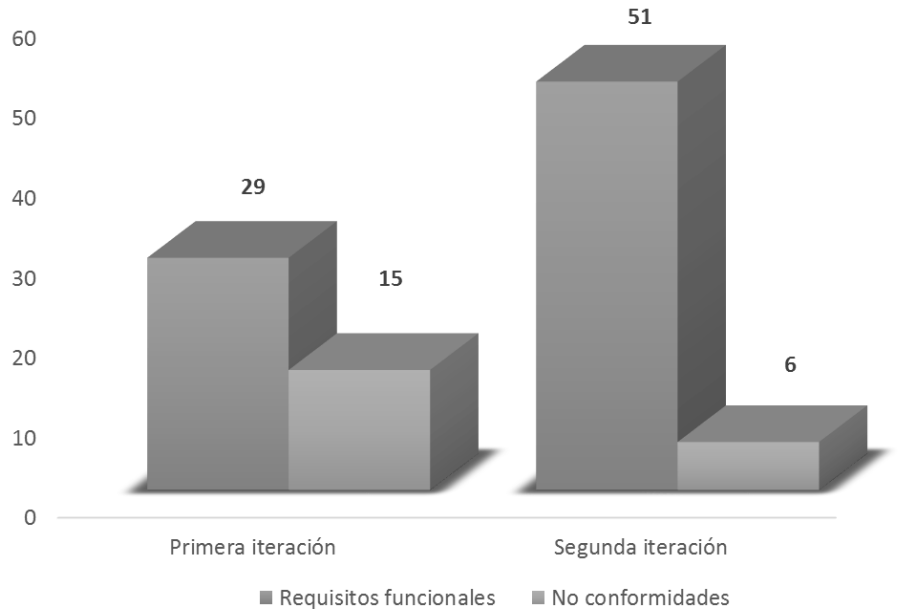


Figura 12. Resultados de las pruebas de aceptación.

Después de haber finalizado las pruebas realizadas se concluyó que el resultado fue satisfactorio. Conjuntamente con este resultado se obtuvo la aprobación de la solución propuesta por parte del cliente.

Conclusiones

Tras haber realizado este capítulo se puede concluir que los estándares de código propuestos facilitan el trabajo del programador a la hora de implementar el módulo. La propuesta de solución estará orientada a objetos debido al uso de las técnicas de programación y herramientas utilizadas. Además que la estrategia de prueba propuesta posibilitó la detección de errores tanto del código como de la interfaz del módulo, contribuyendo a elevar la calidad de la propuesta de solución.

CONCLUSIONES

Una vez finalizada la implementación del módulo Complejo Residencial, los autores consideran que se cumplieron los objetivos trazados concluyendo que:

- La gestión de hospedaje realizada a través de la versión 1.0 del módulo no se efectuaba de forma eficiente debido a las restricciones de la solución, por lo que la versión 2.0 del módulo Complejo Residencial implementa catorce nuevas funcionalidades que aportan robustez y valor agregado a la solución final.
- El estudio de los sistemas homólogos contribuyó a la incorporación de funcionalidades como multimoneda y control de medios básicos que no fueron contempladas inicialmente por el cliente.
- La organización de la información en cuatro agrupaciones funcionales ofrece al usuario final una fácil interacción con el sistema y se ajusta a la definición de los requerimientos no funcionales.

RECOMENDACIONES

- Confeccionar el Manual de usuarios para un mejor manejo y comprensión del sistema por el cliente final.

- El sistema final podrá ser utilizado en otras entidades que tengan características o brinden un servicio similar al del Complejo Residencial de la Universidad, colaborando de esta manera a la informatización de la sociedad cubana.

BIBLIOGRAFÍA REFERENCIADA

Alvarez, Miguel Angel. "*Qué es HTML*." 2001. Disponible en: http://www.desarrolloweb.com/articulos/que-es-html.html.

Alvarez, Miguel Angel. "*CodeIgniter*." 2009. Disponible en: http://www.desarrolloweb.com/articulos/codeigniter.html.

Alvarez, Miguel Angel. "*Introducción a jQuery* ", 2009. Disponible en: http://www.desarrolloweb.com/articulos/introduccion-jquery.html.

Apache Software Foundation. "*Apache Tiles*." 2010. Disponible en: http://tiles.apache.org/2.2/framework/tutorial/basic/concepts.html.

Apache Software Foundation. "*Apache Jmeter*." 2012. Disponible en: http://jmeter.apache.org/.

Blaha, Michael. *Patterns of Data Modeling. Emerging Directions in Database Systems and Applications*. 2010. Disponible. ISSN.

Bonanata, Maximiliano. *Programación y algoritmos*. Buenos Aires, MP Ediciones, 2003. Nº de pág. ISSN 9789875261563.

Casanovas, Josep. "*Usabilidad y arquitectura del software.*", 2004. Disponible en: http://www.desarrolloweb.com/articulos/1622.php.

CQR Sistemas S.R.L. "*Hotel Express*." 2012. Disponible en: http://www.software-hotelero.com.ar/productos_express.html.

Datys. "*eHotel*." 2012. Disponible en: http://www.datys.cu/wpinfproducto.aspx?6.

Gamma, Erich, et al. *Desing Patterns. Elements of Reusable Object-Oriented Software.* 2009. Nº de pág. 431. ISSN 0201633612.

García Vidal, Yanio. *Arquitectura de software metodología SXP*. 2011

GET, Grupo de Electrónica para el Turismo. "*E-Hotel*." 2012. Disponible en: http://www.get.tur.cu/index.php?option=com_content&view=article&id=63&Itemid=19.

Gutmans, Andi, et al. *PHP 5 power programming*. 1ra, 2005. Disponible en: http://www.phptr.com/content/images/013147149X/downloads/013147149X_book.pdf. ISSN 013147149X.

Juristo, Ana M. Natalia, et al. *Técnicas de evaluación de software*. 2005

Lafosse, Jérome. *Struts 2. El framework de desarrollo de aplicaciones Java EE.* 2010. Disponible en: http://books.google.com.cu/books?id=96HHRq6g5x8C&pg=PA11&dq=Framework&hl=es&sa=X&ei=0S-3T9mhDISmgweL86GnCg&ved=0CFAQ6AEwBQ#v=onepage&q=Framework&f=false. ISSN 978-2-7460-5542-1.

Larman, Craig. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México, Prentice Hall, 1999. Nº de pág. 536. ISSN 970-17-0261-1.

Librosweb.es. "*Capítulo 1. Introducción | Introducción a JavaScript*." 2013. Disponible en: http://www.librosweb.es/javascript/capitulo_1.html.

Librosweb.es. "*Introducción a CSS*." 2013. Disponible en: http://www.librosweb.es/css.

Llano Castro, Eileen. *Propuesta para la integración de prácticas de las metodologías XP y Scrum con el Proceso de Administración de Requisitos del nivel 2 de CMMI.* 2010

Logismic. "*Software HotelOnePlace*." 2012. Disponible en: http://www.logismic.mx/hotel-one-place/.

Martínez, Rafael. "*Sobre PostgreSQL-es*." 2010. Disponible en: http://www.postgresql.org.es/sobre_postgresql.

NetBeans, Oracle Corporation. "*NetBeans: Portal del IDE Java de Código Abierto.*", 2012. Disponible en: http://netbeans.org/index_es.html.

Pencil Project, Evolus. "*Pencil Project: Sketching and Prototyping with Firefox.*", 2010. Disponible en: http://pencil.evolus.vn/en-US/Home.aspx.

Pressman, Roger S. *Software engineering: a practitioner's approach.* McGraw-Hill Higher Education, 7ma, 2010. Nº de pág. 928. ISSN 970-10-5473-3.

Sommerville, Ian. *Ingeniería del software*. Madrid, Pearson Educación, S.A, 7ma, 2005. Nº de pág. 712. ISSN 84-7829-074-5.

Swebok. *Guide to the Software Engineering Body of knoeledge.* Bourque, P. & Dupuis, 2004. Nº de pág. ISSN 0-7695-2330-7.

The pgAdmin Development Team. "*pgAdmin | PostgreSQL Tools.*", 2002. Disponible en: http://pgadmin.org/index.php.

Visual Paradigm International Ltd "*Visual Paradigm for UML (ME)*." 2007. Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p.

BIBLIOGRAFÍA CONSULTADA

- Blaha, Michael. *Patterns of Data Modeling. Emerging Directions in Database Systems and Applications.* Taylor & Francia Group, 2010. Nº de pág. 248. ISSN 978-1-4398-1989-0.
- Bonanata, Maximiliano. *Programación y algoritmos*. Buenos Aires, MP Ediciones, 2003. Nº de pág. ISSN 9789875261563.
- Casanovas, Josep. "*Usabilidad y arquitectura del software.*", 2004. Disponible en: http://www.desarrolloweb.com/articulos/1622.php.
- Casasola, Oscar Romero. "*Introducción a UML. Programación en Castellano.*", 2010. Disponible en: http://www.programacion.com/articulo/introduccion_a_uml_181.
- Díaz de Mera, María del Prado. *Implicaciones del Espaciado Armonizado Europeo de Seguridad y Calidad Industrial en las Metodologías de Gestión de Proyectos Sostenibles.* 2011.
- Díaz, F. Javier, et. al. *Usando Jmeter para pruebas de rendimiento*. LINTI. Facultad de Informática, Universidad Nacional de La Plata, 2008.
- Espinosa, Dayanis. *Desarrollo del módulo complejo residencial del Sistema de Gestión Universitaria*. Universidad de las Ciencias Informáticas. La Habana, 2011.
- Gamma, Erich, et al. *Desing Patterns. Elements of Reusable Object-Oriented Software*. 2009. Nº de pág. 431. ISSN 0201633612.
- García Vidal, Yanio. *Arquitectura de software metodología SXP*. 2011
- Larman, Craig. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México, Prentice Hall, 1999. Nº de pág. 536. ISSN 970-17-0261-1.
- Letelier, Patricio y Penadés, Mª Carmen. *Metodologías para el desarrollo de software: eXtreme Programming (XP).* Disponible. ISSN.
- Llano Castro, Eileen. *Propuesta para la integración de prácticas de las metodologías XP y Scrum con el Proceso de Administración de Requisitos del nivel 2 de CMMI.* 2010
- Palacio, Juan. *Flexibilidad con Scrum*. 2007. Disponible en: http://www.safecreative.org/work/0710210187520. ISSN 0-710210-187520.
- Povea, Yuneisy. *Análisis y diseño del módulo de Hospedaje en el Complejo Residencial del Sistema de Gestión de Residencia.* Universidad de las Ciencias Informáticas. La Habana, 2011. Nº de pág. 73.
- Pressman, Roger S. *Software engineering: a practitioner's approach.* McGraw-Hill Higher Education, 7ma, 2010. Nº de pág. 928. ISSN 970-10-5473-3.Sommerville, Ian. *Ingeniería del software*. Madrid, Pearson Educación, S.A, 7ma, 2005. Nº de pág. 712. ISSN 84-7829-074-5.

- Sommerville, Ian. *Ingeniería del software*. Madrid, Pearson Educación, S.A, 7ma, 2005. Nº de pág. 712. ISSN 84-7829-074-5.
- Swebok. *Guide to the Software Engineering Body of knoeledge.* Bourque, P. & Dupuis, 2004. Nº de pág. ISSN 0-7695-2330-7.

GLOSARIO DE TÉRMINOS

A

AJAX: Acrónimo de *Asynchronous JavaScript And XML*, en español: JavaScript Asíncrono y XML. Es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (*Rich Internet Applications*, aplicaciones de Internet enriquecidas).

API: Por sus siglas en inglés *Application Programming Interface*, en español Interfaz de programación de aplicaciones. Es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro *software* como una capa de abstracción. Son usadas generalmente en las librerías.

C

CamelCase: Es un estilo de escritura que se aplica a frases o palabras compuestas. El nombre se debe a que las mayúsculas a lo largo de una palabra en CamelCase se asemejan a las jorobas de un camello. Se podría traducir como Mayúsculas/Minúsculas Camello. El término case se traduce como "caja tipográfica", que a su vez implica si una letra es mayúscula o minúscula y tiene su origen en la disposición de los tipos móviles en casilleros o cajas.

Check-in: Registro, es el proceso mediante el cual un recepcionista registra la llegada de un cliente a un hotel, aeropuerto o puerto.

CMMI: Por sus siglas en inglés *Capability Maturity Model Integration*, en español: Modelo de Madurez y Capacidad Integrado.

CSS: Por sus siglas en inglés *Cascading Style Sheets*, en español: Estilo de Hojas en Cascada.

D

Datawarehouse: Almacén de datos, colección de datos orientada a un dominio, integrado, no volátil, y que varía en el tiempo.

DOM: abreviatura de *Document Object Model*, en español, pero no oficialmente, Modelo en Objetos para la representación de Documentos o Modelo de Objetos del Documento, es una interfaz de programación de aplicaciones que proporciona un conjunto estándar de objetos para representar documentos HTML y XML. Es un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos.

F

FTP: abreviatura de *file transfer protocol* [protocolo de transferencia de archivos]. Protocolo utilizado para tener acceso a un anfitrión [host] de Internet , y posteriormente para transferir archivos entre ese anfitrión y la computadora que se está utilizando.

G

GOF: Por sus siglas en inglés: *Gang of Four*, en español Banda de los Cuatro, forma en que se conoce al grupo de los 4 autores: Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides.

GRASP: Por sus siglas en inglés: *General Responsibility Assignment Software Patterns*, en español: Patrones Generales de *Software* para Asignar Responsabilidades. Desde el punto de vista técnico, deberíamos escribir "Patrones GRAS" en vez de "Patrones GRASP" pero la segunda expresión suena mejor en inglés, idioma en que fue acuñado el término.

GUI: Por sus siglas en inglés *Graphical User Interface*, en español Interfaz Gráfica de Usuario.

H

HTML: Acrónimo de *HyperText Markup Language* (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web.

HTTP: Por sus sigla en inglés: *HyperText Transfer Protocol*, en español: Protocolo de Transferencia de Hipertexto.

I

IEEE: Por sus siglas en inglés *Institute of Electrical and Electronics Engineers*, en español Instituto de Ingenieros Eléctricos y Electrónicos, una asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas.

Interoperabilidad: Condición necesaria para que los usuarios (humanos o mecánicos) tengan un acceso completo a la información disponible.

IOC: Acrónimo de *Inversion of Control*, en español: Inversión de Control. Es un principio en el cual se basan muchos de los patrones que se ven día a día. Consiste en invertir el flujo de ejecución de modo que el código sea invocado, en vez de los programadores ser los invocadores, lo que implicaría dejar de tener control sobre el flujo de ejecución.

J

jQuery: Es una biblioteca o *framework* de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX a páginas web.

O

Orientado a objetos: Hace referencia a la Programación Orientada a Objetos, un paradigma de programación que usa objetos y sus interacciones, para diseñar aplicaciones y programas de ordenador. Está basado en varias técnicas, incluyendo herencia, abstracción, polimorfismo y encapsulamiento.

P

PHP: Acrónimo de *Hypertext Preprocessor*, en español: Hipertexto Pre-Procesado.

Plugins: también conocido como (*plug-in, add-in, addon, add-on, snap-in or snapin*), es un pequeño programa de computadora que extiende las capacidades de otro programa de mayor tamaño, adicionándole nuevas funcionalidades.

PMS: *Property Management System*, es un programa informático que permite gestionar las distintas áreas de un hotel, desde la gestión del inventario hasta la disponibilidad del establecimiento y las acciones comerciales llevadas a cabo con los clientes.

Python: Es un lenguaje de programación de programación, orientado a objetos, para el desarrollo web y bajo licencia de código abierto.

S

SQL: Por sus siglas en inglés: *Structured Query Language*, en español: Lenguaje de Consulta Estructurado.

SSL: Por sus siglas en inglés: *Secure Sockets Layer*, en español: Capa de Conexión Segura.

T

TIC: Tecnologías de la Información y las Comunicaciones.

TCP/IP: Por sus siglas en inglés: *Transmission Control Protocol/Internet Protocol*, en español: Protocolo de Control de Transmisión/Protocolo de Internet.

U

UML: Por sus siglas en inglés *Unified Modeling Language*, en español: Lenguaje Unificado de Modelado

UNIX: Sistema operativo portable, multitarea y multiusuario; desarrollado, en principio en 1969, por un grupo de empleados de los laboratorios Bell de AT&T.

Unix Domain Sockets: Un socket de dominio UNIX o socket IPC (socket de comunicación interprocesos) es un socket virtual, similar a un socket de Internet que se utiliza en los sistemas operativos POSIX para comunicación entre procesos.

W

Widgets: Formado de la unión de las palabras *window* (ventana) y *gadget* (artilugio) es un componente gráfico, o control, con el cual el usuario interactúa, como por ejemplo, una ventana, una barra de tareas o una caja de texto.

X

XHTML: Por sus siglas en inglés *eXtensible HyperText Markup Language*. XHTML es básicamente HTML expresado como XML válido. Es más estricto a nivel técnico.

XML: Por sus siglas en inglés *Extensible Markup Language*, en español: Lenguaje de Marcas Extensible.

ANEXOS

Anexo 1. Modelo de entrevista

Nombre (s): Víctor Michel
Apellidos: Hernández Gómez
Cargo que ocupa: Director Complejo Residencial
Departamento al que pertenece: Dirección Complejo Residencial

Preguntas:
1. ¿En qué consiste el proceso?
2. ¿Cuáles son las características fundamentales del proceso?
3. ¿Cuál es el objetivo principal del proceso?
4. ¿Existe algún mecanismo para llevar a cabo el proceso?
5. ¿Quién ubica las reservaciones?
6. ¿Qué hace la carpetera?
7. ¿Quién puede gestionar los medios básicos?
8. ¿Qué beneficios brinda el proceso?
9. ¿Existen herramientas que ayuden a la ejecución del proceso?
10. ¿Qué limitaciones presenta el proceso actualmente?
11. ¿Qué desearía aportar al proceso para hacer el trabajo más eficiente?

Anexo 2. Especificación de requisitos funcionales

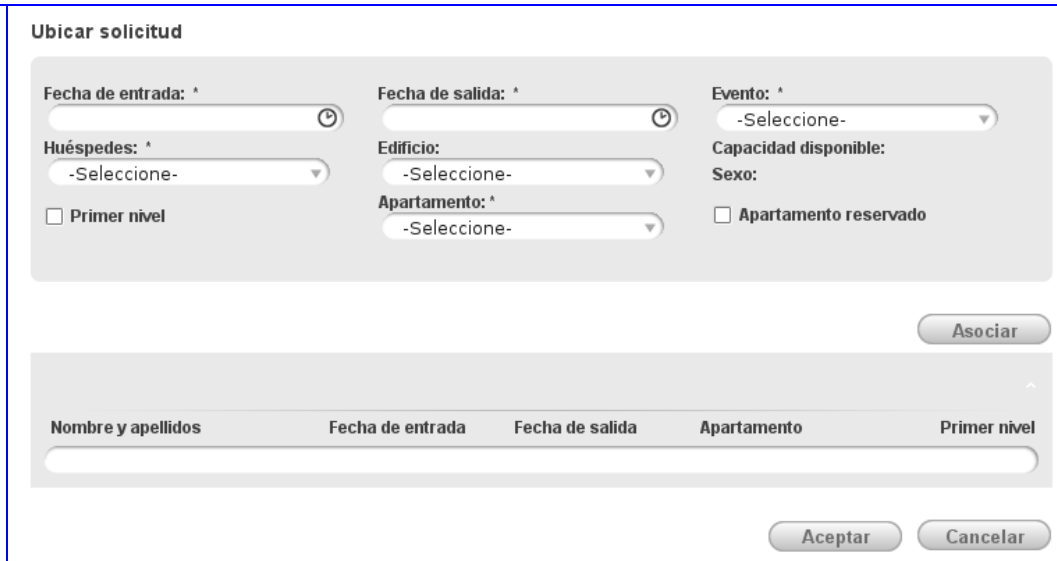
En este anexo solo se hará referencia a cuatro especificaciones de requisitos funcionales. Para buscar más información se puede consultar el expediente del Proyecto Residencia donde se encuentran todas las especificaciones de requisitos.

Tabla 7. Especificación del requisito "Mostrar solicitudes aprobadas".

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RFCR1	Mostrar solicitudes aprobadas	Se listan los solicitantes que tienen solicitudes aprobadas, estas se obtienen del módulo Solicitudes. A partir de allí se pueden ubicar cada una de ellas y ver los detalles de la solicitud. También se puede buscar un solicitante por su nombre y apellidos.	Alta	Alta
	Prototipo			
				
	Observaciones	Si no hay solicitudes se muestra el listado vacío. Permite listar la cantidad de incidencias que se seleccione. Al pasar el puntero por cualquiera de los iconos que contenga la incidencia listada, debe mostrar el nombre de la acción.		

Tabla 8. Especificación del requisito "Ubicar solicitud"

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RFCR2	Ubicar solicitud	Se ubican las solicitudes correspondientes a cada solicitante. Se establece la fecha de la estancia y el evento o motivo de la misma. Se selecciona el huésped, así como el edificio y el apartamento, y se precisa si el huésped es de primer nivel y si el apartamento está reservado. Una vez asociado el huésped se puede eliminar del listado de los huéspedes ubicados.	Alta	Alta
	Prototipo			



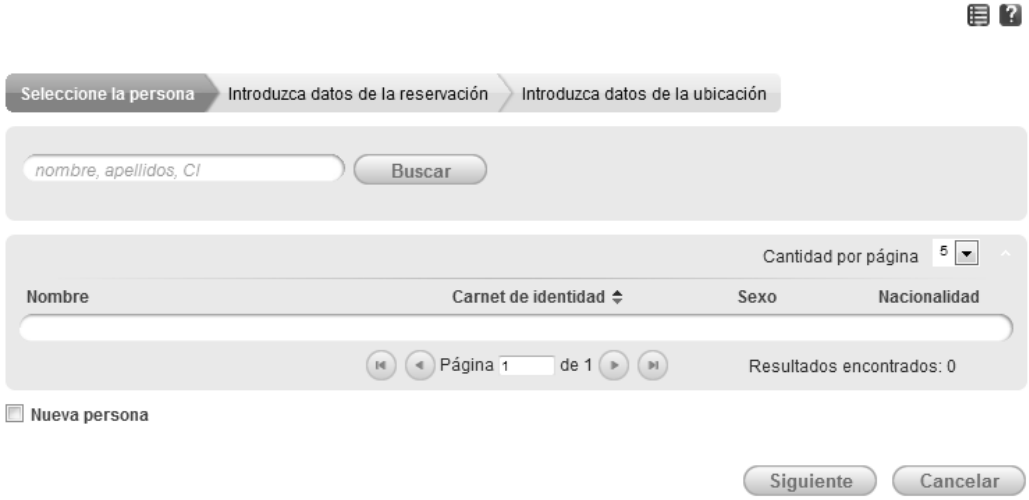
Campos	**Tipos de Datos**	**Reglas o Restricciones**
Fecha de entrada	Fecha	Campo obligatorio. La fecha no debe ser mayor que la de salida.
Fecha de salida	Fecha	Campo obligatorio. La fecha no debe ser menor que la de entrada.
Evento	Lista desplegable	Campo obligatorio
Huéspedes	Lista desplegable	Campo obligatorio
Primer nivel	Booleano	[Activo, Inactivo]
Edificio	Lista desplegable	No procede
Apartamento	Lista desplegable	Campo obligatorio
Capacidad	Varchar	No procede
Sexo	Varchar	No procede
Apartamento reservado	Booleano	[Activo, Inactivo]
Observaciones	Al pasar el puntero por cualquiera de los iconos que contenga la incidencia listada, debe mostrar el nombre de la acción.	

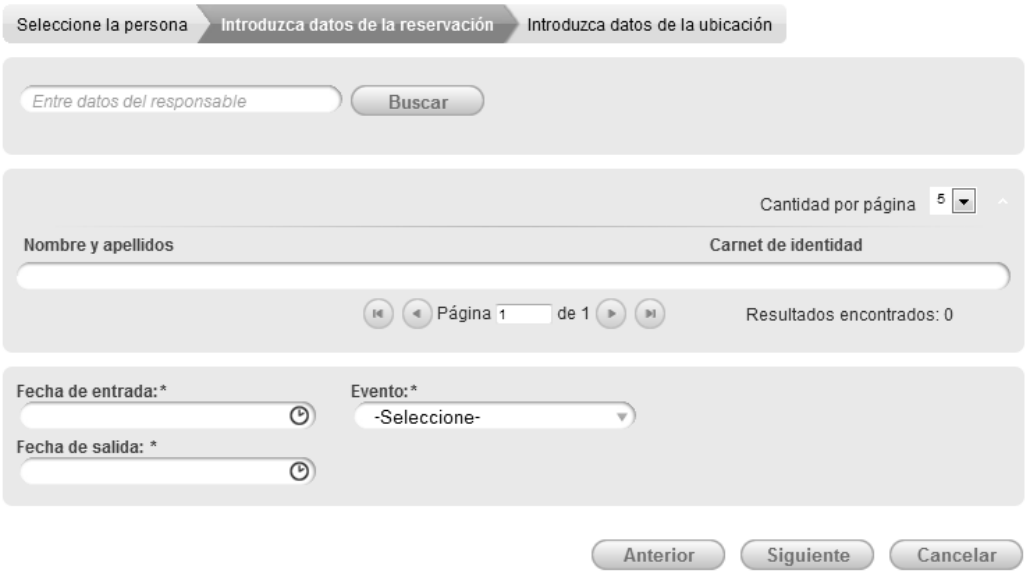
Tabla 9. Especificación del requisito "Crear reservación sin solicitud"

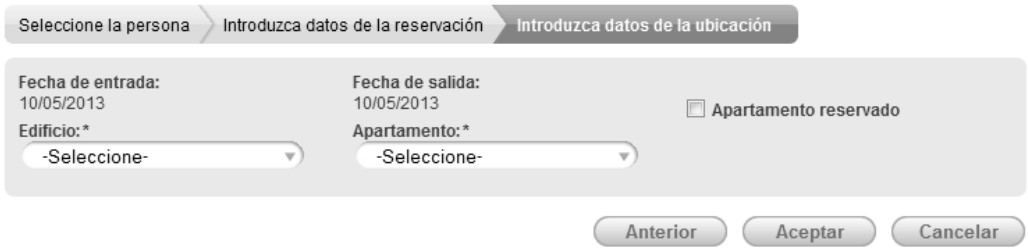
Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RFCR4	Crear reservación sin solicitud	En caso de que una persona no tenga una solicitud de reservación y necesite alojarse se procede a crear dicha reservación. Se busca a la persona atendiendo cualquiera de los criterios: “Nombre”, “Apellido” o “CI”. Si la persona no existe en la base de datos se crea una nueva persona y se introducen los datos de la misma (Ver Crear nueva	Alta	Alta

		persona). Se selecciona el botón “Siguiente” y se procede a introducir los datos de la reservación. Se busca al responsable de la reservación y se especifica la fecha de entrada y de salida, así como el motivo de la misma. Se selecciona el botón “Siguiente” para registrar los datos de la ubicación, donde se escoge el edificio y el apartamento. En caso de que se reserve el apartamento se marca el campo “Reservado”. Se selecciona el botón “Aceptar” y el sistema muestra el siguiente mensaje: “Se ha creado el elemento correctamente”.		

Prototipo







Campos	**Tipos de Datos**	**Reglas o Restricciones**
Buscar persona	Números, Texto	Los nombres y apellidos no deben contener números. En el caso del carnet de identidad no debe exceder los 11 dígitos.
Buscar responsable	Texto	Los nombres y apellidos no deben contener números. En el caso del carnet de identidad no debe exceder los 11 dígitos.
Fecha de entrada	Fecha	Campo obligatorio. La fecha no debe ser mayor que la de salida.
Fecha de salida	Fecha	Campo obligatorio. La fecha no debe ser menor que la de entrada.
Evento	Lista desplegable	Campo obligatorio.
Edificio	Lista desplegable	No procede
Apartamento	Campo de selección única	Campo obligatorio.
Reservado	Booleano	[Activo, Inactivo]
Observaciones	Si selecciona el botón “Cancelar” el sistema muestra un mensaje de confirmación: “¿Está seguro de realizar la acción?”. Si se selecciona el botón “Anterior” el sistema regresa a la vista anterior.	

Tabla 10. Especificación del requisito "Crear nueva persona"

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RFCR5	Crear nueva persona	En caso de que la persona no exista en la base de datos se crea una nueva. Se introducen los datos de la persona y se selecciona el botón “Añadir”. Se muestran los datos de la persona.	Alta	Alta
	Prototipo			

Campos	**Tipos de Datos**	**Reglas o Restricciones**
Primer nombre	Varchar	Campo obligatorio. Admite un rango de caracteres válidos de 1 a 50 caracteres.
Segundo nombre	Varchar	Admite un rango de caracteres válidos de 1 a 50 caracteres.
Primer apellido	Varchar	Campo obligatorio. Admite un rango de caracteres válidos de 1 a 50 caracteres.
Segundo apellido	Varchar	Campo obligatorio. Admite un rango de caracteres válidos de 1 a 50 caracteres.
Sexo	Lista desplegable	Campo obligatorio.
Carnet de identidad	Varchar	Campo obligatorio. No debe exceder los 11 dígitos.
Nacionalidad	Lista desplegable	Campo obligatorio.
Observaciones	Si selecciona el botón “Cancelar” el sistema muestra un mensaje de confirmación: “¿Está seguro de realizar la acción?”.	

Anexo 3. Modelo físico de datos.

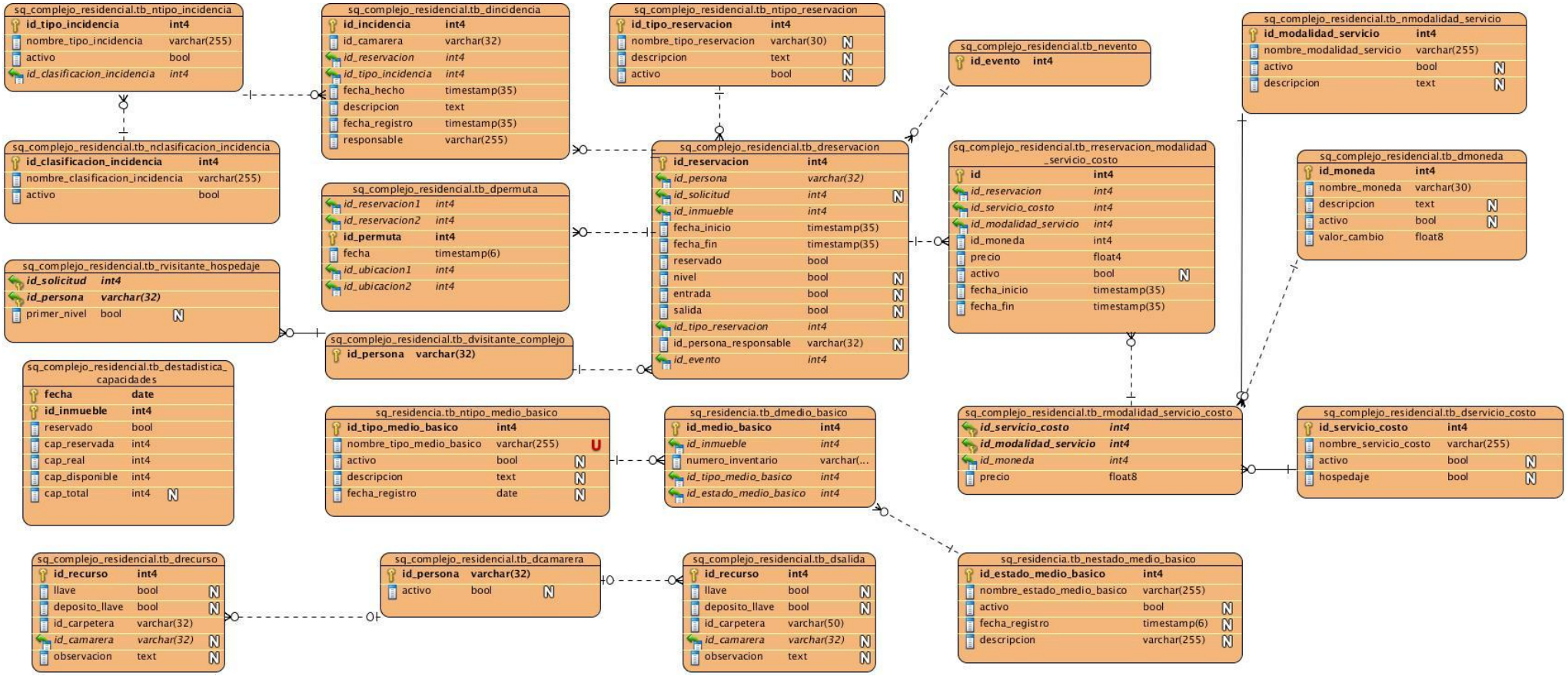


Figura 13. Modelo físico de datos

Anexo 4. Modelo lógico de datos.

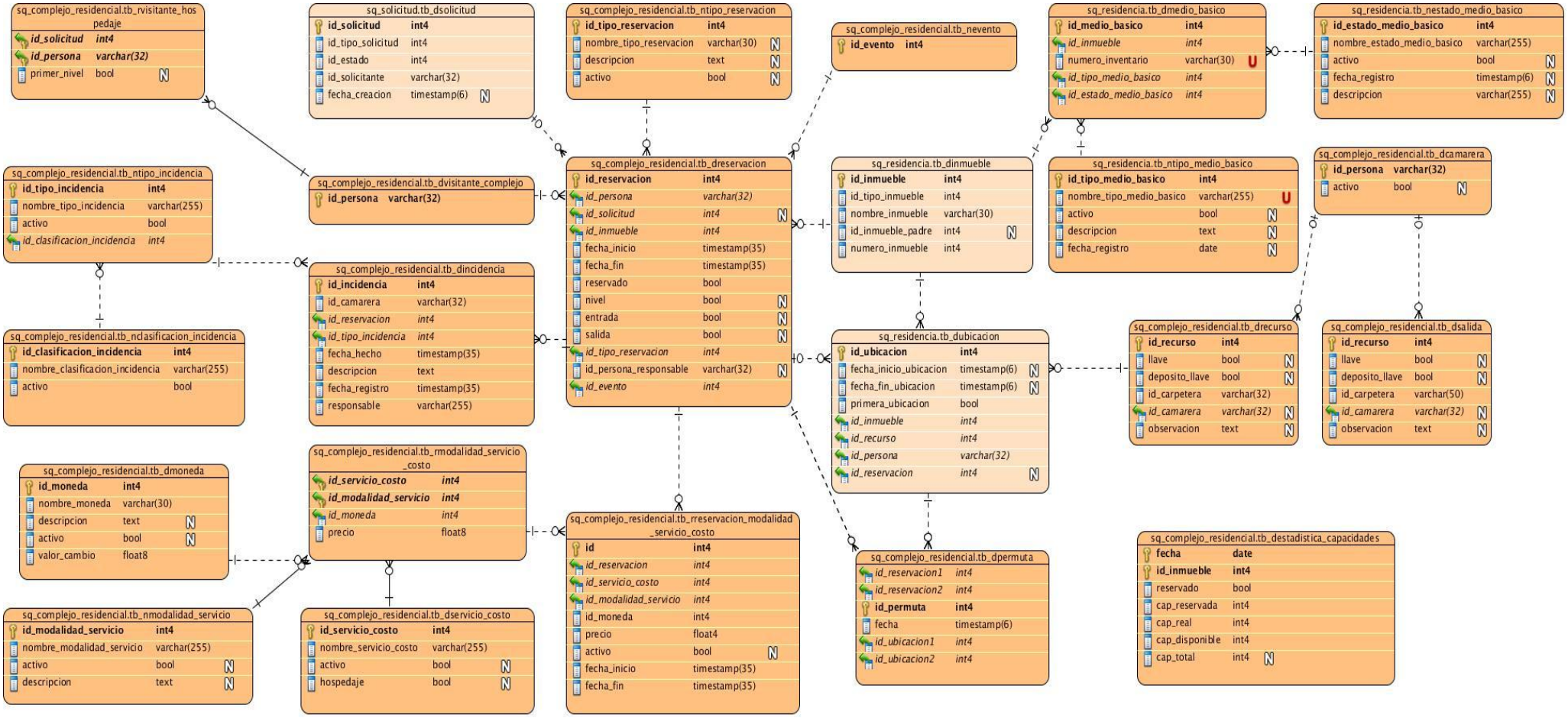


Figura 14. Modelo lógico de datos

Anexo 5. Diseño de casos de prueba.

En este anexo solo se hará referencia a cuatro diseños de caso de prueba basado en requisitos. Para buscar más información se puede consultar el expediente del Proyecto Residencia en donde se encuentran todos los diseños.

Tabla 11. Diseño de caso de prueba del requisito Registrar incidencia

Escenario	Descripción	Variable 1 "Tipo de incidencia	Variable 2 "Responsable"	Variable 3 "Fecha de la incidencia"	Variable 4 "Camarera"	Variable 5 "Descripción"	Respuesta del sistema	Flujo central
EC 1.1 Insertar datos correctamente	Mediante este escenario se inserta en el sistema un tipo de incidencia	V ("Pérdida de llave")	V ("Adriana Alfonso Luis")	V ("10/05/2013")	V ("Laritza González Marrero")	V (Vacío)	El sistema actualiza el listado y muestra el mensaje “El elemento ha sido creado satisfactoriamente”.	• El usuario una vez autenticado selecciona el sistema “Residencia” el cual se encuentra en el escritorio. • El sistema muestra una nueva ventana con varias opciones. • El usuario selecciona el módulo “Complejo Residencial” del sistema “Residencia” • El sistema muestra diferentes opciones del menú. • El usuario selecciona el menú “Configuración” • El usuario selecciona la opción “Tipo de incidencia”. • El sistema muestra los tipos de incidencia existentes en el listado. • El usuario selecciona la opción “Crear” en el área de iconos flotantes. • El usuario llena todos los campos satisfactoriamente y presiona el botón “Aceptar”. • El sistema actualiza el listado y muestra el mensaje “El elemento ha sido creado satisfactoriamente.”
		V ("Ruido")	V ("Adriana Alfonso Luis")	V ("10/05/2013")	V ("Laritza González Marrero")	V ("Música en la habitación a altas horas de la noche")		
EC 1.2 Insertar elementos repetidos	Mediante este escenario se introducen	I ("Pérdida de llave")	V ("Adriana Alfonso Luis")	V ("10/05/2013")	V ("Laritza González Marrero")	V (Vacío)	El sistema muestra un mensaje de error “El elemento ya existe” y no lo	• El usuario una vez autenticado selecciona el sistema “Residencia” el cual se encuentra en el escritorio.

	datos para insertar un tipo de incidencia que ya existe	I ("Pérdida de llave")	V ("Adriana Alfonso Luis")	V (“10/05/2013”)	V ("Laritza González Marrero")	V (Vacío)	suscribe.	• El sistema muestra una nueva ventana con varias opciones. • El usuario selecciona el módulo “Complejo Residencial” del sistema “Residencia” • El sistema muestra diferentes opciones del menú. • El usuario selecciona el menú “Configuración” • El usuario selecciona la opción “Tipo de incidencia”. • El sistema muestra los tipos de incidencias existentes en el listado. • El usuario selecciona la opción “Crear” en el área de iconos flotantes. • El usuario llena todos los campos satisfactoriamente y presiona el botón “Aceptar”. • El sistema muestra un mensaje de error “El elemento ya existe” y no lo suscribe.
EC 1.3 Insertar datos incompletos	Mediante este escenario no se introducen	I (Vacío)	V ("Muy grave")	I (Vacío)	V (“Laritza González Marrero”)	V ("Activo")	El sistema muestra un mensaje de error “El elemento ya existe” y no lo suscribe.	• El usuario una vez autenticado selecciona el sistema “Residencia” el cual se encuentra en el escritorio. • El sistema muestra una nueva
		I	V	V	I	V		

	todos los datos para insertar un evento.	(Vacío)	("Muy grave")	(“10/05/2013”)	(Vacío)	(Vacío)		ventana con varias opciones. • El usuario selecciona el módulo “Complejo Residencial” del sistema “Residencia” • El sistema muestra diferentes opciones del menú. • El usuario selecciona el menú “Configuración” • El usuario selecciona la opción “Tipo de incidencia”. • El sistema muestra los tipos de incidencia existentes en el listado. • El usuario selecciona la opción “Crear” en el área de iconos flotantes. • El usuario no llena los campos satisfactoriamente y presiona el botón “Aceptar”. • El sistema muestra un mensaje: “Uno o varios de los elementos se ha introducido de forma incorrecta”.
EC 1.4 Cancelar operación	Mediante este escenario se cancela la creación de un tipo de incidencia.	V ("Pérdida de llave")	V (“Adriana Alfonso Luis”)	V (“10/05/2013”)	V (“Laritza González Marrero”)	V (Vacío)	El sistema muestra el mensaje "¿Está seguro de realizar esta acción?"	• El usuario una vez autenticado selecciona el sistema “Residencia” el cual se encuentra en el escritorio. • El sistema muestra una nueva ventana con varias opciones. • El usuario selecciona el módulo “Complejo Residencial” del sistema “Residencia” • El sistema muestra diferentes opciones del menú. • El usuario selecciona el menú “Configuración” • El usuario selecciona la opción “Tipo de incidencia”. • El sistema muestra los tipos de incidencia existentes en el listado. • El usuario selecciona la opción

								“Crear” en el área de iconos flotantes. • El usuario llena todos los campos satisfactoriamente y presiona el botón “Cancelar”. • El sistema muestra un mensaje: “¿Está seguro de realizar la acción?".

Tabla 12. Diseño de caso de prueba del requisito Mostrar incidencia

Escenario	Descripción	Variable 1 "Tipo de incidencia"	Variable 2 "Clasificación"	Variable 3 "Fecha del incidente"	Variable 4 "Fecha de registro"	Respuesta del sistema	Flujo central
EC 1.1 Mostrar datos correctamente	Mediante este escenario se muestra un listado con las incidencia existentes hasta la fecha	V ("Pérdida de llave")	V ("Muy grave")	V ("20/03/2013")	V ("21/03/2013")	El sistema muestra un listado con las incidencia existentes hasta la fecha	• El usuario una vez autenticado selecciona el sistema “Residencia” el cual se encuentra en el escritorio. • El sistema muestra una nueva ventana con varias opciones. • El usuario selecciona el módulo “Complejo Residencial” del sistema “Residencia” • El sistema muestra diferentes opciones del menú. • El usuario selecciona el menú “Huéspedes” • El usuario selecciona la opción “Incidencia”. • El sistema muestra un listado de los huéspedes que tienen incidencias. • El usuario selecciona la opción "Ver incidencias" en el área de íconos internos del huésped seleccionado. • El sistema muestra un listado con todas las incidencias que tiene el huésped seleccionado.
		V ("Ruido")	V ("Menos grave")	V ("28/03/2013")	V ("30/03/2013")		
EC 1.2 No existen elementos a mostrar	Mediante este escenario se muestra el listado de elementos vacío.	(Vacío)	(Vacío)	(Vacío)	(Vacío)	El sistema no muestra elementos	• El usuario una vez autenticado selecciona el sistema “Residencia” el cual se encuentra en el escritorio. • El sistema muestra una nueva ventana con varias opciones. • El usuario selecciona el módulo “Complejo Residencial” del sistema “Residencia” • El sistema muestra diferentes opciones del menú. • El usuario selecciona el menú “Huéspedes” • El usuario selecciona la opción “Incidencia”. • El sistema muestra un listado vacío.
EC 1.3 Selecciona cantidad por	Mediante este escenario se	NA	NA	NA	NA	El sistema muestra la cantidad de	• El usuario una vez autenticado selecciona el sistema “Residencia” el cual se encuentra en el escritorio.

página.	puede seleccionar la cantidad de elementos que se desean por páginas, las que pueden ser: 5, 10, 20, 40.					elementos por páginas seleccionado por el usuario.	• El sistema muestra una nueva ventana con varias opciones. • El usuario selecciona el módulo “Complejo Residencial” del sistema “Residencia” • El sistema muestra diferentes opciones del menú. • El usuario selecciona el menú “Huéspedes” • El usuario selecciona la opción “Incidencia”. • El sistema muestra un listado de los huéspedes que tienen incidencias. • El usuario selecciona la cantidad de elementos que desea ver por página.
EC 1.4 Seleccionar página Siguiente.	Mediante este escenario se puede seleccionar la próxima página con el botón *Siguiente*.	NA	NA	NA	NA	El sistema muestra la página a continuación.	• El usuario una vez autenticado selecciona el sistema “Residencia” el cual se encuentra en el escritorio. • El sistema muestra una nueva ventana con varias opciones. • El usuario selecciona el módulo “Complejo Residencial” del sistema “Residencia” • El sistema muestra diferentes opciones del menú. • El usuario selecciona el menú “Huéspedes” • El usuario selecciona la opción “Incidencia”. • El sistema muestra un listado de los huéspedes que tienen incidencias. • El usuario selecciona la opción "Siguiente". • El sistema muestra la página siguiente.

Tabla 13. Diseño de caso de prueba del requisito Modificar incidencia

Escenario	Descripción	Variable 1 "Tipo de incidencia"	Variable 2 "Fecha de la incidencia"	Variable 3 "Responsable"	Variable 4 "Camarera"	Variable 5 "Descripción"	Respuesta del sistema	Flujo central
EC 1.1 Modificar datos correctamente	Mediante este escenario se modifica en el sistema una incidencia	V ("Pérdida de llave")	V ("27/03/2013")	V Osvaldo Aguilera Velázquez	V Laritza González Marrero	V ("El huésped perdío la llave de la habitación en la que está hospedado")	El sistema actualiza el listado y muestra el mensaje “El elemento ha sido modificado satisfactoriamente”.	• El usuario una vez autenticado selecciona el sistema “Residencia” el cual se encuentra en el escritorio. • El sistema muestra una nueva ventana con varias opciones. • El usuario selecciona el módulo “Complejo Residencial” del sistema “Residencia” • El sistema muestra diferentes opciones del menú. • El usuario selecciona el menú “Huéspedes” • El usuario selecciona la opción “Incidencias”. • El sistema muestra un listado de los huéspedes que tienen incidencias. • El usuario selecciona la opción "Ver incidencias" en el área de íconos internos del huésped seleccionado. • El sistema muestra un listado con todas las incidencias que tiene el huésped seleccionado. • El usuario selecciona la opción “Modificar” en el área de iconos internos. • El usuario llena todos los campos satisfactoriamente y presiona el botón “Aceptar”. • El sistema actualiza el listado y muestra el mensaje “El elemento ha sido modificado satisfactoriamente.”
EC 1.2	Mediante	I	V	V	V	V	El sistema muestra	• El usuario una vez autenticado

Modificar elemento por otro ya existente.	este escenario se modifica un tipo de incidencia ya existente en el sistema	("Pérdida de llave")	("27/03/2013")	Osvaldo Aguilera Velázquez	Laritza González Marrero	("El huésped perdío la llave de la habitación en la que está hospedado")	un mensaje: “El elemento ya existe” y no lo suscribe.	selecciona el sistema “Residencia” el cual se encuentra en el escritorio. • El sistema muestra una nueva ventana con varias opciones. • El usuario selecciona el módulo “Complejo Residencial” del sistema “Residencia” • El sistema muestra diferentes opciones del menú. • El usuario selecciona el menú “Huéspedes” • El usuario selecciona la opción “Incidencias”. • El sistema muestra un listado de los huéspedes que tienen incidencias. • El usuario selecciona la opción "Ver incidencias" en el área de íconos internos del huésped seleccionado. • El sistema muestra un listado con todas las incidencias que tiene el huésped seleccionado. • El usuario selecciona la opción “Modificar” en el área de iconos internos. • El usuario llena todos los campos satisfactoriamente y presiona el botón “Aceptar”. • El sistema muestra el mensaje: “El elemento ya existe” y no lo suscribe.
EC 1.3 Modificar datos	Mediante este escenario	I ("Pérdida de llave")	V ("Muy grave")	V Osvaldo Aguilera	V Laritza González	V (Vacío)	El sistema muestra un mensaje: “Uno o varios elementos	• El usuario una vez autenticado selecciona el sistema “Residencia” el cual se

incompletos	no se introducen todos los datos para modificar un tipo de incidencia.			Velázquez	Marrero		se han introducido de forma incorrecta".	encuentra en el escritorio. • El sistema muestra una nueva ventana con varias opciones. • El usuario selecciona el módulo "Complejo Residencial" del sistema "Residencia" • El sistema muestra diferentes opciones del menú. • El usuario selecciona el menú "Huéspedes" • El usuario selecciona la opción "Incidencias". • El sistema muestra un listado de los huéspedes que tienen incidencias. • El usuario selecciona la opción "Ver incidencias" en el área de íconos internos del huésped seleccionado. • El sistema muestra un listado con todas las incidencias que tiene el huésped seleccionado. • El usuario selecciona la opción "Modificar" en el área de iconos internos. • El usuario no llena todos los campos satisfactoriamente y presiona el botón "Aceptar". • El sistema muestra el mensaje: "Uno o varios elementos de han introducido de forma incorrecta" y no lo suscribe.
EC 1.4 Cancelar operación	Mediante este escenario se cancela la modificación de una incidencia.	V ("Pérdida de llave")	V ("Muy grave")	V ("Osvaldo Aguilera Velázquez")	V ("Laritza González Marrero")	V ("Activo")	El sistema muestra el mensaje "¿Está seguro de realizar esta acción?"	• El usuario una vez autenticado selecciona el sistema "Residencia" el cual se encuentra en el escritorio. • El sistema muestra una nueva ventana con varias opciones. • El usuario selecciona el módulo "Complejo Residencial"

								del sistema “Residencia” • El sistema muestra diferentes opciones del menú. • El usuario selecciona el menú “Huéspedes” • El usuario selecciona la opción “Incidencias”. • El sistema muestra un listado de los huéspedes que tienen incidencias. • El usuario selecciona la opción "Ver incidencias" en el área de íconos internos del huésped seleccionado. • El sistema muestra un listado con todas las incidencias que tiene el huésped seleccionado. • El usuario selecciona la opción “Modificar” en el área de iconos internos. • El usuario llena todos los campos satisfactoriamente y presiona el botón “Cancelar”. • El sistema muestra un mensaje: “¿Está seguro de realizar esta acción?".

Tabla 14. Diseño de caso de prueba del requisito Ver detalles de incidencia

Escenario	Descripción	Variable 1 "Tipo de incidencia "	Varable 2 "Clasificación"	Variable 3 "Camarera "	Variable 4 "Responsable "	Variable 5 "Fecha del incidente "	Variable 6 "Fecha de registro "	Variable 7 "Descripción "	Respuesta del sistema	Flujo central
EC 1.1 Mostrar datos correctamente	Mediante este escenario se muestran los datos de la incidencia	NA	NA	NA	NA	NA	NA	NA	El sistema muestra todos los datos relacionados con la incidencia seleccionada.	• El usuario una vez autenticado selecciona el sistema “Residencia” el cual se encuentra en el escritorio. • El sistema muestra una nueva ventana con varias opciones. • El usuario selecciona el módulo “Complejo Residencial” del sistema “Residencia” • El sistema muestra diferentes opciones del menú. • El usuario selecciona el menú “Huéspedes” • El usuario selecciona la opción “Incidencia”. • El sistema muestra un listado de los huéspedes que tienen incidencias. • El usuario selecciona la opción "Ver incidencias" en el

										área de íconos internos. • El sistema muestra un listado con todas las incidencias que tiene el huésped seleccionado. • El sistema muestra todos los datos relacionados con la incidencia seleccionada.
EC 1.2 Pasar el puntero del mouse sobre el ícono "Ver detalles"	Mediante este escenario se indica la acción ver detalles.	NA	NA	NA	NA	Na	NA	NA	El sistema indica la acción.	• El usuario una vez autenticado selecciona el sistema “Residencia” el cual se encuentra en el escritorio. • El sistema muestra una nueva ventana con varias opciones. • El usuario selecciona el módulo “Complejo Residencial” del sistema “Residencia” • El sistema muestra diferentes opciones del menú. • El usuario selecciona el menú “Huéspedes” • El usuario

										selecciona la opción “Incidencia”. • El sistema muestra un listado de los huéspedes que tienen incidencias. • El usuario selecciona la opción "Ver incidencias" en el área de íconos internos. • El sistema muestra un listado con todas las incidencias que tiene el huésped seleccionado. • El usuario pasa el puntero del mouse sobre la opción "Ver detalles" en el área de íconos internos de la incidencia que desea. • El sistema indica la acción
EC 1.3 Cerrar con el botón	Mediante este escenario se cierra la	NA	NA	NA	NA	NA	NA	NA	El sistema cierra la ventana.	• El usuario una vez autenticado selecciona el

"Cerrar" que aparece en la parte derecha inferior de la ventana.	ventana donde se muestran los detalles de la incidencia.									sistema “Residencia” el cual se encuentra en el escritorio. • El sistema muestra una nueva ventana con varias opciones. • El usuario selecciona el módulo “Complejo Residencial” del sistema “Residencia” • El sistema muestra diferentes opciones del menú. • El usuario selecciona el menú “Huéspedes” • El usuario selecciona la opción “Incidencia”. • El sistema muestra un listado de los huéspedes que tienen incidencias. • El usuario selecciona la opción "Ver incidencias" en el área de íconos internos. • El sistema muestra todas las incidencias que tiene el huésped seleccionado. • El usuario selecciona la incidencia deseada

										y la opción "Ver detalles" en el área de íconos internos. • El sistema muestra todos los datos relacionados con la incidencia seleccionada. • El usuario cierra la ventana.
EC 1.4 Cerrar con el botón "X" que aparece en la parte superior derecha de la ventana.	Mediante este escenario se cierra la ventana donde se muestran los detalles de la incidencia.	NA	NA	NA	NA	NA	NA	NA	El sistema cierra la ventana.	• El usuario una vez autenticado selecciona el sistema “Residencia” el cual se encuentra en el escritorio. • El sistema muestra una nueva ventana con varias opciones. • El usuario selecciona el módulo “Complejo Residencial” del sistema “Residencia” • El sistema muestra diferentes opciones del menú. • El usuario selecciona el menú “Huéspedes” • El usuario

										selecciona la opción “Incidencia”. • El sistema muestra un listado de los huéspedes que tienen incidencias. • El usuario selecciona la opción "Ver incidencias" en el área de íconos internos. • El sistema muestra todas las incidencias que tiene el huésped seleccionado. • El usuario selecciona la incidencia deseada y la opción "Ver detalles" en el área de íconos internos. • El sistema muestra todos los datos relacionados con la incidencia seleccionada. • El usuario cierra la ventana.

Anexo 6. Casos de prueba de integración.

Tabla 15. Caso de prueba de integración Int_6

Número de caso de prueba: Int_6	
Funcionalidad a integrar:	Modalidad de servicio
Condiciones de ejecución:	Se debe crear la modalidad de servicio: especial.
Descripción de la prueba:	Comprobar que cuando se le asigne la modalidad de servicio a un servicio determinado y se llenen los datos, se liste la modalidad.
Entrada/Pasos de ejecución:	El usuario se dirige a la agrupación funcional Configuración, accede a la funcionalidad Precio de servicio y escoge la opción de crear en el área de íconos flotantes. Llena los datos del servicio y escoge la modalidad del mismo
Resultado esperado:	Se lista la modalidad
Evaluación:	Satisfactoria

Tabla 16. Caso de prueba de integración Int_7

Número de caso de prueba: Int_7	
Funcionalidad a integrar:	Tipo de medio básico
Condiciones de ejecución:	Se debe crear el tipo de medio básico: cama.
Descripción de la prueba:	Comprobar que cuando aparezca la interfaz para asignar medios básicos a un inmueble se pueda escoger el tipo de medio básico: cama.
Entrada/Pasos de ejecución:	El usuario se dirige a la agrupación funcional Configuración, accede a la funcionalidad Asignar medios básicos, escoge la opción de configurar en el área de íconos internos. Cuando aparezca la interfaz se debe seleccionar el tipo de medio básico cama.
Resultado esperado:	Se lista el tipo de medio básico: cama
Evaluación:	Satisfactoria