Universidad de las Ciencias Informáticas

Facultad 1

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

Sistema de Gestión de hospedaje para el Complejo Residencial de la Universidad de las Ciencias Informáticas

Autor(es):

Raúl López Melcón

Tutor(es):

MSc. Prof. Inst. Leiny Amel Pons Flores

Ing. Prof. Inst. Leovan Peña Serrano

La Habana, Junio de 2020

DECLARACIÓN DE AUTORÍA

Declaración de autoría:

Declaro por este medio que yo **Raúl López Melcón**, con carné de identidad **96100900401** soy el autor principal del trabajo titulado "**Sistema de Gestión de hospedaje para el Complejo Residencial de la Universidad de las Ciencias Informáticas.**" y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste firmamos la presente a los _____ días del mes de _____ de _____

Autor:_____

Tutor:_____

AGRADECIMIENTOS

Agradecimientos

Quiero agradecer primeramente a mi familia y amigos, especialmente a mis padres por su apoyo incondicional durante estos 5 años de estudio y esfuerzo. También agradezco a mis tutores Leiny y Leován por sus consejos y sapiencia ya que sin ellos no hubiera logrado todo esto. Agradezco además el apoyo de mis compañeros de clase, particularmente a mi gente del apartamento 126203 a los cuales les deseo que también culminen sus estudios satisfactoriamente.

DEDICATORIA

Dedicatoria

Este trabajo va dedicado a mis padres, especialmente a mi mamá que siempre me apoyó durante estos 5 años. También dedico este trabajo a mi tía que aunque está lejos, siempre ha estado al tanto de mi desempeño en la carrera. Este trabajo también va dedicado a mis tutores por su apoyo y por su entrega en este momento complejo por la pandemia de la COVID-19 que azota al mundo y afecta a nuestro país.

RESUMEN

Resumen

Los sistemas de gestión de información constituyen una poderosa herramienta en el funcionamiento de las entidades y organismos a nivel mundial. Estos se han convertido en uno de los principales pilares en la gestión de las empresas contribuyendo a lograr un eficiente funcionamiento. Se desarrollan con el objetivo de ayudar al proceso de toma de decisiones y proporcionan altos niveles de confianza. El presente trabajo contiene la investigación y desarrollo de un sistema de gestión de información para el Hotel Universitario de las Ciencias Informáticas (UCI), dirigido al control y procesamiento de la información referente a las reservaciones, la administración de los diferentes componentes de la entidad, los clientes, entre otros. La aplicación desarrollada sobre el marco de trabajo o *framework* Laravel en su versión 6.11 permite llevar a cabo la gestión de la información de los clientes hospedados, las incidencias reportadas y elementos como la disponibilidad de habitaciones y la creación de los distintos reportes. Para el desarrollo de la investigación se empleó la metodología de desarrollo AUP en su variante UCI, posibilitando así la documentación de cada una de las etapas de vida del desarrollo del software. Además se definieron las tecnologías y herramientas adecuadas para concretar la implementación de la aplicación propuesta. Una vez concluido el desarrollo del sistema se aplicaron pruebas necesarias para verificar su correcto funcionamiento. Como resultado final se obtuvo una aplicación que es capaz de gestionar de forma eficiente la información que se genera en el Hotel Universitario de la UCI.

Palabras claves: gestión, herramientas, hospedaje, información, tecnologías.

ÍNDICE

Índice de contenidos

ÍNDICE

ÍNDICE

INTRODUCCIÓN

Introducción

El desarrollo de las Tecnologías de la Información y la Comunicación (TIC) ha traído consigo cambios significativos en la sociedad. La puesta en práctica de las TIC incide en numerosos ámbitos de la vida humana, en términos teóricos y de gestión cotidiana. El elemento más representativo de las nuevas tecnologías es sin duda el ordenador y más específicamente, internet, que ha supuesto un salto cualitativo de gran magnitud, cambiando y redefiniendo los modos de conocer y relacionarse del hombre.

Dentro de los numerosos sectores que se benefician del uso de las TIC encontramos al turismo. Un uso generalizado de estas tecnologías en este sector lo conforman los sistemas o software de gestión de la información hotelera (SGIH), los cuales han evolucionado la forma de trabajo de las empresas turísticas (hoteles, compañías turoperadoras, entre otras) con respecto a los datos que manejan. Un sistema de gestión de información hotelera no es más que un software que a partir de los datos de la institución hotelera u organismo del sector donde se implemente, los informatiza o digitaliza y luego los gestiona o trabaja con estos de una forma más rápida permitiendo la automatización de los procesos que se realicen en dicha entidad, ya sea el hospedaje y recepción de clientes, la administración del hotel (disponibilidad de apartamentos, confort de los mismos, etc.). La principal característica o función primordial de los SGIH es la automatización de los procesos y la informatización de los datos para un uso más eficiente de la información que se maneja, logrando así mejores tiempos de respuesta y ejecución de los procesos de las empresas y entidades hoteleras y simplificando los procesos empresariales (1).

A nivel global, los sistemas de gestión hotelera le han proporcionado a las empresas e instituciones turísticas de todo el mundo un nuevo punto de vista dirigido a la información como principal activo. En la actualidad, la información ha llegado a convertirse en un recurso de gran valor e importancia para determinar la competitividad y sustento a los procesos de toma de decisiones, siendo esta el único elemento capaz de crear conocimiento y satisfacer las necesidades de las personas (tanto clientes como trabajadores) y las instituciones, constituyendo así un pilar indispensable para el desarrollo y control de las entidades. La información y el tiempo son elementos importantes para plantear directrices organizacionales, las cuales son necesarias para lograr una mayor eficiencia y disminuir el consumo de tiempo de procesamiento de la

información y, de esta forma, tomar decisiones oportunas y coherentes que conlleven a un mayor reconocimiento, prestigio y calidad de los servicios hoteleros en el mundo (2).

Cuba no está ajena al desarrollo de los sistemas de gestión hotelera debido a que la dirección de la Revolución definió al turismo y al desarrollo de la informática y las comunicaciones como elementos de gran importancia para el país en aras de mejorar la economía y por consiguiente el bienestar del pueblo. Es por eso que actualmente existen diversas empresas de desarrollo de software cubanas como DATYS y el Grupo de Electrónica del Turismo (GET) afiliada al Ministerio del Turismo (MINTUR) que crean productos de alcance nacional con una gran calidad y servicios de excelencia (3). También destaca en el desarrollo de software (no solo para el turismo sino de forma general) la Universidad de las Ciencias Informáticas (UCI).

La UCI, proyecto pensado y llevado a realidad en el año 2002 por nuestro líder histórico: el Comandante en Jefe Fidel Castro Ruz constituye un motor impulsor en este ámbito situándose a la vanguardia en el proceso de informatización de la sociedad. Esta universidad ha formado en sus primeros 15 años miles de profesionales altamente capacitados en varias esferas del campo de las tecnologías lo que los convierte en individuos de alta capacidad y competencia los cuales pueden desenvolverse tanto dentro como fuera del país con un alto nivel de profesionalismo. Estos resultados se deben a que la UCI posee diversos centros de producción que complementan la formación académica de los estudiantes ya que estos se involucran en proyectos reales que les brindan una valiosa experiencia profesional aún sin haber terminado los estudios (4).

La universidad cuenta con numerosos servicios que se brindan a estudiantes, profesores y visitantes tanto nacionales como internacionales que llegan al centro de altos estudios ya sea por eventos científicos, ferias, conferencias, entre otros. Del servicio de atención a los huéspedes tanto nacionales como extranjeros que están de visita en esta institución docente se encarga el Complejo Residencial de la UCI.

En el Complejo Residencial el proceso de registro y hospedaje de los clientes se realiza utilizando herramientas ofimáticas. Los trabajadores del complejo residencial son personas responsables con su trabajo pero no están exentos de errores producto a la manipulación manual de la información del hospedaje de los clientes. Esto se debe a que al registrar la información de los inquilinos puede cometerse errores

tales como valores incorrectos de los datos personales, dígase nombre, apellidos, carnet de identidad, dirección, teléfono, entre otros elementos. Se ha dado el caso en que por una confusión o descuido se anota la reservación de un mismo cliente más de una vez, esto provoca que exista información duplicada y conlleve al mal manejo de las estadísticas de disponibilidad que afectan el posterior proceso de facturación, procedimiento que también se realiza manualmente. De igual forma, para conocer de los huéspedes alojados, habitaciones disponibles y otras informaciones derivadas del proceso de hospedaje, es necesario revisar en el documento. Esto se realiza con el objetivo de conocer su disponibilidad debido a que toda esta información está registrada en un documento, por tanto, es imposible realizar algún tipo de consulta automática para conocer directamente cuáles son las habitaciones disponibles y/o las condiciones de confort de estas. De la misma manera se realiza la facturación de los clientes hospedados, que tributan ingresos a la universidad, la cual se realiza revisando uno por uno las personas alojadas en el Complejo Residencial y calculando de esta forma el tiempo de permanencia real en la instalación. Cabe destacar que tampoco existe un sistema de reportes de incidencia en el caso de que en algún apartamento del complejo residencial se produzca una avería o daño producto de un hecho ocasionado por el cliente o descubierto por él o los trabajadores. En esos casos se le avisa a la carpeta o a algún trabajador del centro.

Teniendo en cuenta lo expuesto anteriormente se puede afirmar que, a pesar de que el Complejo Residencial funciona de forma correcta, no lo hace de la mejor forma ni de la manera más ágil posible. De ahí que se identifica como **problema de la investigación**: ¿Cómo contribuir a mejorar el proceso de hospedaje en el Complejo Residencial de la Universidad de las Ciencias Informáticas?

Luego de analizada la información del problema se establece como **objetivo general**: Desarrollar un sistema de gestión de información de hospedaje para mejorar este proceso del Complejo Residencial de la Universidad de las Ciencias Informáticas empleando tecnologías de desarrollo web.

Para complementar al **objetivo general** se establecen una serie de **objetivos específicos**, los cuales son:

1. Identificar el marco teórico conceptual referente a los sistemas de gestión de información de hospedaje.
2. Caracterizar las soluciones de sistemas de gestión de información de hospedajes similares

existentes, a nivel nacional e internacional.

3. Seleccionar la metodología de desarrollo, las herramientas y tecnologías a utilizar durante el desarrollo del sistema de gestión de información de hospedaje.
4. Definir las características y el diseño del sistema de gestión de información de hospedaje a implementar.
5. Implementar el sistema de gestión de información de hospedaje.
6. Validar el correcto funcionamiento del sistema de gestión de la información de hospedaje implementado mediante pruebas de software.

Estos objetivos específicos, conjuntamente con la solución informática, responderán a las siguientes **Preguntas Científicas:**

1. ¿Cuáles son los referentes teóricos de la gestión de la información de hospedaje que genera el trabajo del Complejo Residencial de la UCI?
2. ¿Cuáles son las características que deben cumplir los procesos que se desarrollan en el Complejo Residencial de la UCI para su futura informatización?
3. ¿Cómo diseñar un sistema de gestión de información de hospedaje que mejore el proceso de hospedaje el Complejo Residencial de la UCI?
4. ¿Cómo implementar el sistema de gestión de información de hospedaje para mejorar el proceso de hospedaje del Complejo Residencial de la UCI?
5. ¿La aplicación desarrollada contribuye a la gestión de la información de hospedaje del Complejo Residencial de la UCI?

Se define como **objeto de estudio** los sistemas de gestión de información de hospedaje específicamente en el **campo de acción** los sistemas de gestión de información de hospedaje en la UCI. En la investigación se utilizaron diferentes métodos de investigación, tanto teóricos como empíricos, los cuales se describen a continuación:

Métodos Teóricos

- Analítico-Sintético: Este método fue utilizado al estudiar la documentación especializada que permitió la extracción de los elementos más importantes relacionados con los sistemas de gestión permitiendo arribar a conclusiones en la investigación así como determinar las características de los requisitos a implementar en la solución (5).

Métodos Empíricos

- Monitoreo de proyectos: Este método fue utilizado para analizar las aplicaciones existentes relacionadas con la gestión hotelera para utilizarlas como referencia a la hora de realizar la solución propuesta.

- Tormenta de ideas: Fue empleado para obtener las necesidades del cliente y realizar el levantamiento de los requisitos con los que debe cumplir la aplicación.

- Observación participativa: Este método empírico fue utilizado para tener una mejor idea por parte del autor de cómo funciona la entidad a la que se le va a efectuar la solución informática. El autor visitó dicho centro, observó cómo se realizan los procesos y así tuvo una mejor idea de las características de la solución a implementar.

Estructura del documento

El Trabajo de Diploma está estructurado de la siguiente manera: introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas y glosario de términos.

Capítulo 1: “Fundamentos teóricos”

En este capítulo se realiza la elaboración del marco teórico donde se exponen los conceptos asociados a la solución de la problemática y las diferentes soluciones existentes a nivel mundial. De igual manera se describen y caracterizan las herramientas y el lenguaje de programación para el diseño e implementación de la solución, así como la metodología de desarrollo más recomendable.

Capítulo 2: “Análisis y diseño del sistema”

INTRODUCCIÓN

Se exponen las principales características y cualidades de la solución a implementar, además se identifican los requisitos, se escoge la arquitectura y los patrones de diseño, se diseñan las clases y se detallan los pasos de la metodología propuesta, incluyendo varios de sus artefactos y diagramas.

Capítulo 3: “Implementación y pruebas del sistema”

Se muestra la situación física de los distintos componentes lógicos desarrollados a través del modelo de despliegue y la organización del sistema mediante el modelo de componentes. Por último se analiza el funcionamiento del sistema desarrollado aplicándole las pruebas necesarias para demostrar que la solución es correcta.

CAPÍTULO 1

Capítulo 1: Fundamentos teóricos

En el presente capítulo se presentan los conceptos importantes respecto a los sistemas de gestión de información de hospedaje. Se realiza un estudio de sistemas existentes en el mundo y en Cuba orientados a la gestión de la información de hospedajes. Se fundamenta además, mediante una detallada descripción, la metodología, herramientas y tecnologías utilizadas en el desarrollo de la propuesta de solución al problema planteado.

1.1 Principales conceptos asociados al tema de investigación

Gestión de información

Gestión de la información (GI) es la denominación convencional de un conjunto de procesos por los cuales se controla el ciclo de vida de la información, desde su obtención (por creación o captura), hasta su disposición final (su archivo o eliminación). Tales procesos también comprenden la extracción, combinación, depuración y distribución de la información a los interesados. El objetivo de la gestión de la información es garantizar la integridad, disponibilidad y confidencialidad de la información. En el contexto de las organizaciones, la gestión de la información se puede identificar como la disciplina que se encargaría de todo lo relacionado con la obtención de la información adecuada, en la forma correcta, para la persona indicada, al coste adecuado, en el momento oportuno, en el lugar apropiado y articulando todas estas operaciones para el desarrollo de una acción correcta. En este contexto, los objetivos principales de la Gestión de la Información son: maximizar el valor y los beneficios derivados del uso de la información, minimizar el coste de adquisición, procesamiento y uso de la información, determinar responsabilidades para el uso efectivo, eficiente y económico de la información y asegurar un suministro continuo de la información. El uso del término es extendido cuando se quiere hacer énfasis en un modelo de gestión documental que, además de los elementos tradicionales, involucra tecnología de la información y la comunicación (TIC), en la organización, almacenamiento, y recuperación de información. En este contexto, un experto en GI deberá, además de poseer la competencia de archivística, tener competencias en áreas relacionadas con las TIC tales como redes de computadores, criptografía, administración de sistemas operativos y servidores (6).

CAPÍTULO 1

Sistema de información

Cuando se habla de un sistema de información (SI) se refiere a un conjunto ordenado de mecanismos que **tienen como fin la administración de datos y de información**, de manera que puedan ser recuperados y procesados fácil y rápidamente. Todo sistema de información se compone de **una serie de recursos interconectados y en interacción**, dispuestos del modo más conveniente en base al propósito informativo trazado, como puede ser recabar información personal, procesar estadísticas, organizar archivo, etc. Estos recursos pueden ser los recursos humanos, datos, actividades y recursos informáticos. Un sistema de información no es lo mismo que un sistema informático, si bien estos últimos constituyen a menudo el grueso de los recursos de un sistema de información. Pero existen muchos otros métodos para los sistemas de información, que no necesariamente pasan por la informática (7).

Características de un sistema de información

Un sistema de información se caracteriza principalmente por la eficiencia al procesar los datos en relación al área de acción. Los sistemas de información se alimentan de los procesos y herramientas de estadística, probabilidad, inteligencia de negocio, producción, *marketing*, entre otros para llegar a la mejor solución. Un sistema de información se destaca por su diseño, facilidad de uso, flexibilidad, mantenimiento automático de los registros, apoyo en toma de decisiones críticas y mantener el anonimato en informaciones no relevantes. Los componentes que forman un sistema de información son:

1. la entrada: por donde se alimentan los datos,
2. el proceso: uso de las herramientas de las áreas contempladas para relacionar, resumir o concluir,
3. la salida: refleja la producción de la información, y
4. la retroalimentación: los resultados obtenidos son ingresados y procesados nuevamente.

Tipos de sistemas de información

En la cultura organizacional, existen varios tipos de sistemas de información según el nivel operacional en que se utilicen. Algunos de los sistemas más comunes se encuentran a continuación:

- para procesamiento de datos (TPS: *Traditional processing system*): nivel operativo, destinado a procesar grandes volúmenes de información alimentando grandes bases de datos.
- sistema de expertos o basados en el **conocimiento** (KWS: *Knowledge working systems*): nivel operativo, selecciona la mejor solución para el problema presentado.
- para la administración y gerenciales (MIS: *Management information systems*): nivel administrativo, gestiona y elabora informes periódicos.
- Para la toma de decisiones (DSS: *Decision support systems*): nivel estratégico, se destaca por su diseño e inteligencia que permite una adecuada selección e implementación de proyectos (8).

Sistema de gestión de información

Un **sistema de gestión de información (SGI)** es una herramienta que permite optimizar recursos, reducir costes y mejorar la productividad en tu empresa. Este instrumento de gestión te reportará datos en tiempo real que permitirán tomar decisiones para corregir fallos y prevenir la aparición de gastos innecesarios. Los **sistemas de gestión** están basados en normas internacionales que permiten controlar distintas facetas en una empresa, como la calidad de su producto o servicio, los impactos ambientales que pueda ocasionar, la seguridad y salud de los trabajadores, la responsabilidad social o la innovación (9).

Ventajas de los SGI

1- Ahorro de esfuerzo y costos
2- Correcta utilización de recursos tangibles e intangibles
3- Mejor desempeño financiero
4- Conocimiento detallado de la información en tiempo
5- Disminución en el tiempo de respuesta (10).

Sistema gestión de información hotelera

Los sistemas de gestión de información hotelera (SGIH) se han convertido en una herramienta imprescindible para optimizar los procesos administrativos e incrementar las reservas. Estos sistemas no solo permiten automatizar las operaciones cotidianas del hotel, sino que también mejoran la experiencia de

los huéspedes desde el mismo momento en que hacen la reserva hasta que vuelven a casa. Dentro de las funciones principales de los SGIH se encuentran:

- Sistemas de gestión hotelera. Este tipo de software facilita la administración de las actividades diarias del hotel, desde aceptar las reservas hasta gestionar las cancelaciones y hojas de registro.
- Gestores de canales. Se trata de programas de distribución que conectan a los hoteleros con distintos agentes, los cuales tienen acceso permanente y en tiempo real a la disponibilidad del alojamiento.
- Motores de reserva online. Esta herramienta permite que los hoteles acepten reservas online directamente en el sitio web corporativo y gestionan su pago.
- Herramientas de precios. Este programa permite implementar una estrategia de gestión de ingresos más eficaz.
- Herramientas para crear sitios web. El sitio web del hotel representa un escaparate virtual que los posibles huéspedes visitan para decidir si van a alojarse en él o no (11).

Los sistemas de gestión de información hotelera proporcionan numerosas ventajas. Entre las más sobresalientes se encuentran:

1. Aumento de la productividad en el área de recepción y en la gestión de las habitaciones
2. Acceso fácil y flexible a la información
3. Ahorro de dinero y optimización de los ingresos
4. Integración con otras aplicaciones tecnológicas en tiempo real
5. Mejoras en la satisfacción del huésped

Cliente

Un **cliente** es aquella persona que a **cambio de un pago** recibe **servicios** de alguien que se los presta por ese concepto. Del latín "**cliens**" nos encontramos en la historia a un cliente como aquel bajo la **responsabilidad** de otro, este otro ofrecía servicios de **protección, transporte** y **resguardo** en todo momento, las indicaciones se debían cumplir bajo regímenes específicos de orden para que pudieran ser

ejecutadas tal cual al pie de la letra. Desde el punto de vista de la economía, es una **persona que utiliza o adquiere, de manera frecuente u ocasional, los servicios o productos** que pone a su disposición un profesional, un comercio o una empresa (12).

Hospedaje

El término **hospedaje** proviene de la palabra hospedar, recibir huéspedes en un propio albergue. Atender a alguien con un hospedaje, es decir, con la posibilidad de dormir bajo techo es una de las más características atenciones que puede tener un ser humano con otro, y en muchos casos este hospedaje puede ser desinteresado y gratuito dependiendo de quién sea el receptor del mismo. Bajo el mismo término también se puede designar al lugar específico de albergue, ya sea este una casa, un edificio, una cabaña o un departamento. Sin embargo, en la actualidad, la palabra hospedaje se relaciona principalmente con el brindar tal servicio a cambio de una tarifa o dinero de acuerdo a la calidad del lugar como también a otros servicios complementarios(13).

Huésped

Con origen en el latín "***hospes***", la palabra **huésped** describe al **individuo** que se encuentra **alojado u hospedado en un hogar ajeno o en la habitación de un hotel.** La idea de huésped es muy común en el ámbito del turismo. En este sentido, toda la rama del alojamiento se basa justamente en la presencia de huéspedes momentáneos que ocupan espacios, conocidos popularmente como hoteles, que no son los propios pero que les sirven para estar bajo techo y cómodos cuando se encuentran lejos de casa, ya sea por trabajo o por placer (14).

1.2 Soluciones existentes

La gestión de información, unido al progresivo desarrollo tecnológico, condiciona que muchas instituciones y entidades en el aspecto informativo, presenten una excesiva centralización de la información y el flujo abundante de documentos impresos; y sucede además que quienes necesitan la información no disponen de ella en el momento y espacio adecuados. Los directivos con frecuencia se encuentran abrumados por documentos e informaciones innecesarias; en muchos casos, se dispone de software y plataformas

incompatibles entre ellas en una misma institución y se desaprovechan los espacios y recursos tecnológicos. Es por ello que surgen los sistemas que gestionan la información (SGI) provocando un avance cuantitativo en la gestión del conocimiento empresarial y despertando la admiración y reconocimiento de todos en cuanto a su importancia para la ayuda a la toma de decisiones. Un estudio sobre diferentes SGI vinculados a la gestión de hospedajes existentes en el mundo detalló algunas soluciones internacionales como QuoHotel, el motor de reservas iBizi y Hotelerum y como soluciones cubanas eHotel de la empresa DATYS y la suite ZUN del Grupo de electrónica del turismo GET, los cuales a continuación se describen puntualizando los aspectos más importantes de cada una de ellas.

1.2.1 Soluciones existentes en el mundo

QuoHotel

QuoHotel es un sistema para hoteles desarrollado por Quonext Tourism, empresa española del Grupo Quonext especializada en soluciones para el sector turístico (hoteles, aeropuertos, aerolíneas, parques temáticos…) el cual ha realizado más de 700 implantaciones de software a nivel mundial. Todas las soluciones desarrolladas por Quonext Tourism, incluyendo QuoHotel, están desarrolladas sobre Microsoft Dynamics NAV. Es un sistema funcional y de vanguardia que incrementa la eficacia de toda la operación hotelera. Es un software de gestión desarrollado para cadenas hoteleras y hoteles independientes que también cubre necesidades adicionales como gestión de spa, balnearios, gimnasios, salas, eventos, golf, entre otros. QuoHotel ofrece una funcionalidad rica y totalmente integrada, que comprende todas las áreas de gestión empresarial del hotel, desde la recepción hasta la gestión financiera, así como la gestión del sistema POS, almacenamiento, compras centrales, centro de reservas, lealtad, calidad, *Business Intelligence*, CRM, etc. QuoHotel puede satisfacer las necesidades de su negocio hotelero, ya sea un complejo hotelero de vacaciones, un hotel urbano, un complejo turístico, el tiempo compartido o cualquier tipo de cadena hotelera (15).

Características de QuoHotel

- Incremento de los ingresos aumentando la ocupación.

- Optimiza la estrategia de precios de venta.
- Gestión de la reputación online: mejora del impacto positivo que repercute directamente en la ocupación.
- Integrar todas las áreas en una sola aplicación ahorrando en costes de gestión.
- Agilizar las operaciones de *checkin* y *checkout*.

Motor de reserva iBizi

iBizi es un software para gestionar las habitaciones y clientes de un sitio de alojamiento como hoteles, posee una forma intuitiva y funciones que mejoran la gestión de las reservaciones y disponibilidad de habitaciones. Al estar alojado en la nube, es funcional y accesible desde cualquier sitio y dispositivo. Dentro de sus características se encuentran:

1- Crea una reserva de varias habitaciones en un solo paso.
2- Gestión de clientes, empresas y agencias a los que vincular la reserva y habitación.
3- Se puede desglosar en distintas facturas los consumos, servicios y habitaciones de una única reserva (16).

Sirvoy

Sirvoy es una plataforma de gestión hotelera estadounidense basada en la nube adecuada para hoteles, posadas, tiempos compartidos y más. Las características clave incluyen la gestión de la oficina y la propiedad, la gestión de reservas y reservas, la limpieza y la gestión de las relaciones con los huéspedes. El sistema de gestión de reservas de Sirvoy puede integrarse en un sitio web de usuario existente, y los huéspedes pueden reservar habitaciones utilizando dispositivos móviles o de escritorio. El motor de reservas admite varios idiomas y ofrece complementos adicionales para Wordpress y Facebook. Los usuarios pueden configurar campos de entrada y recibir alertas para nuevas reservas. Las herramientas de administración de reservas incluyen una descripción general del calendario, reservas grupales y divididas, PDF de reserva imprimibles, un registro de historial, estadísticas y programación de mantenimiento (17).

1.2.2 Soluciones existentes en Cuba

CAPÍTULO 1

Software eHotel

El software eHOTEL desarrollado por DATYS, ofrece una solución integral para la administración de un hotel o una multipropiedad de hoteles y resorts en un ambiente amigable y flexible. Engloba las alternativas operacionales más utilizadas con una interfaz sencilla e intuitiva. Cuenta con una amplia variedad de soluciones para Cadenas Hoteleras, que incluye:

- *Data Warehouse* (Almacén de Datos).
- Gestión Centralizada del Historial de Huéspedes y Empresas.
- Procesos de *checkin* y *checkout* y gestión de habitaciones con gran flexibilidad.
- Reservaciones individuales y de grupo.
- Consulta en línea de la disponibilidad.
- Facturación y generación de información histórica y estadística (18).

Suite ZUN

La Suite ZUN es un sistema de gestión de entidades hoteleras y extrahoteleras. Aplicable a entidades del MINTUR y otros ministerios. Integrado por módulos interrelacionados, flexibles, parametrizables y adaptables a los requisitos de cualquier entidad, independientemente de su complejidad. Funcionamiento en entorno cliente servidor, desarrollado sobre software propietario, emplea como gestor de base de datos SQL 2008, corre sobre plataforma de Windows. Está integrado por varios subsistemas o módulos como son:

ZUNsa: Módulo de seguridad que constituye la herramienta del informático o administrador del sistema para gestionar la configuración de la entidad para el uso de los módulos utilizados.

ZUNpms: Módulo de gestión *fron*t hotelero, totalmente integrado abarca un conjunto de aplicaciones como: contratación, reserva y registro de clientes, facturación de agencias y extras.

ZUNmp: Módulo multihotel que permite el trabajo en aquellas instalaciones que resulta necesario trabajar con más de un módulo de Front Hotelero (ZUNpms) enlazados a una misma contabilidad (ZUNacc).

ZUNacc: Módulo de Contabilidad que automatiza la gestión contable en instalaciones hoteleras y extrahoteleras, con características multiempresa, permitiendo obtener información de forma independiente y consolidada.

ZUNcr: Módulo de central de reservas que es una aplicación web orientada a la comercialización de productos y servicios (19).

Luego de analizadas las soluciones nacionales e internacionales, el autor tomó una serie de criterios de comparación para observar cuáles de estas soluciones se adecua a lo que necesita:

Tabla 1 Estudio de homólogos: comparación de criterios. Fuente: elaboración propia.

Sistemas	Autoría	Gestión de hospedaje	Búsqueda de información	Control de incidencias	Código abierto	Almacenamiento externo
eHotel	Sí	Sí	Sí	No	No	Sí
Suite ZUN	Sí	Sí	Sí	No	No	Sí
iBizi	Sí	Sí	Sí	Sí	No	Sí
QuoHotel	Sí	Sí	Sí	Sí	No	Sí
Sirvoy	Sí	Sí	Sí	Sí	No	Sí

1.2.3 Conclusiones del análisis de soluciones existentes

A partir del estudio realizado sobre las soluciones existentes, se concluyó que no son factibles para la gestión de información de hospedaje que se desea realizar. La principal razón por la que no constituyen una solución factible es porque no se ajustan a las necesidades del cliente puesto que no están enfocados al negocio detallado que se enuncia en la descripción de la situación problemática planteada. Su adopción constituiría un esfuerzo en redefinir las funcionalidades requeridas mucho mayor que desarrollar una solución desde su inicio. Además, los sistemas analizados están implementadas sobre tecnologías privadas y su uso constituiría un gasto elevado por concepto de pago en licencias de software o no estarían acorde a los esfuerzos desarrollos por el país de migrar a tecnologías libres. Se incluyó en la comparación el almacenamiento externo y aunque cumplen con este parámetro no lo hacen como se espera en la solución. El motivo es que emplean almacenamiento en la nube ya que necesitan publicar sus datos para los turoperadores y sitios de información turística y eso no es necesario para el sistema. Por estas razones se

decide implementar una nueva solución para el hotel universitario teniendo en cuenta las características y funcionalidades presentes en las soluciones estudiadas como base para la implementación.

1.3 Metodología, herramientas y tecnologías

A continuación se presentan las principales herramientas, tecnologías y metodologías utilizadas para dar solución al problema planteado. Para su selección se tuvieron en cuenta las necesidades existentes, tendencias actuales y el entorno donde se desplegará la aplicación.

1.3.1 Metodología de desarrollo de software

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas, herramientas y soporte documental para el desarrollo de un producto (software). Se basan en una combinación de los modelos de proceso genéricos (cascada, evolutivo e incremental.). Son para estructurar, planear y controlar el proceso de desarrollo del software. Constituyen una guía que define las tareas y actividades que se deben realizar para obtener un software de buena calidad. Además, una metodología de desarrollo de sistemas se refiere al marco que se utiliza para estructurar, planificar y controlar el proceso de desarrollo de un sistema de información. Una amplia variedad de tales marcos han evolucionado a lo largo de los años, cada uno con sus fortalezas reconocidas y propias debilidades (20).

AUP-UCI

Como metodología utilizada para guiar el proceso de desarrollo de software en la presente investigación se seleccionó el Proceso Unificado Ágil (AUP por sus siglas en inglés) variación UCI. El autor justifica la elección debido a que la metodología variación de AUP para la UCI está formada por tres fases, (Inicio, Ejecución y Cierre) para el ciclo de vida de los proyectos de la universidad, las cuales contienen las características de las cuatro fases (Inicio, Elaboración, Construcción y Transición) propuestas en AUP. Las características de las fases de la metodología de la universidad son:

- Inicio: En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
- Ejecución: En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
- Cierre: En el cierre se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Disciplinas de variación AUP-UCI

- Modelado de negocio: El modelado del negocio es la disciplina destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito. Para modelar el negocio se proponen las siguientes variantes:

  1. Casos de Uso del Negocio (CUN)
  2. Descripción de Proceso de Negocio (DPN)
  3. Modelo Conceptual (MC)

- Requisitos: El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto. Existen tres formas de encapsular los requisitos [Casos de Uso del Sistema (CUS), Historias de usuario (HU) y Descripción de requisitos por proceso (DRP)], agrupados en cuatro escenarios condicionados por el modelado de negocio.

- Análisis y diseño: En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura).

- Implementación: En la implementación, a partir de los resultados del análisis y diseño se construye el sistema.
- Pruebas internas: En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posible, componentes de prueba ejecutables para automatizar las pruebas.
- Pruebas de liberación: Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.
- Pruebas de aceptación: Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido (21).

La metodología AUP-UCI cuenta con varios escenarios (cuatro exactamente) que definen las pautas del diseño de la solución. El autor determinó luego de un estudio realizado a cada escenario que se utilice el escenario dos de AUP-UCI.

1.3.2 Lenguaje de modelado de sistemas.

Lenguaje de modelado como soporte a la metodología (UML) 2.0

El autor determinó que para el modelado de sistema empleará el Lenguaje de Modelado Unificado (UML), debido a que es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables (22).

CAPÍTULO 1

El modelo UML consta de dos categorías principales de los elementos que lo integran, cada uno de los cuales pueden ser utilizados para hacer declaraciones sobre los diferentes tipos de elementos individuales dentro del sistema que está siendo modelado. Estas categorías son:

- **Clasificadores**. Un clasificador describe un conjunto de objetos. Un objeto es un individuo con un estado y las relaciones de otros objetos. El estado de un objeto identifica los valores para ese objeto de propiedades del clasificador.
- **Comportamientos.** Un comportamiento describe un conjunto de posibles ejecuciones. Una ejecución es una representación de un conjunto de acciones (potencialmente sobre un cierto período de tiempo) que pueden generar y responder a las ocurrencias de eventos, acceder y cambiar el estado de los objetos (23).

1.3.3 Herramientas CASE

Las herramientas CASE son un conjunto de aplicaciones informáticas, usadas para automatizar actividades del ciclo de vida de desarrollo de sistemas (SDLC). Las herramientas CASE son usadas por los directores de proyectos de software, analistas e ingenieros para desarrollar sistemas de software. Hay un gran número de herramientas CASE disponibles para simplificar varias etapas en el desarrollo del ciclo vital del software, como por ejemplo: herramientas de análisis, diseño de herramientas, gestión de proyectos de herramientas, proyectos de gestión de herramientas de bases de datos y de gestión de herramientas de bases de datos (24).

Según Pressman (25) son un conjunto de herramientas y métodos asociados que proporcionan asistencia automatizada en el proceso de desarrollo del software a lo largo de su ciclo de vida, mediante su uso se reduce el tiempo, costo del desarrollo y mantenimiento del software, así como se mejora su calidad. El uso de herramientas CASE acelera el desarrollo del proyecto con tal de producir los resultados deseados y ayuda a encontrar imperfecciones antes de proseguir con la siguiente etapa del desarrollo de software.

Visual Paradigm 8.0

Es una herramienta CASE multiplataforma que cuenta con versiones gratuitas y provee fácil integración con el resto de las herramientas de desarrollo. Ayuda a una más rápida construcción de aplicaciones de calidad y con menor costo. También permite dibujar todos los tipos de diagramas de clases, generar código desde diagramas y generar documentación (26). Se decidió utilizar esta herramienta en el desarrollo de la aplicación por las múltiples ventajas que posee al brindarle soporte a todo el ciclo de vida de implementación del software a desarrollar.

1.3.4 Lenguajes de programación

Un lenguaje de programación es aquel elemento dentro de la informática que nos permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que pone a disposición del programador para que este pueda comunicarse con los dispositivos hardware y software existentes (27). En este acápite se listan los principales lenguajes de programación empleados para el desarrollo de sistemas de gestión. Se describirán estos lenguajes y al finalizar el autor dará su criterio de elección del lenguaje adecuado para la implementación de la solución.

PHP

PHP (acrónimo recursivo de *PHP: Hypertext Preprocessor*) es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página web resultante. PHP ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes. Puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo (28). El gran parecido que posee PHP con los lenguajes más comunes de programación estructurada, como C y Perl, permiten a la mayoría de los programadores crear aplicaciones complejas con una curva de aprendizaje muy corta. También les permite involucrarse con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones.

CAPÍTULO 1

Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado y usa tipado dinámico. Python es un lenguaje de programación multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional. Otros paradigmas están soportados mediante el uso de extensiones. Python usa el conteo de referencias para la administración de memoria. Otra característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado enlace dinámico de métodos) (29).

JavaScript

JavaScript (a veces abreviado como JS) es un lenguaje ligero e interpretado, orientado a objetos con funciones de primera clase, más conocido como el lenguaje de script para páginas web. Es un lenguaje script multiparadigma, basado en prototipos, dinámico y soporta estilos de programación funcional, orientada a objetos. Además permite ejecutar instrucciones como respuesta a las acciones del usuario, permitiendo la validación de los datos introducidos por los usuarios (30).

jQuery

jQuery es una biblioteca de JavaScript rápida, pequeña, y rica en características. Hace cosas como recorrido y manipulación de documentos HTML, manejo de eventos, animación, y Ajax mucho más simple con un API (en español interfaz de programación de aplicaciones) fácil de usar que funciona a través de una multitud de navegadores. Está catalogado como software libre y de código abierto permitiendo su uso en proyectos libres y privados. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio (31).

Lenguaje de Etiquetado de Hipertexto (HTML)

Lenguaje de Etiquetado de Hipertexto (*HyperText Markup Language*). Es un lenguaje comúnmente utilizado para la publicación de hipertexto en la web y desarrollado con la idea de que cualquier persona o tipo de dispositivo pueda acceder a la información en la web. HTML utiliza etiquetas que marcan elementos y estructuran el texto de un documento (32).

1.3.5 Framework

Un *framework*, en el desarrollo de software, es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto (33).

Laravel

Laravel es un *framework* de código abierto para desarrollar aplicaciones y servicios web con PHP 5. Su filosofía es desarrollar código PHP de forma elegante y simple. Fue creado en 2011 y tiene una gran influencia de *frameworks* como Ruby on Rails, Sinatra y ASP.NET MVC. Laravel tiene como objetivo ser un framework que permita el uso de una sintaxis elegante y expresiva para crear código de forma sencilla y permitiendo multitud de funcionalidades. Intenta aprovechar lo mejor de otros *frameworks* y aprovechar las características de las últimas versiones de PHP. Gran parte de Laravel está formado por dependencias, especialmente de Symfony, esto implica que el desarrollo de Laravel dependa también del desarrollo de sus dependencias (34).

Symfony

Symfony es un marco de trabajo o *framework* muy completo, posee una librería cohesiva de clases escritas en PHP. Provee una arquitectura, componentes y herramientas a los desarrolladores para construir aplicaciones web complejas de forma rápida y a la vez permite proporcionar un mantenimiento constante. Está basado en la experiencia, usa las mejores prácticas del desarrollo web y ha sido un proyecto de Código

CAPÍTULO 1

Abierto por más de cuatro años. Se ha convertido en uno de los *framework* de PHP más populares hoy en día y se ha creado una gran comunidad donde se puede encontrar soporte, documentación y aplicaciones libres (35). Es fácil de configurar y adaptable a sus necesidades. Es independiente del sistema gestor de bases de datos. Permite cambiar con facilidad de Sistema Gestor de Base de Datos (SGBD) en cualquier fase del proyecto. Permite la reutilización de componentes.

Django

Django es un *framework* de desarrollo web de código abierto, escrito en Python, que respeta el patrón de diseño conocido como Modelo–Vista–Controlador. La meta fundamental de Django es facilitar la creación de sitios web complejos. Django pone énfasis en el re-uso, la conectividad y extensibilidad de componentes, el desarrollo rápido y el principio No te repitas (DRY, del inglés *Don't Repeat Yourself*). Python es usado en todas las partes del *framework*, incluso en configuraciones, archivos, y en los modelos de datos (36).

Bootstrap 4

Bootstrap es un *framework* o conjunto de herramientas de software libre desarrollado por Twitter para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales. Bootstrap es modular y consiste esencialmente en una serie de hojas de estilo dinámicas (LESS) que implementan la variedad de componentes de la herramienta. Los desarrolladores pueden adaptar el mismo archivo de Bootstrap, seleccionando los componentes que deseen usar en su proyecto. Los ajustes son posibles en una medida limitada a través de una hoja de estilo de configuración central. Los cambios más profundos son posibles mediante las declaraciones LESS. El uso del lenguaje de hojas de estilo dinámico permite el uso de variables, funciones y operadores, selectores anidados, así como clases *mixin*. El paquete consecuentemente generado ya incluye la hoja de estilo CSS (37).

Algunas características de Bootstrap son las siguientes:

- Permite crear interfaces que se adapten a los diferentes navegadores, tanto de escritorio como *tablets* y móviles a distintas escalas y resoluciones.

- Se integra perfectamente con las principales librerías JavaScript, por ejemplo JQuery.
- Ofrece un diseño sólido usando LESS y estándares como CSS3/HTML5.
- Funciona con todos los navegadores, incluido Internet Explorer.
- Dispone de distintos *layout* predefinidos con estructuras fijas a 940 píxeles de distintas columnas o diseños fluidos en las características detalladas previamente.

1.3.6 Servidor web

Un servidor web o servidor HTTP es un programa informático que procesa una aplicación del lado del servidor, realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente y generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente. El código recibido por el cliente suele ser compilado y ejecutado por un navegador web. Para la transmisión de todos estos datos suele utilizarse algún protocolo. Generalmente se usa el protocolo HTTP para estas comunicaciones, perteneciente a la capa de aplicación del modelo OSI. El término también se emplea para referirse al ordenador que ejecuta el programa (38).

Nginx

Nginx (pronunciado en inglés “engine X”) es un servidor web/proxy inverso ligero de alto rendimiento y un proxy para protocolos de correo electrónico (IMAP/POP3). Es software libre y de código abierto, licenciado bajo la Licencia BSD simplificada; también existe una versión comercial distribuida bajo el nombre de Nginx plus. Es multiplataforma, por lo que corre en sistemas tipo Unix (GNU/Linux, BSD, Solaris, Mac OS X, etc.) y Windows. El sistema es usado por una larga lista de sitios web conocidos, como: WordPress, Netflix, Hulu, GitHub, Ohloh, SourceForge, TorrentReactor y partes de Facebook (como el servidor de descarga de archivos *zip* pesados) (39).

Apache 2.X

Apache 2 es un servidor web multiplataforma de código abierto. Es modular (basado en módulos), donde cada módulo ofrece un grupo de funcionalidades específicas al servidor. Es uno de los servidores web más utilizado en Internet, lo que facilita el acceso a la documentación. Provee un alto nivel de seguridad y

eficiencia, permitiendo además el uso de una versión local, la cual hace posible que el servidor actúe como servidor y cliente al mismo tiempo, creando así la posibilidad de pre visualizar y probar el código mientras este es desarrollado. Apache presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración (40).

1.3.7 Servidor de Base de Datos

Los sistemas de gestión de bases de datos o SGBD (en inglés *Database Management System*, abreviado DBMS) son un tipo de software dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Son grandes proveedores de información para todo tipo de usuarios. Ellos deben ofrecer soluciones de forma fiable, rentable y de alto rendimiento. A estas tres características, se le debe añadir una más: debe proporcionar servicios de forma global y, en la medida de lo posible, independientemente de la plataforma (41).

PostgreSQL 9.3

Es un sistema de gestión de base de datos relacional orientada a objetos, libre y publicado bajo la licencia BSD. Además es el gestor de bases de datos de código abierto más avanzado en la actualidad, ofreciendo control de concurrencia multiversión, soportando casi todas las sintaxis SQL como subconsultas, transacciones y funciones definidas por el usuario, cuenta además con un amplio conjunto de enlaces con lenguajes de programación como C, C++, Java y Python. PostgreSQL está dirigido por una comunidad bien organizada de desarrolladores y organizaciones comerciales, de ahí el nivel de aceptación y de calidad del mismo.

Algunas de las características que posee son: gran escalabilidad, capaz de ajustarse al número de Unidades Centrales de Procesamiento (CPU) y a la cantidad de memoria que posee el sistema de forma óptima, tiene la capacidad de comprobar la integridad referencial, así como también la de almacenar procedimientos en la propia base de datos y la simplificación del proceso de administración de licencias de software, que no es necesario cuando se usa software libre (42).

CAPÍTULO 1

MySQL

MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation y está considerada como la base datos *open source* más popular del mundo, y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web. Es muy utilizado en aplicaciones web, como Joomla, Wordpress, Drupal o PhpBB, en plataformas (Linux/Windows-Apache-MySQL-PHP/Perl/Python), y por herramientas de seguimiento de errores como Bugzilla. Su popularidad como aplicación web está muy ligada a PHP, que a menudo aparece en combinación con MySQL. Es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones (43).

1.3.8 Aplicación gráfica para gestionar el gestor de bases de datos

PgAdmin III

PgAdmin es una popular plataforma de código abierto para el desarrollo con PostgreSQL. Rico en funciones para la administración y desarrollo para PostgreSQL. Puede ser usado en las plataformas Linux, FreeBSD, Solaris, MacOSX y Windows. Está diseñado con el fin de satisfacer la necesidad de todos los usuarios, para escribir simples sentencias SQL y desarrollar complejas bases de datos. La interfaz gráfica soporta todas las características de PostgreSQL y hace más fácil la administración. La aplicación además incluye un editor de sintaxis SQL, editor de código del lado del servidor y una consola. Las conexiones al servidor pueden ser realizadas usando TCP/IP o *Unix Domain Sockets* (para plataformas Unix). Además de tener la posibilidad de usar encriptado SSL para seguridad. No requiere drivers adicionales para conectarse con el servidor de base de datos. PgAdmin es desarrollado por expertos de la comunidad de PostgreSQL alrededor del mundo y se encuentra disponible en una docena de idiomas. Fue liberado bajo licencia libre PostgreSQL (44).

PhpMyAdmin

PhpMyAdmin es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web, utilizando Internet. Actualmente puede crear y eliminar Bases de Datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios y exportar datos en varios formatos.

1.3.9 Entorno integrado de desarrollo IDE

Para la ejecución del sistema se hizo necesario escoger un entorno integrado de desarrollo (IDE) que proporcionara el uso y la integración de todas las tecnologías y lenguajes antes mencionados, por lo que se estudió una serie de IDE con características y funcionalidades afines al lenguaje de programación escogido por el autor.

Visual Studio Code Insiders

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta múltiples lenguajes de programación tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby, PHP; al igual que entornos de desarrollo web como ASP.NET MVC, Django, etc., a lo cual sumarle las nuevas capacidades online bajo Windows Azure en forma del editor Monaco. Visual Studio permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET. Así se pueden crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos, consolas, entre otros (45).

JetBrains PhpStorm 9.0

JetBrains PhpStorm es un IDE multiplataforma para PHP desarrollado sobre la plataforma JetBrains IntelliJ IDEA. Proporciona un editor para PHP, HTML y JavaScript con el análisis de código en la marcha, la prevención de errores y refactorizaciones automáticas para PHP y el código JavaScript. También posee una compatibilidad ajustada y óptima con diferentes *frameworks* cómo son Symfony, Drupal, WordPress, Zend Framework, Laravel, Magento, CakePHP, Yii, y otros *frameworks* de menor trascendencia (JetBrains, 2015).

CAPÍTULO 1

La idea de seleccionar PhpStorm 9.0 como IDE parte de su relación con Symfony, el *framework* de desarrollo escogido, ya que PhpStorm en esta versión le da soporte, facilitando el trabajo del programador con el *framework*. De esta manera se tiene completamiento de código no solo de la sintaxis de PHP sino de las clases creadas que se generen dentro del proyecto (46).

Para resumir las herramientas escogidas por el autor para el desarrollo de la solución se muestra a continuación la siguiente tabla que resume lo anteriormente comentado.

Tabla 2 Herramientas para el desarrollo de la aplicación. Fuente: elaboración propia.

Herramienta CASE	Lenguaje principal	Lenguajes auxiliares	Frameworks	Servidor web	Servidor BD	Gestión BD	IDE
Visual Paradigm 8	PHP 7.3.2	HTML5, CSS3, JavaScript	Laravel 6.11, Bootstrap 4	Apache 2	MySQL	PhpMyAdmin	Visual Studio Code Insiders 1.28

1.4 Conclusiones del capítulo

En este capítulo se esclarecieron los conceptos asociados con el dominio del problema y se definieron las principales tecnologías que contribuirán con el desarrollo de la solución propuesta, aportando así una base para que los investigadores tengan una mejor visión del ámbito donde se desarrolla la investigación. A partir de estos conceptos se encaminó el estudio del estado del arte de sistemas que gestionan información, evidenciándose que no resuelven el problema planteado. Aun así, estos sistemas aportan elementos importantes para el desarrollo del sistema que se propone realizar. Se realizó además un análisis profundo del objeto de estudio, el cual permitió tener una noción más nítida del problema a resolver. Finalmente la selección de las herramientas y tecnologías seleccionadas permitió definir el entorno técnico donde se desarrollará la solución propuesta.

CAPÍTULO 2

Capítulo 2: Análisis y diseño del Sistema de Gestión de hospedaje para el Complejo Residencial de la Universidad de las Ciencias Informáticas

En el capítulo que se presenta a continuación se exponen las principales características y cualidades de la solución a implementar, se describe y detalla todo lo referente al modelo del negocio, se identifican los Requisitos Funcionales y No Funcionales; también se obtiene el diagrama de los Casos de Uso del Negocio y del Sistema conjunto a su descripción textual, así como los diagramas correspondientes a la representación del diseño. Como parte de la definición de los elementos arquitectónicos fundamentales se determina el estilo y el patrón arquitectónico a emplear además se relacionan los patrones de diseño que guiarán la construcción del sistema propuesto. Al concluir este capítulo se contará con una comprensión de los problemas actuales del negocio y una idea pertinente del sistema a desarrollar.

2.1 Modelo Conceptual

Un modelo conceptual es una representación de un sistema, hecho de la composición de conceptos que se utilizan para ayudar a las personas a conocer, comprender o simular un tema que representa el modelo. También es un conjunto de conceptos. Algunos modelos son objetos físicos; por ejemplo, un modelo de juguete que se puede ensamblar y se puede hacer que funcione como el objeto que representa. El término modelo conceptual puede usarse para referirse a modelos que se forman después de un proceso de conceptualización o generalización. Los modelos conceptuales son a menudo abstracciones de cosas en el mundo real, ya sean físicas o sociales. Los estudios semánticos son relevantes para varias etapas de la formación de conceptos. La semántica se trata básicamente de conceptos, el significado que los seres pensantes dan a varios elementos de su experiencia (47).

CAPÍTULO 2

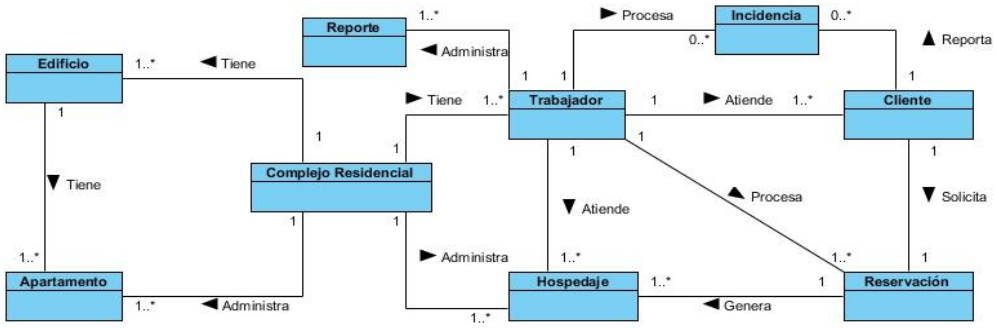


Figura 1. Modelo conceptual del Complejo Residencial UCI. Fuente: elaboración propia.

Definición de los conceptos del modelo

1. **Complejo residencial**: Lugar donde se realizan todas las operaciones del sistema. Aquí intervienen todos los elementos que propician el correcto funcionamiento de la entidad.
2. **Edificio**: Esta estructura tiene como relevancia que almacena los apartamentos donde se alojan los clientes.
3. **Apartamento**: Estructura donde una vez reservado y registrado el cliente, se hospeda hasta la fecha acordada.
4. **Trabajador**: Entidad que representa al personal que labora en el complejo residencial.
5. **Cliente**: Como se definió en el capítulo anterior, es toda persona o grupo de personas que llegan al complejo residencial solicitando sus servicios de hospedaje.
6. **Reservación**: Solicitud que realiza un cliente para hospedarse en el complejo residencial en un tiempo determinado.
7. **Hospedaje**: Registro que contiene los clientes que están alojados en el complejo.
8. **Incidencia**: Suceso que ocurre dentro del complejo que puede influir de forma negativa en el funcionamiento de la entidad.

2.2 Propuesta de solución

CAPÍTULO 2

Para dar solución a la problemática planteada en el capítulo anterior se decide desarrollar un sistema de gestión de la información. Este SGI contempla la administración y el control por parte de los administrativos correspondientes de las reservaciones y clientes hospedados, los edificios pertenecientes al Complejo Residencial, además de los apartamentos y sus condiciones. También se llevará un control de las incidencias que reporten los huéspedes y trabajadores del centro. Para acceder al sistema cada usuario debe autenticarse con su usuario y contraseña. Se asignarán roles a los usuarios y los permisos estarán asignados a los roles. La visibilidad de las funcionalidades en el sistema estará en dependencia de los permisos de cada usuario. Los administradores y trabajadores podrán consultar en cualquier momento la información de los clientes hospedados, las reservaciones realizadas y las incidencias reportadas, pero solo los administradores podrán editar o suprimir el contenido de las reservaciones (en este caso cancelarlas), la información de los clientes hospedados y de las incidencias además de tener acceso a otras funciones. Alguna de las funciones que el administrador podrá ejecutar es la obtención de un reporte (diario, mensual, o anual) de lo que ha sucedido en ese lapso de tiempo (nuevas reservaciones, clientes registrados, nuevas incidencias, entre otras.). Este reporte podrá ser exportado a un formato PDF y luego será descargado para que el administrador pueda tener ese documento a la mano y seguir ejecutando otras tareas en el sistema. Este SGI se implementará siguiendo la arquitectura Modelo-Vista-Controlador (MVC) la cual será detallada en los acápites posteriores.

Para tener una mejor visión de la estructura de la solución se propone el siguiente diagrama que representa la arquitectura de la información del sistema a desarrollar. La arquitectura de información no es más que la disciplina y arte encargada del estudio, análisis, organización, disposición y estructuración de la información en espacios de información, y de la selección y presentación de los datos en los sistemas de información interactiva y no interactiva. La arquitectura de información trata indistintamente del diseño de: sitios web, interfaces de dispositivos móviles o *gadgets*, etc. Su principal objetivo es facilitar al máximo los procesos de comprensión y asimilación de la información, así como las tareas que ejecutan los usuarios en un espacio de información definido (48).

Comentado [L1]: Creo que debemos revisar este término porque lo que veo debajo no es específicamente una arquitectura de información, pero la imagen la vamos a mantener porque quedó muy buena, sólo debemos modificar este párrafo.

CAPÍTULO 2

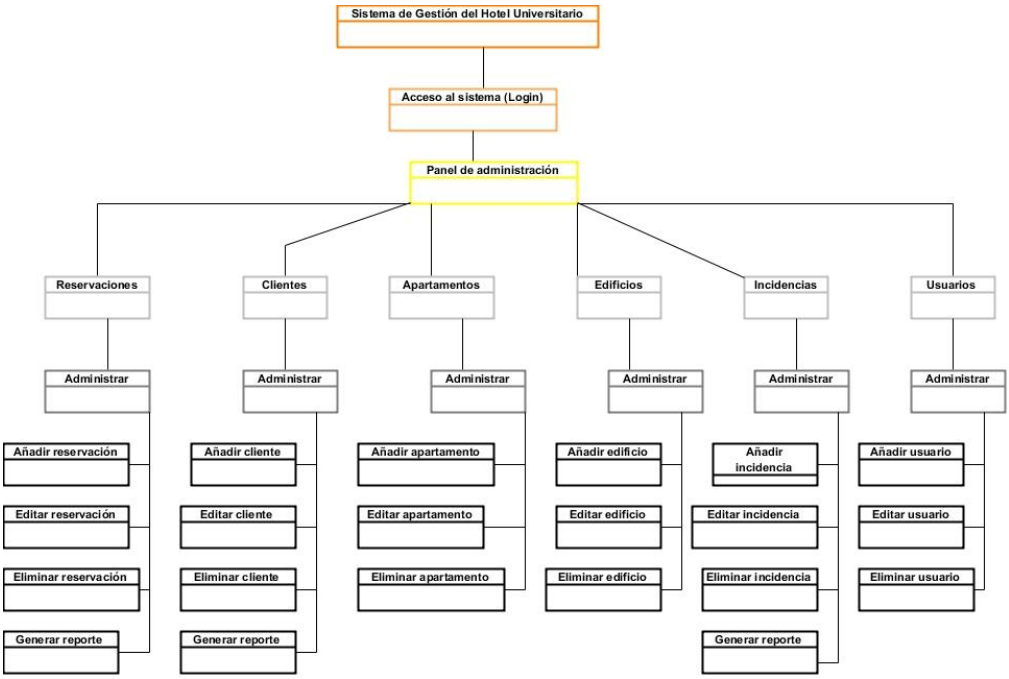


Figura 2. Arquitectura de la información de la propuesta de solución. Fuente: elaboración propia.

2.3 Requisitos de software

Según Pressman y Ramdhani (49) los requisitos de software son las necesidades de los clientes, los servicios que los usuarios desean que proporcione el sistema de desarrollo y las restricciones en las que debe operar. Los requisitos se dividen en Funcionales y No Funcionales y muestran las capacidades o condiciones que el sistema debe cumplir y las propiedades o cualidades que el producto debe tener. En este acápite se listarán los requisitos funcionales y los no funcionales que serán aplicados en la solución a implementar.

2.3.1 Requisitos funcionales

Un **requisito funcional** define una función del sistema de software o sus componentes. Una función es descrita como un conjunto de entradas, comportamientos y salidas. Los requisitos funcionales pueden ser: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que se supone, un sistema debe cumplir. Los requerimientos de comportamiento para cada requerimiento funcional se muestran en los casos de uso. Son complementados por los requisitos no funcionales, que se enfocan en cambio en el diseño o la implementación. Como se define en la ingeniería de requisitos, los requisitos funcionales establecen los comportamientos del sistema (50).

A continuación se muestra en formato de tabla, los requisitos funcionales que serán implementados en la solución informática:

Tabla 3. Requisitos funcionales. Fuente: elaboración propia.

Requerimientos	Descripción	Prioridad/Complejidad
RF1. Autenticar Usuario	Permite la autenticación de cada uno de los usuarios para acceder al sistema	Alta/Baja
RF2. Administrar sistema	Permite agregar un usuario, darle o quitarle privilegios, eliminarlo. Se permitirá crear y eliminar edificios, apartamentos, reservaciones y hospedajes en caso de ser necesario.	Alta/Alta
RF3. Crear usuario **RF4. Mostrar usuario** **RF5. Actualizar usuario** **RF6. Eliminar usuario** **Gestionar usuario (RF 3-6)**	Permite la gestión por parte del administrador de cada uno de los usuarios del sistema. Podrá crear, editar, mostrar y eliminar usuarios, así como asignarle roles.	Alta/Media
RF7. Crear edificio **RF8. Mostrar edificio** **RF9. Actualizar edificio**	Permite la gestión por parte del administrador de cada uno de los edificios del Complejo. Podrá	Alta/Media

RF10. Eliminar edificio **Gestionar edificio (RF 7-10)**	crear, editar, mostrar y eliminar edificios.	
RF11. Crear apartamento **RF12. Mostrar apartamento** **RF13. Actualizar apartamento** **RF14. Eliminar apartamento** **Gestionar apartamento (RF 11-14)**	Permite la gestión por parte del administrador de cada uno de los apartamentos del Complejo. Podrá crear, editar, mostrar y eliminar apartamentos. Los que no tengan rol administrador solo podrán observar el listado de apartamentos.	Alta/Media
RF15. Crear reservación **RF16. Mostrar reservación** **RF17. Actualizar reservación** **RF18. Eliminar reservación** **Gestionar reservación (RF 15-18)**	Permite la gestión por parte del administrador de cada una de las reservaciones del Complejo. Podrá crear, editar, mostrar y eliminar reservaciones. Los que no tengan rol administrador solo podrán ver el listado de las reservaciones.	Alta/Media
RF19. Crear hospedaje **RF20. Mostrar hospedaje** **RF21. Actualizar hospedaje** **RF22. Eliminar hospedaje** **Gestionar hospedaje (RF 19-22)**	Permite la gestión por parte del administrador de cada uno de los registros de hospedaje de clientes del Complejo. Podrá crear, editar, mostrar y eliminar dichos registros. Los que no posean rol administrador podrán crear un registro y listarlos pero no podrán realizar el resto de operaciones.	Alta/Media
RF23. Crear incidencia **RF24. Mostrar incidencia** **RF25. Actualizar incidencia** **RF26. Eliminar incidencia** **Gestionar Incidencia (RF 23-26)**	Permite la gestión por parte del administrador de cada uno de los registros de incidencia del Complejo. Podrá crear, editar, mostrar y eliminar dichos registros. Los que no posean rol administrador podrán crear un registro y listarlos pero no podrán realizar el resto de operaciones.	Alta/Media
RF27. Confeccionar reporte **RF28. Exportar reporte** **RF29. Descargar reporte**	Permite al administrador confeccionar un reporte (diario, semanal, mensual, etc.) de las operaciones realizadas en el	Alta/Media

	complejo en un período de tiempo. Este reporte puede ser exportado a formato PDF para su posterior descarga.	
RF30. Buscar	Permite al usuario del sistema realizar una búsqueda de un elemento específico a través de una frase o palabra específica.	Alta/Media

2.3.2 Requisitos no funcionales

Un **requisito no funcional** o **atributo de calidad** es, en la ingeniería de sistemas y la ingeniería de software, un requisito que determina los criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que éstos corresponden a los requisitos funcionales (51).

A continuación se muestra los requisitos no funcionales divididos por categorías que serán implementados en la solución:

Usabilidad

- RNF1. El diseño del sistema estará enfocado en los usuarios del mismo. Contará con acceso sencillo e intuitivo a las funcionalidades principales (botones y enlaces a las funcionalidades).
- RNF2. Se tendrá en cuenta los errores que cometan los usuarios en la entrada de datos alertándolos que error realizaron.

Fiabilidad

- RNF3. La información estará disponible todo el tiempo.

Diseño e implementación

- RNF4. Lenguaje de programación: PHP 7, Bootstrap 4, JavaScript, HTML5 (lenguaje etiquetado), CSS 3.
- RNF5. Gestor de bases de datos: PhpMyadmin (MySQL).

- RNF6. El sistema puede ser accedido desde cualquier navegador web que tenga soporte para los estándares de la WC3.

Seguridad

- RNF7. Los datos de los usuarios se encontrarán encriptados (contraseñas).
- RNF8. Existirá un sistema de roles el cual controlará el acceso y control de las diferentes funcionalidades del sistema.

Requisitos de software

- RNF9. Para las PC clientes solo deben contar con un navegador que soporte el estándar WC3 aunque se recomienda que estos navegadores estén actualizados a versiones recientes.
- RNF10. Para las PC servidor deben integrarse al gestor de bases de datos PhpMyadmin y el servidor web Apache.

Requisitos de hardware

- RNF11. Para las PC clientes se requerirán máquinas con procesador de velocidad igual o mayor a 2.3 GHz de micro, tarjeta de red a 100 Mb/s o equivalente y 512 Mb de RAM.
- RNF12. Para las PC servidor el hardware donde se instalará el sistema debe poseer al menos una interfaz de red cuya velocidad de transferencia iguale o supere los 100 Mb/s. El servidor donde se instale el sistema debe tener como mínimo un procesador con velocidad igual o mayor a 3.0 GHz, 4 GB de memoria RAM y un espacio libre en Disco Duro de al menos 500 GB.

2.4 Modelo del sistema

El Modelo de Casos de Uso permite describir los RF del sistema, dando lugar a un acuerdo entre el cliente y los desarrolladores de la aplicación. Proporciona una explicación clara y consistente de lo que debería hacer el software, de modo que el modelo se use a lo largo del proceso de desarrollo. Posibilita que se obtenga una base para realizar verificaciones del producto informático, siendo la entrada fundamental para el análisis, el diseño y las pruebas. Los casos de uso son descripciones funcionales del sistema que

permiten definir los límites del software y las relaciones entre la aplicación y el entorno; describen cómo los actores pueden usar un sistema. Especifican una secuencia de acciones que debe devolver algún resultado de valor a un actor (52).

2.4.1 Actores del sistema

Tabla 4. Actores del sistema. Fuente: elaboración propia.

Actor	Descripción
Administrador	Es el usuario del sistema, con todos los permisos para gestionar la información referente a la estructura y operaciones del Complejo Residencial.
Recepcionista	Usuario responsable de la creación de registros de hospedaje y de incidencias del Complejo Residencial.

2.4.2 Diagrama de casos de uso del sistema

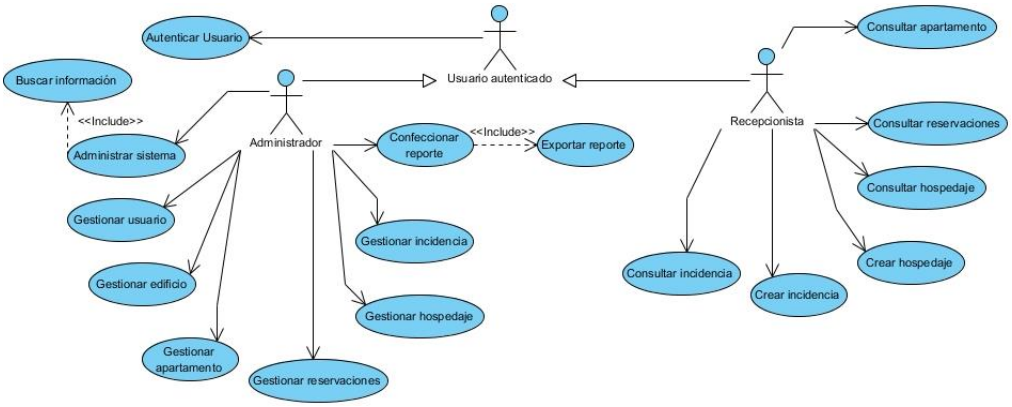


Figura 3. Diagrama de casos de uso del sistema. Fuente: elaboración propia.

En la siguiente tabla se especifica la descripción textual del caso de uso del Sistema “Gestionar Apartamento”

Tabla 5. Caso de Uso del Sistema "Gestionar Apartamento". Fuente: elaboración propia.

Caso de uso	**Gestionar Apartamento**
Actores	Usuario autenticado (Administrador): inicia.
Propósito	Este caso de uso se lleva a cabo con el objetivo de gestionar los apartamentos del Complejo Residencial UCI.
Resumen	Este caso de uso se inicia cuando el administrador selecciona la opción de listar apartamentos y culmina con la realización de una de las siguientes operaciones sobre un apartamento: Crear, Editar, Eliminar o Listar.
Precondiciones	El actor debe estar autenticado en el sistema y debe tener permisos para realizar las mencionadas operaciones. Debe seleccionarse el apartamento sobre el cual se pretende realizar una de las siguientes acciones: Modificar, Eliminar o Listar. Para realizar una de estas acciones debe existir en el sistema al menos un apartamento.
Referencias	RF5., RF5.1., RF5.2., RF5.3., RF5.4
Prioridad	Alta
Flujo Normal de Eventos	
Acción del actor	**Respuesta del Sistema**
1- El caso de uso inicia con la selección de la sección Apartamentos al nivel correspondiente.	2- Muestra una interfaz con las opciones crear, editar y mostrar En el caso de seleccionar: "Agregar apartamento" ver sección Crear apartamento. "Editar" ver sección editar apartamento. "Detalles" ver sección Mostrar apartamento.
3- El usuario selecciona la acción que desea realizar	
Prototipo de interfaz	

CAPÍTULO 2

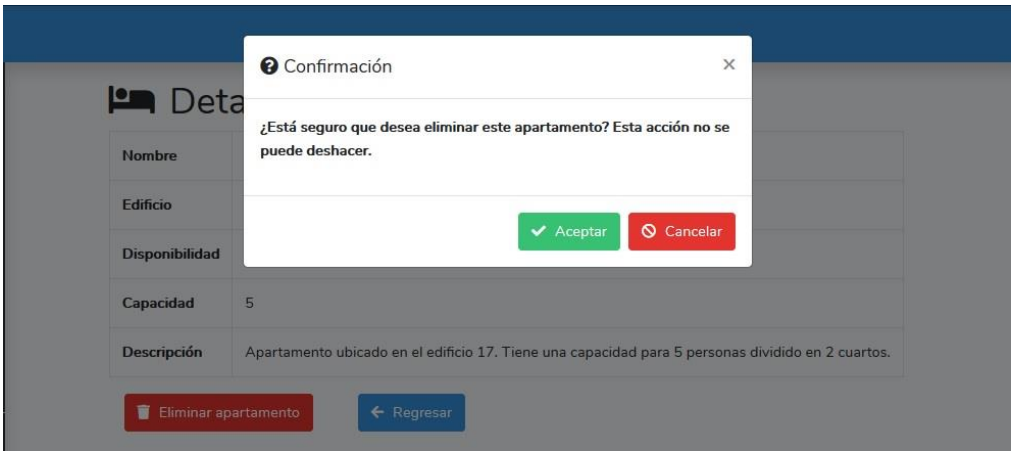
Sección agregar apartamento	
Flujos Básicos	
Acción del actor	**Respuesta del Sistema**
	1- Se muestra una interfaz para introducir los datos referentes a la creación de un apartamento
2- Se introducen los datos necesarios y selecciona el botón guardar.	3- Se verifica que no existan campos vacíos. 3.1- Verifica que los datos introducidos sean válidos.
	4- Se crea un nuevo apartamento, redirige a la vista listar apartamentos.
Prototipo de intefaz	

<table>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Flujos alternos</td></tr>
<tr><td></td><td>3- Muestra un mensaje informando que existen campos vacíos.</td></tr>
<tr><td></td><td>3.1- Muestra un mensaje informando que existe algún campo inválido.</td></tr>
<tr><td colspan="2">Sección editar apartamento</td></tr>
<tr><td colspan="2">Flujos básicos</td></tr>
<tr><td>Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td></td><td>1- Muestra una interfaz con los campos editables del apartamento y la opción actualizar.</td></tr>
<tr><td>2- El usuario determina los datos que desea editar y selecciona el botón actualizar.</td><td>3- Verifica que no existan campos vacíos.
3.1- Verifica que los datos introducidos sean válidos.</td></tr>
<tr><td></td><td>4- Edita el apartamento</td></tr>
<tr><td colspan="2">Prototipo de interfaz</td></tr>
</table>

Flujos alternos	
Acción del actor	**Respuesta del sistema**
	3- Muestra un mensaje informando que existen campos vacíos
	3.1- Muestra un mensaje informando que existe algún campo inválido.
Sección mostrar apartamento	
Flujos básicos	
Acción del actor	**Respuesta del sistema**
	1 Muestra la interfaz que contiene todos los detalles asociados a un apartamento. Además muestra la opción eliminar apartamento. En caso de seleccionar eliminar ver sección eliminar
Prototipo de interfaz	

<table>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Flujos alternos</td></tr>
<tr><td>Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td colspan="2">Sección eliminar apartamento</td></tr>
<tr><td colspan="2">Flujos básicos</td></tr>
<tr><td>Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td></td><td>1- Muestra una petición de confirmación y una advertencia de que se eliminaran de forma permanente los datos</td></tr>
<tr><td>2- El actor confirma mediante la selección del botón aceptar.</td><td>3- Elimina el apartamento y regresa a la vista de listar apartamentos</td></tr>
<tr><td colspan="2">Prototipo de interfaz</td></tr>
</table>

CAPÍTULO 2



Flujos alternos	
Acción del actor	**Respuesta del sistema**
2 El actor selecciona la opción cancelar	3 Mantiene la interfaz mostrar apartamento
Poscondiciones	Queda creado, modificado o eliminado el plan de trabajo docente metodológico.

2.4.3 Patrones de caso de uso utilizados

Los patrones de casos de uso aplicados en la construcción del diagrama de casos de uso del Sistema fueron los siguientes

CRUD Completo

Este patrón consiste en un caso de uso para administrar información, nos permite modelar las diferentes operaciones para administrar una entidad de información, tales como crear, Modificar, eliminar y listar. El mismo se manifiesta en los casos de uso: Gestionar apartamento (53) el cual posee las funciones CRUD.

Extensión Concreta o Inclusión

Una relación de inclusión es una relación desde un caso de uso base a un caso de uso de inclusión, que especifica cómo el comportamiento definido para el caso de uso de inclusión se inserta explícitamente dentro del comportamiento definido para el caso de uso base. Se utiliza para dividir partes de un flujo de trabajo de cuyos resultados, y no del método para obtenerlo, depende el caso de uso base. Se puede hacer esta partición si simplifica la comprensión del caso de uso base o si el comportamiento separado puede reutilizarse en otros casos de uso. Administrar sistema contiene a Notificar por correo (54).

Generalización/Especialización entre actores

Una relación de generalización de una clase hija de actor a otra clase padre de actor indica que el hijo hereda el rol que la clase padre puede jugar respecto a un caso de uso. Varios actores pueden jugar el mismo rol en un caso de uso particular, como se pone de manifiesto en el diagrama de casos de uso del Sistema con la generalización/especialización establecida entre los actores del sistema “usuario autenticado” y el “administrador” ya que el usuario autenticado realiza sus funciones específicas pero a ellas se le agregan otras funcionalidades como administrador (55).

2.5 Elementos fundamentales de la arquitectura

La arquitectura de software es la estructura o estructuras del sistema, que incluyen los componentes del software, las propiedades visibles externamente de esos componentes y las relaciones entre ellos. Debemos tener en cuenta que la arquitectura no es el sistema operativo es una representación que permite que un ingeniero del software analice la efectividad del diseño para cumplir con los requisitos establecidos, considere opciones arquitectónica en una etapa en que aún resulta relativamente fácil hacer cambios al diseño, reduzca los riesgos asociados con la construcción del software (56).

2.5.1 Patrones arquitectónicos y de diseño

Estilo Arquitectónico

Un estilo arquitectónico es una transformación impuesta al diseño de todo un sistema. El objetivo es establecer una arquitectura para todos los componentes del sistema. En caso de que una arquitectura

existente se valla a someter a la reingeniería, la imposición de un estilo arquitectónico desembocará en cambios fundamentales en la estructura del software, incluida una reasignación de la funcionalidad de los componentes.

Patrón arquitectónico

Un patrón arquitectónico al igual que un estilo, impone una transformación en el diseño de una arquitectura. Sin embargo, un patrón difiere de un estilo de varios elementos fundamentales: el alcance de un patrón es menor, ya que se concentra en un aspecto en lugar de hacerlo en toda la arquitectura, un patrón impone una regla sobre la arquitectura, pues describe la manera en que el software manejara algún aspecto de su funcionalidad al nivel de la infraestructura. Los patrones arquitectónicos tienden a abarcar aspectos específicos del comportamiento dentro del contexto de la arquitectura. Los patrones se usan junto con un estilo arquitectónico para determinar la forma de la estructura general de un sistema (57).

2.5.2 Patrón Modelo-Vista-Controlador

El MVC surge con el objetivo de reducir el esfuerzo de programación, necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos, a partir de estandarizar el diseño de las aplicaciones. El patrón MVC divide las partes que conforman una aplicación en el Modelo, las Vistas y los Controladores, permitiendo la implementación por separado de cada elemento, garantizando así la actualización y mantenimiento del software de forma sencilla. A partir del uso de *frameworks* basados en el patrón MVC se puede lograr una mejor organización del trabajo y mayor especialización de los desarrolladores y diseñadores (58). Laravel está basado en el patrón arquitectónico de diseño web MVC, el mismo está estructurado por tres niveles o componentes:

1. El Modelo representa toda la información con la que trabaja el sistema, es decir, su lógica de negocio.
2. La Vista transforma el modelo en una página web que permite que los usuarios interactúen con el sistema.
3. El Controlador se encarga de procesar las interacciones o peticiones realizadas por el usuario y realiza los cambios pertinentes tanto en el modelo como en la vista.

CAPÍTULO 2

Laravel toma lo mejor de la arquitectura MVC y la realiza de modo que el desarrollo de aplicaciones sea rápido y sencillo. En el controlador se encuentran las acciones, las cuales son el núcleo de la aplicación, pues contienen toda la lógica de la aplicación. Estas acciones utilizan el modelo y precisan las variables para la vista. Al realizarse una petición web en una aplicación Laravel, la *Uniform Resource Identifier* que en español significa Identificador Uniforme de Recurso (URL) define una acción y los parámetros de la petición.

La vista es la encargada de originar las páginas que son mostradas como resultado de las acciones del usuario. La vista en Laravel está conformada por varias partes o capas, esto se debe a la estructura de clases donde se evidencia la herencia, esta herencia está conformada por la vista *app.blade.php*, la cual contiene los aspectos más elementales de la vista como son: funciones Javascript, jQuery, elementos básicos de diseño como reglas *css* entre otras. En esta vista se encuentran además los elementos de la estructura de las páginas que son generales para todas las demás, estos elementos están conformados por menús y funcionalidades genéricas de la aplicación y finalmente se encuentran las vistas de la aplicación que contienen los aspectos más específicos de las vistas, estas vistas heredan de *app.blade.php* y por esta estructura es que las modificaciones resultan más sencillas.

En el Modelo se encuentran las clases, que son generadas de forma automática según la estructura de la BD, esto es posible mediante una herramienta de Mapeo Relacional de Objeto (*ORM Object Relational Mapper*) la cual implementa el propio *framework*. En Laravel, el acceso y la modificación de los datos que se almacenan en la base de datos, se realiza mediante objetos. El objetivo del controlador es crear y devolver un objeto. Para ello, a veces obtiene información de la petición, o busca un recurso en la base de datos o guarda información en la sesión del usuario. Independientemente de lo que haga, el controlador siempre devuelve un objeto que se utiliza para generar la respuesta que se envía al usuario (59).

CAPÍTULO 2

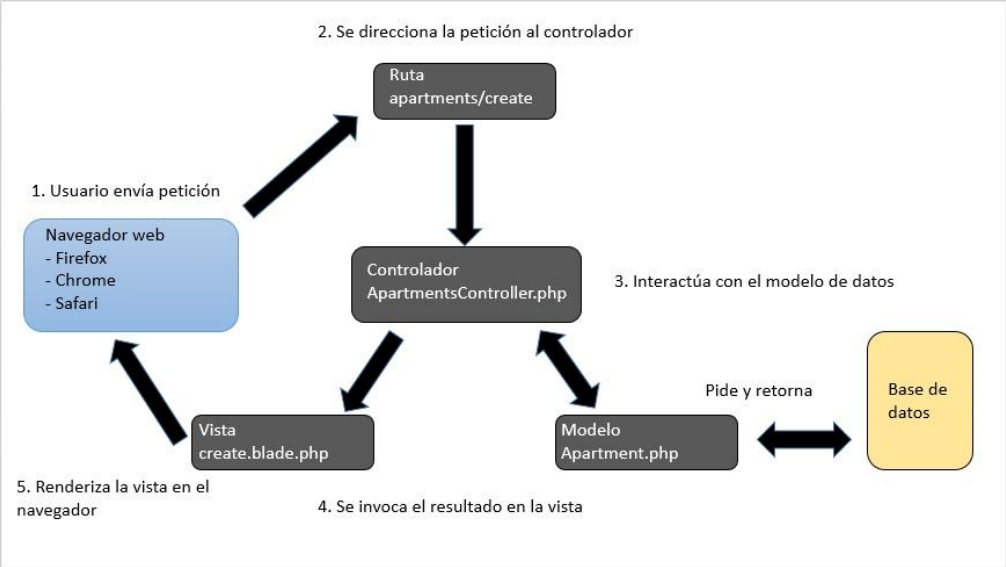


Figura 4. Ejemplo MVC en Laravel (crear apartamento). Fuente: elaboración propia.

2.6 Patrones de diseño

Los patrones de diseño son un conjunto de estrategias, o buenas prácticas, que pueden facilitar el trabajo en muchas situaciones a la hora de realizar una aplicación orientada a objetos. Los patrones GRASP describen los principios fundamentales del diseño de objetos para la asignación de responsabilidades. Por su parte los patrones GOF se clasifican en tres grupos: estructurales, creacionales y comportamiento. Los estructurales tratan la combinación de clases u objetos, su relación y la formación de estructuras de alta complejidad, mientras que los creacionales tratan la creación de instancias y los de comportamientos tratan la interacción y la cooperación entre clases (60).

CAPÍTULO 2

Patrones GRASP

1. **Experto**: utilizado con el objetivo de darle a las clases las responsabilidades necesarias siempre que cuentan con la información para cumplirlas. Un ejemplo de este patrón se evidencia en el fichero ApartmentsController.php que actúa como controlador de las acciones que se realicen en los apartamentos.
2. **Alta cohesión**: es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Se evidencia en las clases controladoras ApartmentsController.php, BuildingsController.php las cuales tienen sus respectivas funciones que facilitan en flujo de datos de una mejor manera a tener todas las funciones en una sola clase.
3. **Bajo acoplamiento**: es una medida de la fuerza con que una clase está conectada a otras, las conoce y recurre a ellas. Una clase con bajo acoplamiento no depende de muchas otras clases. Este patrón tiene como idea, tener las clases lo menos ligadas entre sí que se pueda. Esto se ve en la clase ApartmentsController.php y la clase IncidentsController.php las cuales no están estrechamente relacionadas de tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en la otra y en el resto de clases.
4. **Controlador**: consiste en asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. La arquitectura del marco de trabajo (MVC) brinda una capa específicamente para los controladores, que son el núcleo del mismo y especifica la presencia de este patrón. Todo esto está evidenciado en las clases controladoras del sistema como ApartmentsController.php que realiza funciones específicas.
5. **Creador**: Este patrón guía la asignación de responsabilidades relacionada con la creación de objetos. Se observa en el sistema propuesto, dentro de la capa controlador correspondiente a cada uno de los casos de uso la cual distribuye las responsabilidades de comunicación con las instancias de la capa de presentación, asignándole a un grupo de clases la responsabilidad de crear instancias de las clases que sirven la información necesaria para su manejo (61). Cada una de las clases controladoras del sistema que gestionan entidades como ApartmentsController tiene la responsabilidad de crear dicho objeto (en este caso un apartamento).

CAPÍTULO 2

Patrones GoF

1. **Patrón Singleton**: Este patrón garantiza que solamente se cree una instancia de las clases y provee un punto de acceso global a toda la aplicación ya que se encarga de enrutar todas las peticiones realizadas, es por ello que esta clase es utilizada por el controlador frontal de la aplicación y se evidencia su implementación en la clase Web.php, utilizándose para activar los distintos componentes del sistema que son necesarios para ejecutar la acción encomendada por el usuario.
2. **Patrón Decorator**: Este patrón se evidencia en la vista app.blade.php, padre de todas las vistas, que contiene un decorador para permitir agregar funcionalidades dinámicamente a cada página (gracias a la función @extend), esta clase contiene elementos de estilos *css* y validaciones JavaScript y jQuery que contiene los elementos de diseño que son comunes para todas las páginas y de esta forma no tener que repetirlas en cada una (62).

2.7 Modelo de Diseño

El modelo del diseño se encarga de describir la realización física de los casos de uso y se corresponde directamente con los elementos físicos del ambiente de implementación. Consiste en colaboraciones de clases, que pueden ser agregadas en paquetes y/o subsistema (63).

2.7.1 Diagrama de clases del diseño

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación y permiten modelar la vista de diseño del sistema. A continuación el diagrama de clases del diseño para los casos de uso: crear Apartamento, editar apartamento y eliminar apartamento y el Diagrama Entidad-Relación. El resto de diagramas se muestran en los anexos.

CAPÍTULO 2

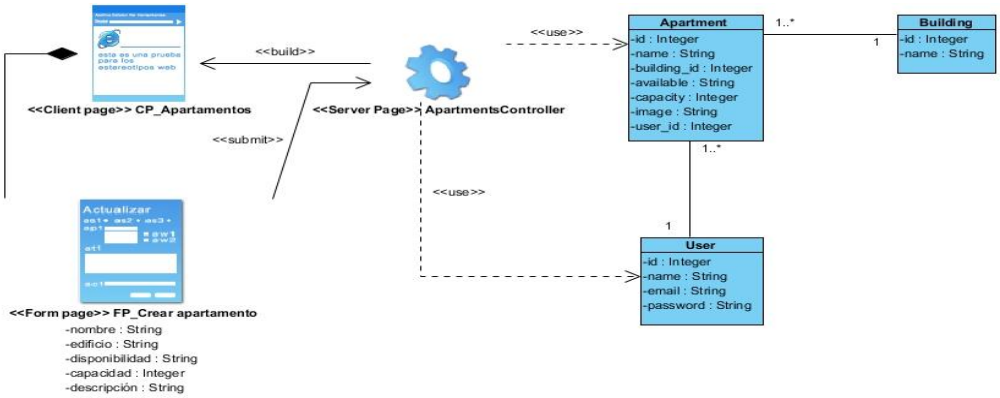


Figura 5. Diagrama de clases con estereotipos web: Crear apartamento. Fuente: elaboración propia.

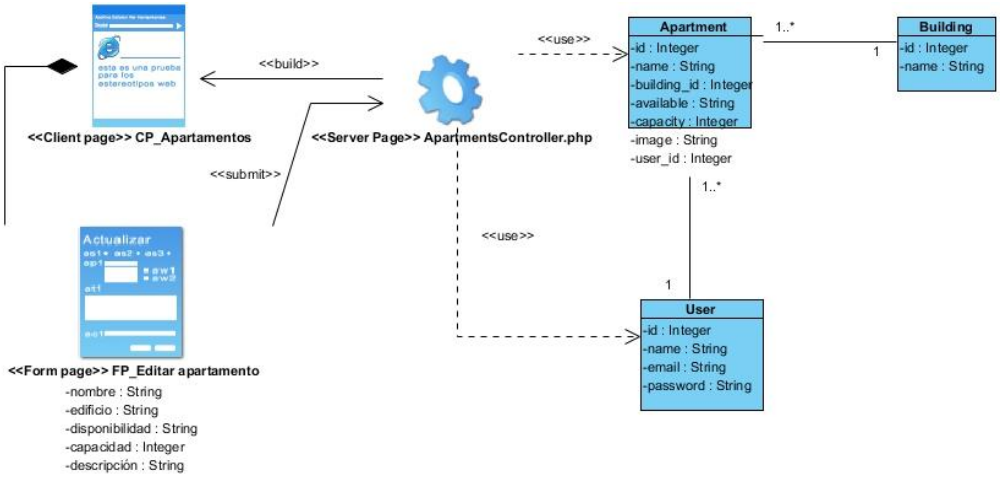


Figura 6. Diagrama de clases con estereotipos web: Editar apartamento. Fuente: elaboración propia.

CAPÍTULO 2

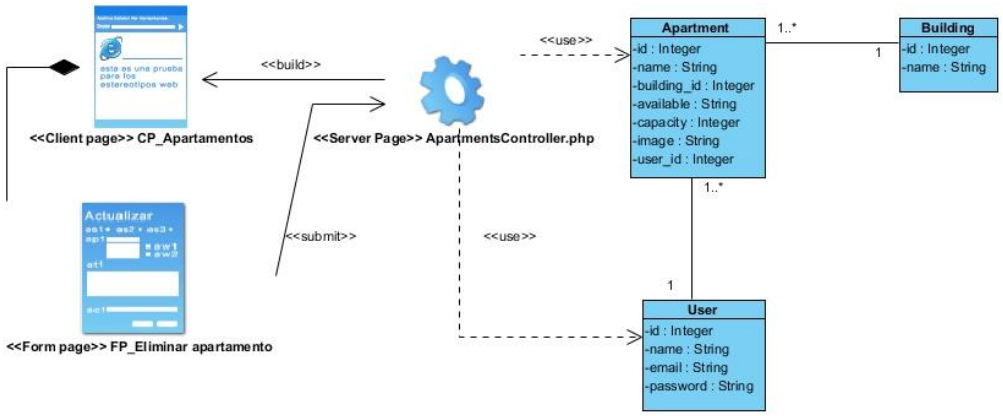


Figura 7. Diagrama de clases con estereotipos web: Eliminar apartamento. Fuente: elaboración propia.

2.8 Diagramas de secuencia

Un diagrama de secuencias muestra la interacción de un conjunto de objetos de una aplicación a través del tiempo, en el cual se indican los módulos o clases que formaran parte del programa y las llamadas que se hacen cada uno de ellos para realizar una tarea determinada. Por esta razón permite observar la perspectiva cronológica de las interacciones. Es importante recordar que el diagrama de secuencias se realiza a partir de la descripción de un caso de uso (64).

Así como se representaron los Diagramas de clases del diseño para los CU de la gestión de apartamentos, también se muestran los diagramas de secuencia de los mismos.

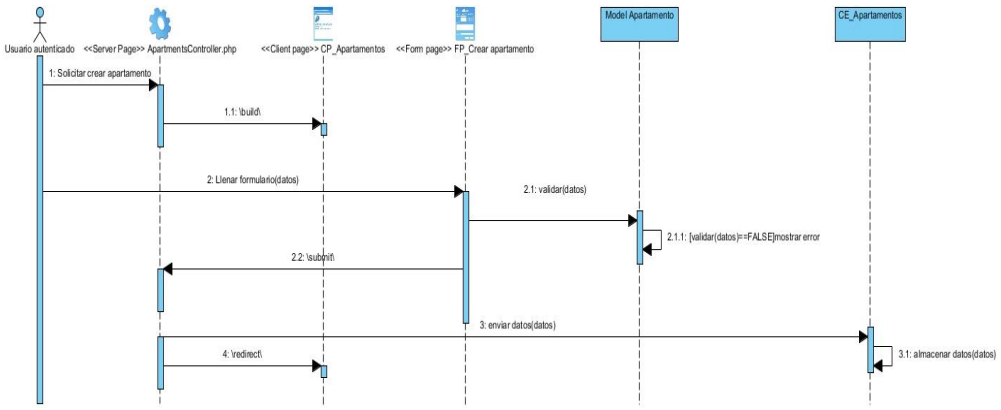


Figura 8. Diagrama de secuencia del caso de uso Crear apartamento. Fuente: elaboración propia.

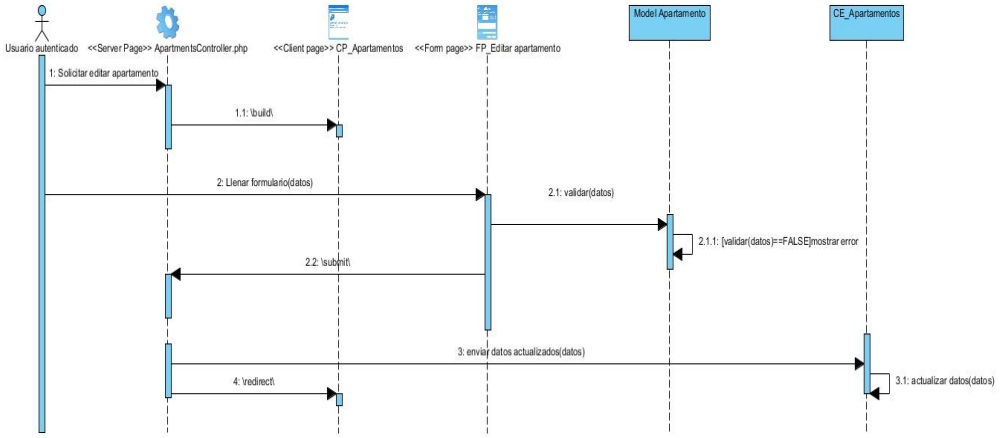


Figura 9. Diagrama de secuencia del caso de uso Editar apartamento. Fuente: elaboración propia.

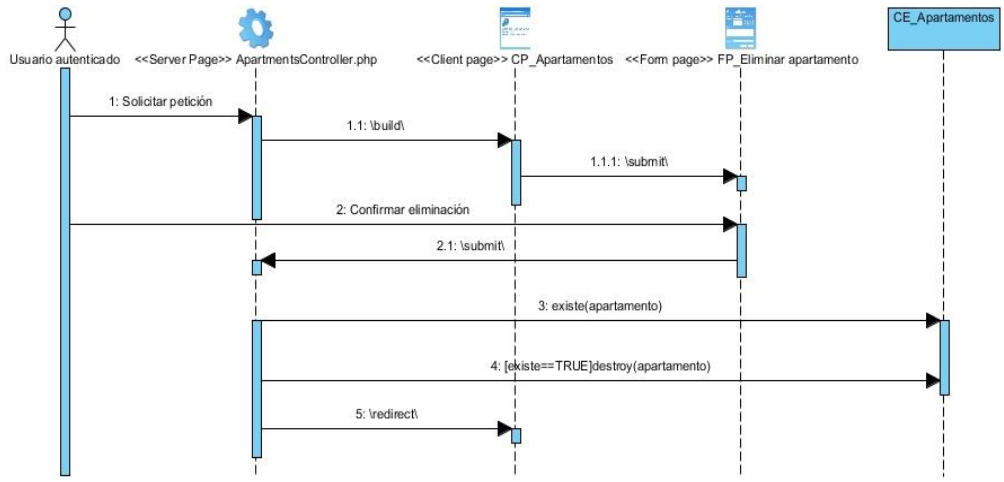


Figura 10. Diagrama de secuencia del caso de uso Eliminar apartamento. Fuente: elaboración propia.

2.9 Conclusiones parciales

En este capítulo se detallaron los elementos concernientes al proceso de análisis y diseño de la solución. Para ello se tuvo como en cuenta la metodología seleccionada (véase capítulo 1) y los distintos escenarios con que cuenta la misma. Se especificó la propuesta de solución argumentada con una arquitectura de la información de dicha propuesta. Se determinó mediante la realización del modelo conceptual, los principales elementos que caracterizan la entidad donde será desplegada la solución. Posteriormente se diseñó el diagrama de casos de uso del sistema que facilitó la obtención de los requisitos funcionales y no funcionales, así como la descripción de los casos de uso del sistema con sus flujos e interfaces. Fue desglosado los elementos del patrón arquitectónico de la solución (Modelo-Vista-Controlador), así como los patrones de diseño empleados (GRASP y GoF). Se realizaron además los diagramas de clases con estereotipos web y los diagramas de secuencia que reflejan el flujo de la información en la solución así como la respuesta dada por el sistema. Con todos estos elementos detallados se puede proceder a la implementación del sistema y las posteriores pruebas que serán argumentadas en el próximo capítulo.

Comentado [L2]: revisar esto conmigo

CAPÍTULO 3

Capítulo 3: Implementación y pruebas del sistema

Una vez concluido el modelo del diseño, se tienen los detalles suficientes para comenzar la construcción del sistema, y una vez concluido este, se procede a la verificación del cumplimiento de los requisitos funcionales mediante las pruebas de software. En el presente capítulo se describe cómo los elementos del modelo del diseño se implementan en términos de componentes y se muestra una descripción detallada de los paquetes de implementación. Además se presentan algunas imágenes del software para brindar una mejor comprensión del sistema. Posteriormente se pasa a la validación de la aplicación mediante las pruebas de software de caja negra, para comprobar la operatividad de las principales funcionalidades. También se expone el diagrama de despliegue, con una breve descripción de los nodos que lo conforman. Y finalmente se muestra el modelo de pruebas utilizado en el cual se expone la correcta implementación de los requisitos propuestos.

3.1 Modelo de implementación

El modelo de implementación es una correspondencia directa entre los modelos de diseño y de despliegue. Describe tanto los elementos del diseño como las clases. Se implementan en términos de componentes como ficheros de código fuente, ejecutables, entre otros. Representa además cómo se organizan los componentes de acuerdo con los mecanismos de estructuración disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados y cómo dependen los componentes unos de otros (65).

3.1.1 Diagrama de Componentes

Dentro del Modelo de Implementación se encuentran los diagramas de componentes. Un diagrama de componentes representa cómo un sistema de software es dividido en componentes, mostrando las dependencias que existen entre ellos. Los componentes físicos incluyen: archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables y paquetes. Los diagramas de componentes prevalecen en el campo de la arquitectura de software, pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema; estos son utilizados para modelar la vista estática y dinámica de un sistema. Muestran la

organización y las dependencias entre un conjunto de componentes (66). La siguiente figura representa el diagrama de componentes del caso de uso Gestionar Apartamento.

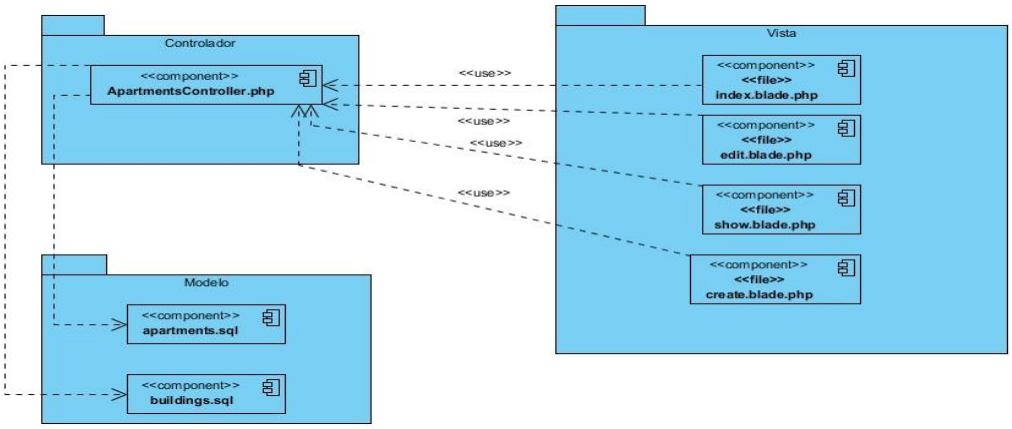


Figura 11. Diagrama de componentes del caso de uso: Gestionar apartamento. Fuente: elaboración propia.

3.2 Estándares de Codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad, es de gran importancia para la calidad del software y para obtener un buen rendimiento (67).

A continuación se exponen algunas de las buenas prácticas o estándares de codificación del lenguaje php empleadas en el desarrollo del sistema:

1. Todas las clases php deben estar delimitadas por las etiquetas de apertura y cierre de php (<?php>...?>).
2. El nombre de las clases se realiza en *UpperCamelCase*, es decir, que comienza por mayúscula.
3. Los bloques de código siempre deben estar encerrados por llaves.
4. Los nombres de los controladores deben terminar en la palabra reservada *Controller*.
5. Los nombres de las entidades deben comenzar con mayúsculas y escribirse en singular.

6. Por un consenso internacional sobre el uso de este *framework* los nombres de métodos, clases, controladores, entidades, plantillas, variables entre otros elementos están implementados en inglés.
7. Se declaran las propiedades de la clase antes que los métodos.

3.3 Código fuente

El código fuente es un conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para ejecutar dicho programa. Por tanto, en el código fuente de un programa está descrito por completo su funcionamiento. Estas instrucciones son escritas en un lenguaje de programación que consiste en un conjunto de símbolos, reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones (68).

app > Http > Controllers > ApartmentsController.php > ...

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Building;
use App\Http\Requests\StoreApartment;
use App\Apartment;

class ApartmentsController extends Controller
{
    public function __construct()
    {
        $this->middleware('auth');
    }
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index(Request $request)
    {
        $tittle = 'Apartamentos';
        $buildings = Building::orderBy('name')->pluck('name', 'id');
        $apartments = Apartment::orderBy('building_id', 'asc')->paginate();

        if ($request->has('available')) {
            $apartments = Apartment::orderBy('building_id', 'asc')->where('available', $request->get('avail
            $tittle = 'Apartamentos ' . $request->get('available') . 's';
        }
        elseif ($request->has('building_id')) {
            $apartments = Apartment::orderBy('building_id', 'asc')->where('building_id', $request->get('bui
```

Figura 12. Fragmento de código fuente: ApartmentsController.php. Fuente: elaboración propia.

3.4 Diagrama de despliegue

CAPÍTULO 3

Los diagramas de despliegue sirven para visualizar el diseño arquitectónico permitiendo ver los aspectos físicos (computadoras, sistemas, dispositivos) que implementa un sistema mediante la representación de nodos, artefactos y las relaciones entre ellos. Además se pueden organizar agrupándolos en paquetes y se pueden conectar entre sí, normalmente como asociaciones. También pueden tener atributos y operaciones y representan el empaquetamiento de elementos lógicos, bits, código y describe una topología del sistema (69).

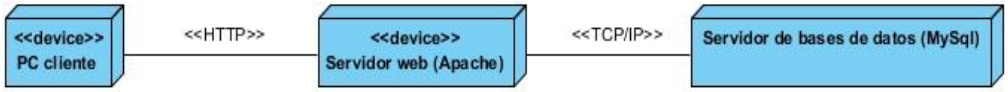


Figura 13. Diagrama de despliegue. Fuente: elaboración propia.

Descripción de los nodos

1. Nodo PC cliente: Ordenador desde el cual accedería el cliente al sistema.
2. Nodo Servidor web: Ordenador donde se ejecutará el sistema.
3. Nodo Servidor de Base de Datos: Ordenador donde se almacenará la información del sistema.

3.5 Modelo de pruebas

Las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente. La prueba de software es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación.

En el flujo de trabajo prueba se verifica el resultado de la implementación, probando cada construcción. En las pruebas se definen los casos de prueba y los procedimientos de prueba. Un caso de prueba es una forma de probar el sistema, incluyendo la entrada o resultado con la que se ha de probar y las condiciones bajo las que ha de probarse. Por su parte, un procedimiento de prueba, especifica cómo realizar uno o varios casos de prueba o partes de estos. En el ámbito de la Ingeniería de Software existen dos métodos fundamentales de pruebas: pruebas de caja blanca y pruebas de caja negra (70).

3.5.1 Pruebas de caja negra

Los casos de prueba de caja negra se enfocan en los requisitos funcionales del software. Así mismo permiten obtener conjuntos de condiciones de entrada que ejercitan completamente todos los requisitos funcionales del sistema. Además este tipo de prueba intenta encontrar errores que se clasifican en diferentes categorías como son: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos externas, errores de rendimiento y errores de inicialización y de terminación (71).

Para la aplicación de esta prueba se utiliza la técnica de partición equivalente la cual divide el campo de entrada en clases de datos de los cuales se pueden derivar casos de prueba. A continuación se describe el caso de prueba del caso de uso Gestionar Apartamento, incluyendo los resultados obtenidos en su aplicación durante la última iteración de pruebas realizadas.

Tabla 6. Descripción de las variables del caso de prueba. Fuente: elaboración propia.

No.	Nombre del campo	Clasificación	Valor nulo	Descripción
1	Nombre	Campo de texto	No	Debe introducirse caracteres numéricos.
2	Edificio	Campo de selección	No	Debe seleccionarse un elemento.
3	Disponibilidad	Campo de selección	No	Debe seleccionarse un elemento.
4	Capacidad	Campo de selección	No	Debe seleccionarse un elemento.
5	Descripción	Campo de texto	Sí	Debe introducirse caracteres alfabéticos.

Diseño de Casos de Prueba del caso de uso: Gestionar Apartamento

Descripción general

Este caso de uso se inicia cuando el usuario selecciona la opción de crear un nuevo apartamento y culmina con la realización de una de las siguientes operaciones sobre un plan: Crear, Editar, Eliminar o Mostrar.

Condiciones de ejecución

El actor debe estar autenticado en el sistema y debe tener permisos para realizar las mencionadas operaciones. Debe seleccionarse el apartamento sobre el cual se pretende realizar una de las siguientes acciones: Modificar, Eliminar o Listar. Para realizar una de estas acciones debe existir en el sistema al menos un apartamento.

Tabla 7. Secciones a probar en el caso de uso: Gestionar apartamento. Fuente: elaboración propia.

Nombre de la sección	Escenario de la sección	Descripción de la funcionalidad	Flujo central
SC 1: "Crear Apartamento".	EC 1.1: El usuario inserta y selecciona datos correctos en los campos de entrada.	El sistema almacena un nuevo Apartamento. Muestra un mensaje confirmando el éxito de la operación.	• Click en "Apartamentos". • Click en "Todos". • Click en "Agregar nuevo". • Inserta datos en los campos. • Click en el botón "Guardar."
	EC 1.2: El usuario deja campos vacíos.	El sistema muestra un mensaje indicando que se deben especificar todos los campos para crear un PTDM.	• Click en "Apartamentos". • Click en "Todos". • Click en "Agregar nuevo". • Inserta datos en los campos. • Click en el botón "Guardar".
	EC 1.3: El usuario introduce datos incorrectos.	Se muestra un mensaje que indica que existen datos incorrectos.	• Click en "Apartamentos". • Click en "Todos". • Click en "Agregar nuevo". • Inserta datos en los campos. • Click en el botón "Guardar".
SC 2: "Editar Apartamento".	EC 2.1: El usuario modifica correctamente los datos en los campos.	El sistema modifica el apartamento. Muestra un mensaje indicando el éxito de la operación.	• Click en "Apartamentos". • Click en "Todos". • Click en "Editar" en el apartamento correspondiente. • Inserta datos en los campos. • Click en el botón "Guardar".
	EC 2.2: El usuario deja campos vacíos.	El sistema indica que se deben completar todos los campos requeridos.	• Click en "Apartamentos". • Click en "Todos". • Click en "Editar" en el apartamento correspondiente. • Inserta datos en los campos.

			• Click en el botón “Guardar”.
	EC 2.3: El usuario introduce datos incorrectos.	El sistema indica que existen datos incorrectos.	• Clic en “Apartamentos”. • Clic en “Todos”. • Click en “Editar” en el apartamento correspondiente. • Inserta datos en los campos. • Clic en el botón “Guardar”.
SC 3: “Mostrar Apartamento”.	EC 3.1: Mostrar Apartamento.	El sistema muestra los datos de un apartamento dado.	• Clic en “Apartamentos”. • Clic en “Todos”. • Click en “Detalles” en el apartamento correspondiente.
SC 4: “Eliminar Apartamento”.	EC 4.1: Eliminar Apartamento.	El sistema elimina el apartamento seleccionado.	• Clic en “Apartamentos”. • Clic en “Todos”. • Click en “Detalles” en el apartamento correspondiente. • Click en “Eliminar apartamento”. • Click en “Aceptar”.

Tabla 8. Matriz de datos: crear apartamento. Fuente: elaboración propia.

Escenario	Nombre	Edificio	Disponibilidad	Capacidad	Descripción	Respuesta del sistema
El usuario inserta y selecciona datos correctos en los campos de entrada.	17101	Edificio 17	Disponible	3	Apartamento para 3 personas.	El sistema almacena un nuevo apartamento, muestra un mensaje indicando el éxito de la operación.
El usuario deja campos vacíos.	Vacío	Edificio 17	Disponible	3	Vacío	El sistema muestra un mensaje indicando que se deben especificar todos los campos requeridos para crear un apartamento
El usuario inserta y selecciona datos incorrectos en los	Ernesto	Edificio 17	Disponible	3	Vacío	El sistema muestra un mensaje indicando que existen datos incorrectos.

campos de entrada.						

Tabla 9. Matriz de datos: editar apartamento. Fuente: elaboración propia.

Escenario	Nombre	Edificio	Disponibilidad	Capacidad	Descripción	Respuesta del sistema
El usuario inserta y selecciona datos correctos en los campos de entrada.	17102	Edificio 17	Disponible	2	Apartamento para 2 personas.	El sistema actualiza los datos del apartamento, muestra un mensaje indicando el éxito de la operación.
El usuario deja campos vacíos.	Vacío	Edificio 17	Disponible	3	Vacío	El sistema muestra un mensaje indicando que se deben especificar todos los campos requeridos para editar un apartamento
El usuario inserta y selecciona datos incorrectos en los campos de entrada.	A	Edificio 17	Disponible	4	Vacío	El sistema muestra un mensaje indicando que existen datos incorrectos.

Tabla 10. Matriz de datos: mostrar apartamento. Fuente: elaboración propia.

Escenario	Nombre	Edificio	Disponibilidad	Capacidad	Descripción	Respuesta del sistema
El usuario selecciona un apartamento	N/A	N/A	N/A	N/A	N/A	El sistema muestra la información del apartamento seleccionado.

para ver sus datos.						

Tabla 11. Matriz de datos: eliminar apartamento. Fuente: elaboración propia.

Escenario	Nombre	Edificio	Disponibilidad	Capacidad	Descripción	Respuesta del sistema
El usuario selecciona un apartamento eliminarlo.	N/A	N/A	N/A	N/A	N/A	El sistema elimina toda la información de ese apartamento.

Utilizando los casos de prueba diseñados se realizaron tres iteraciones de pruebas para encontrar la mayor cantidad de errores en el funcionamiento del sistema, las cuales fueron corregidas a medida que se fue avanzando en el proceso de prueba: A continuación se muestra en la siguiente ilustración las No conformidades detectadas y su clasificación:

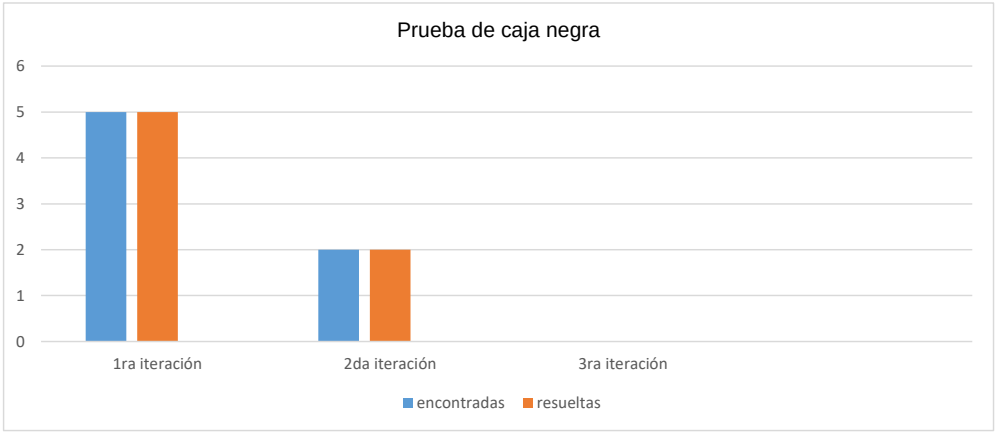


Figura 14. Ejecución de la prueba de caja negra. Fuente: elaboración propia.

3.5.2 Prueba de validación del sistema

Con el objetivo de verificar que el tiempo de creación de reportes disminuye haciendo uso del sistema desarrollado, se realizaron cuatro pruebas para comparar el tiempo de creación de reportes. Para la implementación de estas pruebas se tuvieron en cuenta dos escenarios de análisis, por lo que se describe un primer escenario (A) dividido en 2 pruebas: primero la creación de los reportes de forma manual contando solo con el apoyo de la herramienta Excel. La segunda prueba de este escenario es utilizando el sistema. En las dos primeras pruebas se crearon 5 reportes a partir de 5 reservaciones almacenadas previamente. En las pruebas que corresponden al escenario (B) 3 y 4 solamente se cambió el número de reservaciones analizadas, aumentándose la cantidad de estas a 10, manteniéndose las condiciones previamente argumentadas con el objetivo de ver el comportamiento de estos a mayor volumen de información a revisar.

Creación de reportes a partir de 5 reservaciones.

Prueba 1: Se utilizó el escenario (A) y se midió el tiempo en el que el usuario demoró en realizar los 5 reportes. Como resultado se obtuvo que el usuario demoró un tiempo de 24 minutos en realizar los reportes, debido a que el procesamiento de información se tuvo que hacer directamente por el propio usuario.

Prueba 2: Se utilizó el escenario (A) y se midió el tiempo en el que el sistema demoró en realizar los 5 reportes a partir de la interacción del usuario con el mismo. Como resultado se obtuvo que el usuario con la utilización del sistema demoró un tiempo de 2 minutos en realizar los reportes debido a que el procesamiento de información fue realizado en su totalidad por el sistema.

Creación de reportes a partir de 10 reservaciones.

Prueba 3: Se utilizó el escenario (B) y se midió el tiempo en el que el usuario demoró en realizar los 10 reportes. Como resultado se obtuvo que el usuario demoró un tiempo de 32 minutos en realizar los reportes debido a que el procesamiento de información la tuvo que hacer el mismo usuario mencionado anteriormente y el volumen a analizar de información fue mayor que en las pruebas anteriores.

Prueba 4: Se utilizó el escenario (B) y se midió el tiempo en el que el sistema demoró en realizar los 10 reportes a partir de la interacción del usuario con el mismo. Como resultado se obtuvo que el usuario con la utilización del sistema como apoyo, demoró un tiempo aproximado de 2,5 minutos (2 minutos y medio).

En el siguiente gráfico se muestra los tiempos de ejecución de las pruebas realizadas:

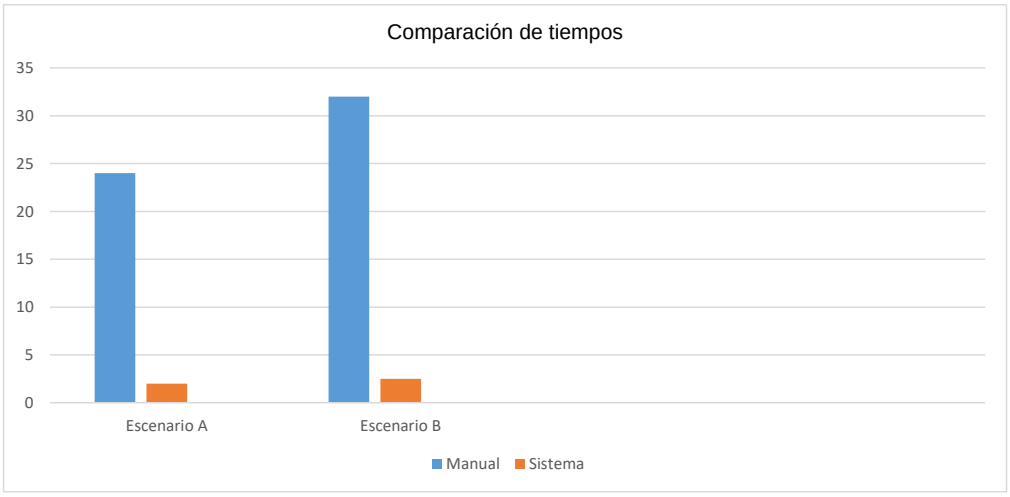


Figura 15. Prueba de validación del sistema. Fuente: elaboración propia.

Al término de la realización de esta prueba se pudieron apreciar las ventajas que brinda la utilización del sistema desarrollado mediante el procesamiento de información y creación de reportes. Se demostró que la utilización del mismo disminuyó notablemente el tiempo empleado en procesar información para generar los reportes y propició que el trabajo del usuario fuera menor al no tener que realizar este último los análisis correspondientes.

3.6 Conclusiones del capítulo

CAPÍTULO 3

Durante el desarrollo de este capítulo se definieron los estándares de codificación lo que favoreció la reutilización y mantenimiento del código fuente del sistema desarrollado. Además se describieron los principales elementos que intervienen en la solución mediante el diagrama de componente lo que permitió establecer el alcance de las funcionalidades y las relaciones entre ellas. Por su parte, el diagrama de despliegue evidenció la relación necesaria entre el hardware y el software para beneficiar el entorno físico con el cual contará el sistema. Una vez concluido el desarrollo del software se realizó la prueba de caja negra, usando la técnica de partición equivalente, esta prueba permitió hacerle una evaluación al sistema y corregir las no conformidades detectadas para contribuir con la calidad del software, también se realizó la prueba de validación, aportando la información necesaria para darle cumplimiento al objetivo general de la investigación.

CONCLUSIONES GENERALES

Conclusiones Generales

La investigación realizada arroja como principal resultado, el desarrollo de un sistema gestión de información para el Hotel Universitario de la Universidad de las Ciencias Informáticas, a partir de lo cual se arriban a las siguientes conclusiones:

1. El estudio realizado de los principales conceptos relacionados con el objeto de estudio de la investigación permitió elaborar el marco teórico de la misma.
2. El estudio del estado del arte realizado demostró que no existe un sistema capaz de realizar la gestión de los procesos realizados el Hotel Universitario de las Ciencias Informáticas con las características que este centro necesita.
3. El análisis de un conjunto de herramientas y tecnologías permitió definir el marco tecnológico en ajuste a las políticas de migración a software libre por la cual aboga el país.
4. El proceso de desarrollo de la solución propuesta fue guiado por AUP-UCI, la metodología de desarrollo seleccionada, quedando documentada cada etapa del ciclo de vida. Los artefactos generados servirán como base de futuras actualizaciones del sistema de gestión.
5. Se demuestra que el desarrollo de la aplicación permitió reducir el tiempo de procesamiento de la información y disminuir notoriamente las probabilidades de errores al aportar un valor agregado al proceso de ayuda a la toma de decisiones en el Hotel Universitario de la Universidad de las Ciencias Informáticas.

Recomendaciones

Luego de haber arribado a las conclusiones generales de esta investigación se recomienda generalizar el sistema desarrollado en todas las universidades o sedes del país que cuenten con una entidad similar al Hotel Universitario de la UCI. Se recomienda además que se vaya actualizando el mismo en el futuro para ampliar sus funcionalidades.

REFERENCIAS BIBLIOGRÁFICAS

Referencias Bibliográficas

1. GARCÍA-SANTILLÁN, Arturo, SOTO, María Cristina and GONZALEZ, Nereyda. LOS SISTEMAS INFORMÁTICOS DE GESTIÓN HOTELERA Y LOS BENEFICIOS DE SU IMPLEMENTACIÓN. *Revista de investigación de turismo y desarrollo local* [online]. 2011. Available from: www.eumed.net/rev/turydes/
2. SERFATY, Dominique. 5 beneficios de un Sistema de Gestión Hotelera Online. [online]. 2018. Available from: https://www.bebetterhotels.com/
3. Radio Habana Cuba | FitCuba 2019: La tecnología al servicio del turismo. [online]. 2019. [Accessed 12 December 2019]. Available from: http://www.radiohc.cu/noticias/ciencias/190123-fitcuba-2019-la-tecnologia-al-servicio-del-turismo
4. UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS. La UCI de un vistazo. *UCI* [online]. 2019. Available from: https://www.uci.cu/universidad/la-uci-de-un-vistazo
5. RUBIO, Maria Cruz. *EL ANÁLISIS DOCUMENTAL: INDIZACIÓN Y RESUMEN EN BASES DE DATOS ESPECIALIZADAS*.
6. ESTRADA VILLACÍS, Mónica Elizabeth. Gestión de la Información versus Gestión del Conocimiento; términos que maneja a diario el profesional de la información. [online]. 29 July 2016. Available from: www.infotecarios.com
7. Concepto de sistema de información. [online]. 2019. Available from: https://concepto.de/
8. CHEN, Caterina. Significado de Sistema de información. [online]. 21 May 2019. Available from: www.significados.com
9. Sistemas de Gestión. [online]. March 2017. Available from: https://www.consultoresdesistemasdegestion.es/
10. GIRON, Lina Marcela. ¿Qué es un sistema de gestión? [online]. 2 September 2018. Available from: https://www.siigo.com/
11. Software de gestión hotelera. [online]. 2019. Available from: https://es.eserp.com/
12. Significado de Cliente. [online]. 2019. Available from: www.significados.com
13. Definición de hospedaje. [online]. 2019. Available from: https://definicion.de/
14. Definición de huésped. [online]. 2019. Available from: https://www.definicionabc.com/

REFERENCIAS BIBLIOGRÁFICAS

15. BAYO, Marc. QuoHotel | Software de gestión hotelera. *QuoHotel* [online]. [Accessed 2 December 2019]. Available from: https://www.quohotel.com/

16. Motor de reservas para hoteles iBizi. *iBizi programa gestión hotelera* [online]. [Accessed 2 December 2019]. Available from: https://ibizi.net/motor-reservas-hoteles/

17. Sistema de reservas hoteleras | Software del hotel | Sirvoy. [online]. [Accessed 2 December 2019]. Available from: https://sirvoy.es/

18. DATYS. eHOTEL. [online]. [Accessed 2 December 2019]. Available from: http://www.datys.cu/spa/site/product/24Sistema para gestión hotelera.

19. GET sube apuesta por su sistema de gestión hotelera 100% cubana Excelencias Cuba. *Excelencias Cuba* [online]. [Accessed 2 December 2019]. Available from: https://www.excelenciascuba.com/es/noticia/get-sube-apuesta-por-su-sistema-de-gestion-hotelera-100-cubana

20. Metodología de Desarrollo de Software | Ingeniería de software | Software. *Scribd* [online]. [Accessed 2 December 2019]. Available from: https://es.scribd.com/doc/12983329/Metodologia-de-Desarrollo-de-Software

21. R.S., Tamara. *Metodología de desarrollo para la Actividad productiva de la UCI*. 3 June 2015.

22. The Unified Modeling Language. [online]. 2019. Available from: https://www.uml-diagrams.org/

23. YANEZ, Carlos. Elaborar diagramas de comportamiento en entornos de desarrollo. [online]. 28 December 2016. Available from: https://www.ceac.es/

24. Software - CASE Herramientas. [online]. 2019. Available from: https://www.tutorialspoint.com/

25. PRESSMAN, Roger. *Ingeniería de Software. Un enfoque práctico. Parte 1*. Séptima edición. McGraw-hill, 2010.

26. Visual Paradigm 8. [online]. 2010. Available from: https://www.visual-paradigm.com/cn/aboutus/newsreleases/

27. What is a Programming Language? [online]. 2019. [Accessed 2 December 2019]. Available from: https://www.computerhope.com/jargon/p/programming-language.htmComputer dictionary definition for what programming language means including related links, information, and terms.

28. ¿Qué es PHP? [online]. 2019. Available from: www.php.net

REFERENCIAS BIBLIOGRÁFICAS

29. Welcome to Python.org. *Python.org* [online]. [Accessed 2 December 2019]. Available from: https://www.python.org/about/The official home of the Python Programming Language

30. What is JavaScript? [online]. 2019. [Accessed 2 December 2019]. Available from: https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/Qu%C3%A9_es_JavaScript

31. What is jQuery? [online]. [Accessed 2 December 2019]. Available from: https://www.tutorialsteacher.com/jquery/what-is-jquery

32. HTML5. [online]. 2019. [Accessed 2 December 2019]. Available from: https://developer.mozilla.org/es/docs/HTML/HTML5

33. Qué es un framework en informática o programación | | Blog HostDime Colombia, Servidores dedicados. [online]. 5 April 2018. [Accessed 2 December 2019]. Available from: https://blog.hostdime.com.co/que-es-un-framework-informatica-programacion/

34. Laravel Documentation. [online]. 2019. Available from: https://laravel.com/

35. *Symfony The Best Practices Book version 4.2* [online]. 2019. Available from: https://sensiolabs.com/

36. Django. [online]. 2019. [Accessed 2 December 2019]. Available from: https://www.djangoproject.com/Django: The web framework for perfectionists with deadlines.

37. CONTRIBUTORS, Mark Otto, Jacob Thornton, and Bootstrap. Bootstrap. [online]. [Accessed 2 December 2019]. Available from: https://getbootstrap.com/The most popular HTML, CSS, and JS library in the world.

38. ROUSE, Margaret. ¿Qué es Servidor Web? - Definición en WhatIs.com. [online]. 2019. [Accessed 2 December 2019]. Available from: https://searchdatacenter.techtarget.com/es/definicion/Servidor-Web

39. NGINX | High Performance Load Balancer, Web Server, & Reverse Proxy. *NGINX* [online]. 2019. [Accessed 2 December 2019]. Available from: https://www.nginx.com/

40. APACHE SOFTWARE FOUNDATION. About Apache. [online]. 2019. Available from: http://httpd.apache.org/ABOUT_APACHE.

41. BORGES, Esteban. Servidor Base de Datos. Tipos y Características. [online]. 2019. Available from: https://www.infranetworking.com/

42. POSTGRESQL. About Postgresql. [online]. 2019. Available from: https://www.postgresql.org/about/

43. *MySQL Documetation* [online]. Available from: https://downloads.mysql.com/docs/refman-8.0-en.pdf

REFERENCIAS BIBLIOGRÁFICAS

44. Pgadmin. [online]. 2019. Available from: https://www.pgadmin.org/about

45. Visual Studio Code - Code Editing. Redefined. [online]. 2019. [Accessed 3 December 2019]. Available from: https://code.visualstudio.com/

46. PhpStorm: el IDE rápido e inteligente para programación en PHP de JetBrains. *JetBrains* [online]. 2019. [Accessed 3 December 2019]. Available from: https://www.jetbrains.com/phpstorm/

47. MEDINA, Oscar Carlos, MARCISZACK, Marcelo Martín and GROPPO, Mario Alberto. Trazabilidad y validación de requerimientos funcionales de sistemas informáticos mediante la transformación de modelos conceptuales. *ReCIBE. Revista electrónica de Computación, Informática, Biomédica y Electrónica*. 2016. Vol. 1, no. 1.

48. RODRÍGUEZ CASTILLA, Liuris, GONZÁLEZ HERNÁNDEZ, Delly Lien and PÉREZ GONZÁLEZ, Yudeisy. De la arquitectura de información a la experiencia de usuario: Su interrelación en el desarrollo de software de la Universidad de las Ciencias Informáticas. *E-Ciencias de la Información*. 2017. Vol. 7, no. 1, p. 155–176.

49. RAMDHANI, Muhammad Ali, MAYLAWATI, Dian Sa'adillah, AMIN, Abdusy Syakur and AULAWI, Hilmi. Requirements elicitation in software engineering. *International Journal of Engineering & Technology (UEA)*. 2018. Vol. 7, no. 2.19, p. 772–775.

50. RAMDHANI, Muhammad Ali, MAYLAWATI, Dian Sa'adillah, AMIN, Abdusy Syakur and AULAWI, Hilmi. Requirements elicitation in software engineering. Functional requirements. *International Journal of Engineering & Technology (UEA)*. 2018. Vol. 7, no. 2.19.

51. RAMDHANI, Muhammad Ali, MAYLAWATI, Dian Sa'adillah, AMIN, Abdusy Syakur and AULAWI, Hilmi. Requirements elicitation in software engineering. Non-Functional requirements. *International Journal of Engineering & Technology (UEA)*. 2018. Vol. 7, no. 2.19.

52. SCHMULLER, Joseph. *Learn UML in 24 hours*. Third Edition. Indiana, USA : Sams Publishing, [no date]. Sams Learn yourself. ISBN 0-672-32640-X.

53. SCHMULLER, Joseph. *Learn UML in 24 hours. Full CRUD pattern*. Third Edition. Indiana, USA : Sams Publishing, [no date]. Sams Learn yourself. ISBN 0-672-32640-X.

54. SCHMULLER, Joseph. *Learn UML in 24 hours. Concrete extension pattern*. Third Edition. Indiana, USA : Sams Publishing, [no date]. Sams Learn yourself. ISBN 0-672-32640-X.

REFERENCIAS BIBLIOGRÁFICAS

55. SCHMULLER, Joseph. *Learn UML in 24 hours. Generalization pattern*. Third Edition. Indiana, USA : Sams Publishing, [no date]. Sams Learn yourself. ISBN 0-672-32640-X.

56. RICHARDS, Mark. *Software architecture patterns*. O'Reilly Media, Incorporated 1005 Gravenstein Highway North, Sebastopol, CA …, 2015.

57. RICHARDS, Mark. *Software architecture patterns. Styles and Design.* O'Reilly Media, Incorporated 1005 Gravenstein Highway North, Sebastopol, CA …, 2015.

58. SHARAN, Kishori. Model-view-controller pattern. In : *Learn JavaFX 8*. Springer, 2015. p. 419–434.

59. SINHA, Sanjib. Laravel 5.7.* All Model Relations Explained: A detailed discussion of MVC Pattern, Composer, Migrations, One to One, One to Many, Many to Many, and Polymorphic Relationships. . 2018.

60. CONNOLLY, Randy. *Fundamentals of web development.* Pearson Education, 2015.

61. CONNOLLY, Randy. *Fundamentals of web development. GRASP patterns. Sumary.* Pearson Education, 2015.

62. HUSSAIN, Shahid, KEUNG, Jacky and KHAN, Arif Ali. Software design patterns classification and selection using text categorization approach. *Applied soft computing*. 2017. Vol. 58, p. 225–244.

63. ROSSI, Bruno. Entity relationship diagram. . 2014.

64. VISUAL PARADIGM TEAM. What is Sequence Diagram? [online]. [Accessed 18 March 2020]. Available from: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/

65. RAMOS, Daniel, NORIEGA, Raúl, LAÍNEZ, José Rubén and DURANGO, Alicia. *Curso de Ingeniería de Software: 2ª Edición*. IT Campus Academy, 2017.

66. PALOMO, Sebastián Rubén Gómez and GIL, Eduardo Moraleda. *Aproximación a la ingeniería del software*. Editorial Centro de Estudios Ramon Areces SA, 2020.

67. SIERRA, F, ACOSTA, J, ARIZA, J and SALAS, M. Estudio y análisis de los framework en php basados en el modelo vista controlador para el desarrollo de software orientado a la web. *Investigación y desarrollo en TIC*. 2013. Vol. 4, no. 2, p. 14–26.

68. CASALE, Juan Carlos. *Introducción a la programación: Aprenda a programar sin conocimientos previos*. RedUsers, 2016.

69. PALOMO, Sebastián Rubén Gómez and GIL, Eduardo Moraleda. *Aproximación a la ingeniería del software*. Editorial Centro de Estudios Ramon Areces SA, 2020.

REFERENCIAS BIBLIOGRÁFICAS

70. RAMOS, Daniel, NORIEGA, Raúl, LAÍNEZ, José Rubén and DURANGO, Alicia. *Curso de Ingeniería de Software: 2ª Edición*. IT Campus Academy, 2017.

71. RAMOS, Daniel, NORIEGA, Raúl, LAÍNEZ, José Rubén and DURANGO, Alicia. *Curso de Ingeniería de Software: 2ª Edición*. IT Campus Academy, 2017.