

**UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS**

**FACULTAD DE TECNOLOGÍAS EDUCATIVAS**

**Título**: Sistema de Gestión Hotelera

**Trabajo de Curso de Ingeniería de Software II**

Autores:

David Avila de la Rosa

Frank Ernesto Cortiñas Peña

Mabel María Castillo Moreno

Maydelín del Carmen Ibarra Núñez

Oniel José Aguilar Milián

Tutor:

Faure Arturo Fiallo Zequeira

**La Habana, septiembre 2025**

**“Año 66 de la Revolución”**

# Resumen

En los últimos años, la industria hotelera ha adoptado tecnologías avanzadas que transforman su funcionamiento, optimizando procesos y mejorando la experiencia del cliente. Los sistemas de gestión hotelera (SGH) han evolucionado hacia soluciones basadas en la nube, análisis de datos y metodologías ágiles, adaptándose a un entorno competitivo y cambiante. Este trabajo propone el desarrollo de un SGH utilizando un modelo prescriptivo y la metodología RUP, con Django para el backend y React para el frontend. Se analiza el estado del arte, identificando herramientas similares y los conceptos clave que guían esta evolución hacia sistemas ágiles, escalables y centrados en el usuario. El proyecto busca alinear la digitalización en la industria hotelera con un enfoque estructurado y flexible, ofreciendo una solución moderna y eficiente para la gestión operativa. Se utilizarán herramientas como Visual Studio Code, para el desarrollo del frontend y el backend y PgAdmin para la gestión eficiente de bases de datos. Este conjunto de herramientas y tecnologías no solo optimiza el desarrollo, sino que también garantiza la alta calidad, seguridad y escalabilidad del sistema a lo largo del tiempo. Además, la integración fluida entre backend, frontend y base de datos permitirá una implementación eficiente y sin interrupciones, asegurando que el sistema de gestión hotelera sea robusto, flexible y capaz de adaptarse a las necesidades del negocio conforme este crezca. La elección de tecnologías de código abierto también ofrece una ventaja en términos de costos y soporte continuo.

**Palabras clave:**

1. Gestión
2. Desarrollo
3. Tecnologías
4. Escalabilidad
5. Integración

**Abstract**

In recent years, the hospitality industry has adopted advanced technologies that are transforming its operations, optimizing processes, and enhancing the customer experience. Hotel management systems (HMS) have evolved into cloud-based solutions, data analysis tools, and agile methodologies, adapting to a competitive and ever-changing environment. This work proposes the development of an HMS using a prescriptive model and the RUP methodology, with Django for the backend and React for the frontend. The state of the art is analyzed, identifying similar tools and key concepts that guide this evolution toward agile, scalable, and user-centered systems. The project seeks to align digitalization in the hospitality industry with a structured and flexible approach, offering a modern and efficient solution for operational management. Tools such as Visual Studio Code will be used for frontend and backend development, and PgAdmin for efficient database management. This set of tools and technologies not only optimizes development but also ensures high quality, security, and scalability of the system over time. Furthermore, seamless integration between backend, frontend, and database will enable efficient and uninterrupted implementation, ensuring the hotel management system is robust, flexible, and capable of adapting to business needs as it grows. The choice of open-source technologies also provides an advantage in terms of costs and continuous support.

**Keywords:**

1. Management

2.Development

3.Technologies

4. Scalability

5.Integration

# Índice

[Resumen I](#_Toc170819654)

[Índice II](#_Toc170819655)

[Introducción 1](#_Toc170819656)

[Capítulo I. Análisis del sistema a desarrollar 4](#_Toc170819657)

[Introducción 4](#_Toc170819658)

[1.1 Descripción del sistema a desarrollar 4](#_Toc170819659)

[Conclusiones parciales 8](#_Toc170819660)

[Capítulo II. Vista de Gestión del Modelo 9](#_Toc170819661)

[Introducción 9](#_Toc170819662)

[2.1 Vista de Gestión del Modelo 9](#_Toc170819663)

[Conclusiones parciales 10](#_Toc170819664)

[Capítulo III. Vista de Presentación 10](#_Toc170819665)

[Introducción 10](#_Toc170819666)

[3.1 Mapa de navegación 10](#_Toc170819667)

[3.2 Definición de contenidos pantalla de bienvenida 11](#_Toc170819668)

[3.2.1 Descripción de los contenidos a mostrar en pantalla de bienvenida 11](#_Toc170819669)

[3.3 Definición de contenidos pantallas genéricas por roles 12](#_Toc170819670)

[3.3.1 Descripción de los contenidos a mostrar en pantalla del rol <Usuario> 12](#_Toc170819671)

[3.3.2 Descripción de los contenidos a mostrar en pantalla del rol <Director General> 12](#_Toc170819672)

[3.3.2 Descripción de los contenidos a mostrar en pantalla del rol <Director Tecnico> 13](#_Toc170819673)

[Conclusiones parciales 14](#_Toc170819674)

[Capítulo IV. Vista Lógica 15](#_Toc170819675)

[Introducción 15](#_Toc170819676)

[4.1 Estilo arquitectónico 15](#_Toc170819677)

[4.2 Diagramas de clases 15](#_Toc170819678)

[4.3 Patrones de diseño 23](#_Toc170819679)

[4.3.1 Patrones GRASP 24](#_Toc170819680)

[4.3.2 Patrones GOF 25](#_Toc170819681)

[4.4 Diagramas de Interacción 25](#_Toc170819682)

[4.4.1 Diagrama de secuencia del Caso de uso <Nombre del Caso de Uso>. <Nombre del Escenario> 26](#_Toc170819683)

[4.5 Modelo de Datos 26](#_Toc170819684)

[Conclusiones parciales 26](#_Toc170819685)

[Capítulo V. Vista de Implementación 28](#_Toc170819686)

[Introducción 28](#_Toc170819687)

[5.1 Diagrama de Componentes 28](#_Toc170819688)

[Conclusiones parciales 29](#_Toc170819689)

[Capítulo VI. Vista de Despliegue 29](#_Toc170819690)

[Introducción 29](#_Toc170819691)

[4.1 Diagrama de Despliegue 30](#_Toc170819692)

[4.1.1 Nombre de dispositivo: descripción de la capacidad que el dispositivo provee al sistema 30](#_Toc170819693)

[4.1.2 Nombre del procesador: descripción de la funcionalidad y capacidad del nodo 30](#_Toc170819694)

[4.1.3 <<Nombre tipo de conexión>>: Características físicas de la conexión 30](#_Toc170819695)

[Conclusiones parciales 30](#_Toc170819696)

[Capítulo VII. Prueba 31](#_Toc170819697)

[Introducción 31](#_Toc170819698)

[7.1 Diseño de casos de prueba con caja negra 31](#_Toc170819699)

[7.1.1 Descripción General 31](#_Toc170819700)

[7.1.2 Condiciones de Ejecución 31](#_Toc170819701)

[7.1.3 Secciones a probar en el Caso de Uso 32](#_Toc170819702)

[7.2 Métodos a probar de las secciones 33](#_Toc170819703)

[7.3 Diseño de casos de prueba con caja blanca 33](#_Toc170819704)

[7.4 Registro de defectos y dificultades detectados 35](#_Toc170819705)

[Conclusiones parciales 35](#_Toc170819706)

[Referencias Bibliográficas 38](#_Toc170819707)

[Bibliografía Consultada 40](#_Toc170819708)

[Anexos 41](#_Toc170819709)

**Introducción**

En el presente documento se plantea el desarrollo de un sistema de gestión hotelera, el cual surge ante la necesidad de optimizar los procesos internos y mejorar la experiencia de los huéspedes en el entorno altamente competitivo del sector hotelero.

**Situación Problemática:**

Actualmente, los hoteles dependen de sistemas fragmentados (ej.: PMS para reservas, CRM para clientes, BI para informes), lo que genera redundancia de datos y altos costos de integración. Sistemas como Hotelogix y RoomRaccoon ofrecen soluciones en la nube pero carecen de personalización para hoteles medianos. Nuestra propuesta integra todas las funcionalidades en una plataforma unificada, priorizando escalabilidad y adaptabilidad mediante código abierto, a diferencia de soluciones propietarias como Oracle Opera.

**Tema de Investigación:**

Desarrollo de un sistema de gestión hotelera que centralice y automatice las funciones críticas (registro, reservas, disponibilidad, pagos, informes y gestión de usuarios) utilizando metodologías tradicionales y herramientas tecnológicas modernas.

**Objetivo General:**

Diseñar e implementar un sistema de gestión hotelera que integre funciones operativas y administrativas, optimizando procesos y garantizando una experiencia de usuario de calidad.

**Objetivos Específicos:**

- Centralizar el registro de clientes y la gestión de reservas en una plataforma única.

- Automatizar tareas operativas (verificación de disponibilidad, procesamiento de pagos, etc.) para minimizar errores.

- Proveer herramientas de análisis y generación de informes que apoyen la toma de decisiones.

- Facilitar la gestión de usuarios y roles (recepcionistas, gerentes y personal de apoyo).

**Acciones a Investigar:**

- Análisis de los flujos de trabajo actuales en hoteles de mediano tamaño.

- Estudio comparativo de sistemas existentes en el mercado (PMS, CRM, BI).

- Diseño y modelado de la arquitectura del sistema, definiendo módulos esenciales.

- Pruebas piloto y recopilación de retroalimentación para la mejora continua del sistema.

**Justificación de la Investigación:**

La implementación de este sistema permitirá modernizar la operación hotelera, reduciendo costos operativos y mejorando la eficiencia en la atención al cliente. Además, se favorecerá la integración de herramientas tecnológicas que faciliten la toma de decisiones y la adaptabilidad ante cambios en el mercado.

El proyecto se sustentó en:

Revisión bibliográfica: Análisis de artículos científicos, documentos técnicos y casos de éxito de sistemas como Oracle Hospitality y Cloudbeds.

Encuestas y entrevistas: Realizadas a gerentes y recepcionistas de hoteles medianos para identificar necesidades operativas.

Benchmarking: Evaluación comparativa de 10 sistemas PMS (Property Management Systems) para determinar funcionalidades críticas.

Prototipado iterativo: Desarrollo de MVP (Minimum Viable Product) validado con usuarios reales mediante pruebas piloto.

**Introducción a los Capítulos del Documento**

**Capítulo I: Análisis del Sistema a Desarrollar**

Objetivo: Definir los requisitos y tecnologías del sistema.

Actividades:

Identificar necesidades operativas del sector hotelero.

Establecer requisitos funcionales (ej.: gestión de reservas) y no funcionales (ej.: seguridad).

Seleccionar tecnologías (Django, Angular, PostgreSQL) y metodologías (RUP, SCRUM).

Modelar casos de uso para visualizar interacciones entre usuarios y el sistema.

**Capítulo II: Vista de Gestión del Modelo**

Objetivo: Diseñar la estructura modular del sistema.

Actividades:

Segmentar el sistema en subsistemas clave: reservas, habitaciones, pagos e informes.

Especificar responsabilidades y flujos de datos entre módulos.

Garantizar escalabilidad y mantenibilidad mediante una arquitectura modular.

**Capítulo III: Vista de Presentación**

Objetivo: Diseñar interfaces de usuario adaptadas a roles.

Actividades:

Crear wireframes y mapas de navegación para cada rol (ej.: recepcionista, director general).

Definir contenidos y funcionalidades en pantallas clave (ej.: bienvenida, dashboard).

Priorizar usabilidad y accesibilidad en dispositivos múltiples.

**Capítulo IV: Vista Lógica**

Objetivo: Estructurar la lógica interna del sistema.

Actividades:

Definir la arquitectura MVC (backend) y basada en componentes (frontend).

Elaborar diagramas de clases y secuencia para modelar interacciones.

Aplicar patrones de diseño (GRASP, GOF) para optimizar cohesión y reducir acoplamiento.

**Capítulo V: Vista de Implementación**

Objetivo: Traducir el diseño en componentes funcionales.

Actividades:

Desarrollar módulos independientes (ej.: gestión de reservas).

Integrar backend, frontend y base de datos.

Garantizar coherencia entre componentes mediante pruebas unitarias.

**Capítulo VI: Vista de Despliegue**

Objetivo: Planificar la infraestructura de implementación.

Actividades:

Diseñar diagramas de despliegue para entornos cloud (AWS, Render).

Especificar requisitos de hardware, redes y seguridad.

Asegurar alta disponibilidad y escalabilidad horizontal.

**Capítulo VII: Pruebas**

Objetivo: Validar la calidad y funcionalidad del sistema.

Actividades:

Ejecutar pruebas de caja negra (funcionalidades visibles) y blanca (código interno).

Documentar defectos y optimizar el sistema basado en retroalimentación.

Garantizar cumplimiento de requisitos iniciales y estándares de rendimiento.

**Capítulo I. Análisis del Sistema a Desarrollar**

**Introducción**

El análisis detallado presentado en este capítulo establece una base sólida para el desarrollo del sistema de gestión hotelera. La definición clara de los requisitos funcionales (como registro de clientes, gestión de reservas y procesamiento de pagos) y no funcionales (usabilidad, seguridad y escalabilidad) garantiza que el sistema cubra las necesidades operativas y estratégicas del sector hotelero. La incorporación de tecnologías modernas como Django, React y PostgreSQL, junto con metodologías ágiles como RUP y SCRUM, asegura un desarrollo eficiente, mantenible y adaptable a futuras demandas. Los diagramas de casos de uso, como el de reserva, refuerzan la comprensión de las interacciones entre usuarios y el sistema, alineando la solución con los flujos de trabajo reales de los hoteles. Este capítulo no solo delimita el alcance técnico, sino que también valida la viabilidad del proyecto mediante herramientas y enfoques probados en la industria.

**1.1 Descripción del Sistema a Desarrollar**

El sistema propuesto es una solución integral para la gestión hotelera, la cual abarca desde el registro de clientes hasta la administración de reservas, disponibilidad de habitaciones, procesamiento de pagos y generación de informes. Entre sus principales características destacan:

- Centralización de la información: Un único repositorio de datos que permita la integración de todas las operaciones del hotel.

- Automatización de procesos: Mecanismos para la verificación en tiempo real de la disponibilidad de habitaciones, asignación automática y cálculo de tarifas.

- Interfaz intuitiva: Diseño responsivo y amigable, adaptable a distintos dispositivos y roles (huésped, recepcionista, gerente).

- Seguridad y confiabilidad: Implementación de estándares de seguridad, copias de respaldo y mecanismos de recuperación de datos.

**1.2 Requisitos del Sistema**

El sistema se definirá mediante dos grandes categorías de requisitos: funcionales y no funcionales.

**1.2.1 Requisitos Funcionales:**

- RF01 – Registro de Clientes: Permitir el registro y actualización de la información personal de nuevos clientes.

- RF02 – Gestión de Reservas: Facilitar la creación, modificación y cancelación de reservas, incluyendo la asignación de habitaciones.

- RF03 – Verificación de Disponibilidad: Consultar y mostrar en tiempo real la disponibilidad de habitaciones según las fechas solicitadas.

- RF04 – Procesamiento de Pagos: Integrar un módulo para el procesamiento seguro de pagos y emisión de recibos.

- RF05 – Generación de Informes: Proveer herramientas para la elaboración de reportes sobre ocupación, reservas y rendimiento financiero.

- RF06 – Gestión de Usuarios: Administrar perfiles y permisos de acceso para los diferentes actores del sistema (recepcionistas, gerentes, etc.).

**1.2.2 Requisitos No Funcionales:**

- Usabilidad: La interfaz debe ser clara, intuitiva y accesible para usuarios con distintos niveles de experiencia.

- Eficiencia: El sistema debe procesar transacciones en menos de 2 segundos durante picos de demanda y soportar múltiples usuarios concurrentes.

- Seguridad: Implementación de autenticación robusta (incluyendo autenticación de dos factores) y encriptación de datos sensibles.

- Compatibilidad: Debe funcionar correctamente en los principales navegadores (Chrome, Firefox, Safari) y en diferentes dispositivos.

- Escalabilidad y Mantenibilidad: El diseño debe permitir futuras ampliaciones y actualizaciones con facilidad, apoyado en una documentación clara y modularidad del código.

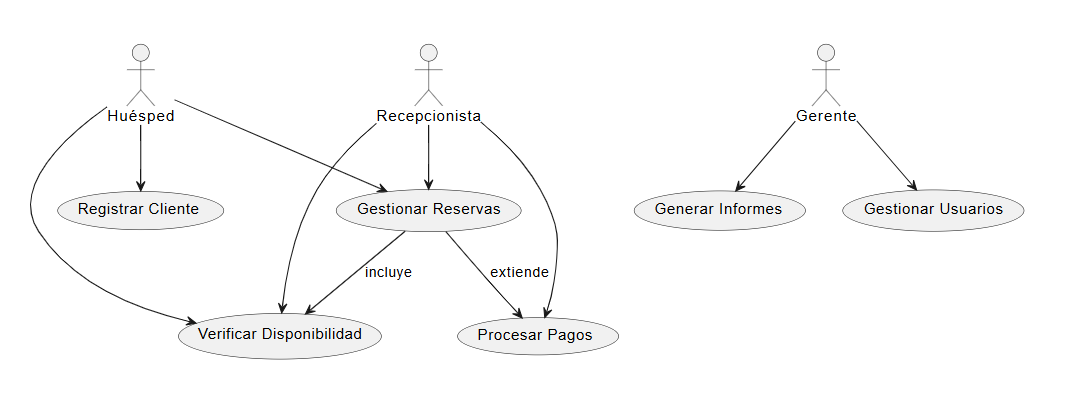
**1.3 Tecnologias Utilizadas**

El sistema se desarrollará utilizando un conjunto de tecnologías modernas y escalables que garantizan eficiencia, seguridad y mantenibilidad:

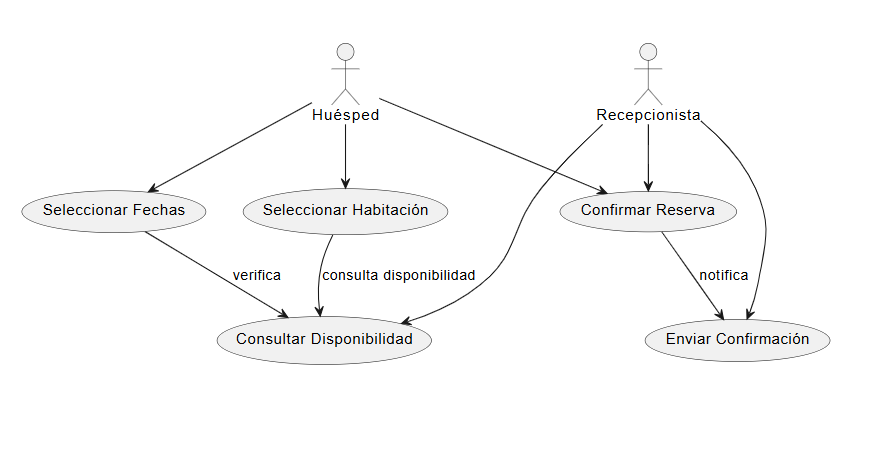
* Backend: Django (Python) por su robustez, escalabilidad y soporte para ORM, facilitando la integración con bases de datos.
* Frontend: React (Javascript) para una interfaz dinámica y responsiva, aprovechando su arquitectura modular y enlace de datos bidireccional.
* Base de datos: PostgreSQL, gestionado con PgAdmin, por su rendimiento en transacciones concurrentes y soporte ACID.
* Herramientas de desarrollo: Visual Studio Code para edición de código, Git para control de versiones y Docker para containerización.
* Metodología: RUP (Rational Unified Process) para un enfoque iterativo e incremental, combinado con SCRUM para gestión ágil de tareas.

**1.4 Diagrama de Casos de Uso del Sistema**

**1.4.1 Diagrama de Casos de Uso General del Sistema**

****

**1.4.2 Diagrama detallado de un Caso de Reserva**

****

## Capítulo II. Vista de Gestión del Modelo

## Introducción La vista de gestión del modelo evidencia la robustez del diseño modular del sistema, estructurado en subsistemas especializados (reservas, habitaciones, pagos e informes) que interactúan de manera cohesiva. Esta segmentación permite una gestión integral de las operaciones críticas del hotel, optimizando procesos como la asignación automática de habitaciones y la generación de informes analíticos. La arquitectura modular no solo facilita la escalabilidad, al permitir actualizaciones independientes en cada paquete, sino que también mejora la mantenibilidad al reducir la complejidad del código. Además, la integración fluida entre subsistemas asegura una sincronización precisa de datos, minimizando errores y redundancias. En conjunto, este enfoque organizacional garantiza que el sistema no solo cumpla con las necesidades actuales, sino que también esté preparado para adaptarse a futuras innovaciones tecnológicas o cambios en las

## 2.1 Vista de Gestión del Modelo



### 2.2 Descripción de los paquetes o subsistemas

En este apartado se profundiza en la estructura modular del sistema de gestión hotelera, presentando una descripción detallada de cada uno de los paquetes o subsistemas que, al integrarse, permiten cubrir la totalidad de las operaciones y procesos críticos del negocio. Cada paquete ha sido diseñado para agrupar funcionalidades específicas que aseguran no solo la operatividad diaria del hotel, sino también la eficiencia y escalabilidad del sistema en el tiempo.

**Paquete de Gestión de Reservas:**  
Este subsistema se encarga de administrar todo el ciclo de vida de una reserva. Desde la creación inicial, que incluye la validación de fechas, selección de habitación y verificación de disponibilidad, hasta la modificación y cancelación de reservas, el paquete garantiza que el proceso se realice de manera fluida y sin errores. Además, incorpora mecanismos para la asignación óptima de habitaciones basados en criterios de preferencia y disponibilidad, lo que permite maximizar la ocupación y mejorar la experiencia del cliente.

**Paquete de Gestión de Habitaciones:**  
El control de las habitaciones es fundamental para el funcionamiento del hotel. Este subsistema se encarga de mantener actualizada la disponibilidad de las mismas, gestionar su asignación y llevar un registro del mantenimiento preventivo y correctivo. Mediante la integración de herramientas de seguimiento y notificaciones, se optimiza la utilización de las habitaciones y se asegura que siempre se encuentren en condiciones adecuadas para recibir a los huéspedes.

**Paquete de Gestión de Pagos:**  
La solidez financiera del sistema se sustenta en el adecuado procesamiento de las transacciones económicas. Este paquete se dedica a la gestión integral de los pagos, abarcando desde la validación de los métodos de pago hasta la emisión de recibos y la conciliación de ingresos. Con sistemas de seguridad robustos y procesos automatizados, se minimizan los errores y se garantiza la integridad y confidencialidad de la información financiera, proporcionando además reportes detallados para la toma de decisiones estratégicas.

**Paquete de Reportes e Informes:**  
La capacidad de análisis es crucial para optimizar la operación y la estrategia del hotel. Este subsistema permite la generación de informes analíticos sobre diversos indicadores clave, tales como la ocupación, ingresos, tendencias de reserva y desempeño de los servicios. Los reportes, presentados en formatos visuales e intuitivos, facilitan la identificación de oportunidades de mejora y permiten a los directivos tomar decisiones informadas y oportunas, ajustando las estrategias de negocio según las necesidades del mercado.

### Conclusiones Parciales

La descripción detallada de los paquetes o subsistemas expuesta en este capítulo evidencia la estructura modular y la integración de funcionalidades específicas orientadas a cubrir todos los aspectos críticos del sistema de gestión hotelera. Cada subsistema, desde la gestión de reservas y habitaciones hasta el procesamiento de pagos y la generación de informes, se configura como un bloque esencial que contribuye a la operatividad integral y al éxito del sistema. La segmentación en paquetes permite además una mayor escalabilidad, facilitando futuras mejoras y adaptaciones sin comprometer la estabilidad del sistema global. Esta organización modular no solo optimiza los procesos internos, sino que también mejora la experiencia del usuario, garantizando eficiencia, seguridad y un alto nivel de servicio.

# Capítulo III. Vista de Presentación

## 3.1 Introduccion

La Vista de Presentación constituye un elemento crítico en el desarrollo de nuestro Sistema de Gestión Hotelera, ya que define la forma en que los usuarios interactuarán con el software y perciben su funcionalidad. Este capítulo detalla el diseño de las interfaces de usuario, siguiendo principios de usabilidad y experiencia de usuario (UX) para garantizar que el sistema sea intuitivo, eficiente y accesible para todos los perfiles de usuarios.

El diseño de las interfaces se ha desarrollado siguiendo las metodologías de Diseño Centrado en el Usuario (DCU), priorizando la simplicidad, consistencia y eficiencia en las interacciones. Cada pantalla ha sido conceptualizada considerando los flujos de trabajo reales del entorno hotelero, asegurando que las tareas frecuentes requieran el mínimo de pasos posibles.

La arquitectura visual del sistema implementa un enfoque responsivo basado en componentes React, garantizando que las interfaces se adapten correctamente a diferentes dispositivos (ordenadores de escritorio, tablets y smartphones), lo que permite a los usuarios acceder al sistema desde cualquier ubicación y dispositivo.

# 3.2 Definición de contenidos pantalla de bienvenida

# 3.2.1 Descripción de los contenidos a mostrar en pantalla de bienvenida

|  |  |  |  |
| --- | --- | --- | --- |
| Componente | Ubicación | Descripción | Funcionalidad |
| Logo del hotel | Esquina superior izquierda | Imagen corporativa del hotel | Enlace a inicio |
| Menú de navegación | Cabecera horizontal | Opciones: Inicio, Nuestras Habitaciones, Servicios, Contacto | Navegación principal del sitio |
| Selector de idioma | Cabecera, lado derecho | Iconos de banderas para español e inglés | Cambio de idioma del sitio |
| Botón de acceso | Cabecera, extremo derecho | Botón "Iniciar sesión/Registrarse" | Acceso al portal del usuario |
| Carrusel de imágenes | Sección principal superior | Fotografías destacadas del hotel con transiciones | Presentación visual del hotel |
| Mensaje de bienvenida | Sobre el carrusel | Texto personalizado según temporada | Información contextual |
| Buscador de disponibilidad | Centro de la página | Formulario con campos para fechas, huéspedes y tipo de habitación | Verificación de disponibilidad |
| Resumen de servicios | Sección media | Iconos y descripciones breves de servicios destacados | Información rápida de servicios |
| Galería de habitaciones | Sección inferior | Miniaturas de tipos de habitaciones disponibles | Vista previa de opciones |
| Ofertas especiales | Lateral derecho | Banners con promociones actuales | Resaltar ofertas vigentes |
| Información de contacto | Pie de página | Teléfono, correo electrónico, dirección | Facilitar comunicación |
| Enlaces legales | Pie de página | Políticas de privacidad, términos y condiciones | Cumplimiento normativo |
| Enlaces a redes sociales | Pie de página | Iconos de plataformas sociales | Presencia en redes |

# 3.3 Definición de contenidos pantallas genéricas por roles

# 3.3.1 Descripción de los contenidos a mostrar en pantalla del rol <Huésped>

|  |  |  |  |
| --- | --- | --- | --- |
| Componente | Ubicación | Descripción | Funcionalidad |
| Barra superior | Parte superior | Contiene logo, nombre del huésped, foto de perfil, notificaciones y opciones de perfil | Navegación principal y acceso rápido |
| Panel lateral | Izquierda | Menú con opciones: Dashboard, Mis reservas, Nueva reserva, Servicios, Preferencias, Soporte | Navegación del portal |
| Saludo personalizado | Dashboard principal | Mensaje con nombre del huésped | Personalización |
| Resumen de reservas | Dashboard principal | Tarjetas visuales con próximas reservas | Visualización de estado |
| Accesos rápidos | Dashboard principal | Botones para servicios populares | Acceso directo a funciones |
| Recomendaciones | Dashboard principal | Sugerencias basadas en estancias anteriores | Personalización |
| Estado de fidelización | Dashboard principal | Información del programa de lealtad | Seguimiento de beneficios |
| Filtros de reservas | Sección Mis Reservas | Opciones para filtrar por estado | Organización de información |
| Lista de reservas | Sección Mis Reservas | Información condensada de cada reserva | Visualización de historial |
| Opciones por reserva | Sección Mis Reservas | Botones para ver detalles, modificar o cancelar | Gestión de reservas |
| Formulario de búsqueda | Sección Nueva Reserva | Campos para fechas, tipo de habitación, etc. | Iniciar proceso de reserva |
| Resultados de búsqueda | Sección Nueva Reserva | Lista de habitaciones disponibles con filtros | Selección de opciones |
| Proceso guiado | Sección Nueva Reserva | Pasos secuenciales para completar reserva | Asistencia al usuario |
| Resumen final | Sección Nueva Reserva | Vista previa de la reserva antes de confirmar | Verificación de datos |

# 3.3.2 Descripción de los contenidos a mostrar en pantalla del rol <Recepcionista>

|  |  |  |  |
| --- | --- | --- | --- |
| Componente | Ubicación | Descripción | Funcionalidad |
| Barra superior | Parte superior | Logo, nombre del recepcionista, indicador de turno, reloj, contador | Contexto operativo |
| Buscador global | Barra superior | Campo de búsqueda para reservas y huéspedes | Acceso rápido a información |
| Panel lateral | Izquierda | Menú con opciones: Dashboard, Gestión de reservas, Check-in/Check-out, Habitaciones, Cobros, Mensajes, Incidencias | Navegación principal |
| Vista calendario | Dashboard principal | Calendario con ocupación diaria codificada por colores | Visualización de estado |
| Listado de llegadas/salidas | Dashboard principal | Tabla con huéspedes programados para el día | Planificación diaria |
| Alertas de mantenimiento | Dashboard principal | Notificaciones sobre habitaciones en mantenimiento | Prevención de problemas |
| Notificaciones especiales | Dashboard principal | Solicitudes especiales de huéspedes | Atención personalizada |
| Buscador avanzado | Gestión de reservas | Filtros múltiples para localizar reservas | Búsqueda eficiente |
| Vista tabular | Gestión de reservas | Tabla ordenable con todas las reservas | Visualización de datos |
| Formulario de reservas | Gestión de reservas | Campos para crear/modificar reservas | Edición de información |
| Lista check-in/check-out | Sección Check-in/out | Huéspedes pendientes de llegada/salida | Gestión de flujo |
| Formulario de registro | Sección Check-in/out | Campos para completar registro de huéspedes | Procesamiento de entradas |
| Asignación de habitaciones | Sección Check-in/out | Interfaz visual para asignar habitaciones | Gestión de ocupación |
| Procesamiento de pagos | Sección Check-in/out | Interfaz para cobros y depósitos | Gestión financiera |
| Plano visual | Sección Habitaciones | Mapa del hotel con estado de habitaciones | Gestión visual |
| Filtros de habitaciones | Sección Habitaciones | Opciones por tipo, estado y planta | Segmentación |
| Gestión de cambios | Sección Habitaciones | Interfaz para cambios de habitación | Reasignación |
| Comunicación limpieza | Sección Habitaciones | Sistema para notificar al departamento de limpieza | Coordinación |

# 3.3.3 Descripción de los contenidos a mostrar en pantalla del rol <Gerente>

|  |  |  |  |
| --- | --- | --- | --- |
| Componente | Ubicación | Descripción | Funcionalidad |
| Barra superior | Parte superior | Logo, nombre del gerente, selector de periodos, configuración, notificaciones | Contexto administrativo |
| Panel lateral | Izquierda | Menú con opciones: Dashboard ejecutivo, Análisis, Tarifas, RR.HH., Inventario, Configuración, Seguridad | Navegación principal |
| KPIs principales | Dashboard ejecutivo | Indicadores clave: ocupación, RevPAR, ADR, comparativas | Visión general |
| Gráficos de tendencias | Dashboard ejecutivo | Visualizaciones de ocupación e ingresos | Análisis visual |
| Alertas estratégicas | Dashboard ejecutivo | Notificaciones sobre desviaciones de objetivos | Control de gestión |
| Previsiones | Dashboard ejecutivo | Proyecciones basadas en reservas y tendencias | Planificación |
| Informes departamentales | Sección Análisis | Datos detallados por áreas operativas | Evaluación detallada |
| Análisis de rentabilidad | Sección Análisis | Métricas por tipo de habitación | Optimización |
| Estadísticas de canales | Sección Análisis | Rendimiento de canales de reserva | Distribución |
| Satisfacción de clientes | Sección Análisis | Datos de encuestas y valoraciones | Calidad de servicio |
| Exportación de informes | Sección Análisis | Opciones para diferentes formatos | Portabilidad de datos |
| Panel de precios | Sección Tarifas | Control de precios dinámicos | Gestión de ingresos |
| Configuración de temporadas | Sección Tarifas | Definición de temporadas y eventos | Planificación |
| Comparativa competencia | Sección Tarifas | Datos de precios de competidores | Posicionamiento |
| Simulador de escenarios | Sección Tarifas | Herramienta de predicción | Toma de decisiones |
| Gestión de turnos | Sección RR.HH. | Calendario y asignación de personal | Administración |
| Rendimiento del personal | Sección RR.HH. | Métricas por empleado | Evaluación |
| Asignación de permisos | Sección RR.HH. | Gestión de roles en el sistema | Seguridad |
| Formación y evaluación | Sección RR.HH. | Seguimiento de capacitación | Desarrollo |

# 3.4 Wireframes principales

|  |  |  |
| --- | --- | --- |
| Pantalla | Descripción | Elementos clave |
| Bienvenida | Diseño de página inicial | Logo, menú, buscador, carrusel |
| Dashboard Huésped | Vista principal del huésped | Reservas activas, recomendaciones, servicios |
| Dashboard Recepcionista | Centro de control operativo | Llegadas/salidas del día, ocupación, alertas |
| Dashboard Gerente | Panel ejecutivo | KPIs, gráficos, previsiones, alertas |
| Proceso de reserva | Flujo de creación de reserva | Pasos secuenciales, formularios, confirmación |
| Check-in | Interfaz de registro de entrada | Datos del huésped, asignación de habitación, cobro |
| Gestión de habitaciones | Mapa de estado | Plano visual, leyenda de estados, filtros |
| Informes analíticos | Pantalla de reportes | Gráficos, tablas, filtros, opciones de exportación |

# 3.5 Conclusiones Parciales

# El diseño de la Vista de Presentación del Sistema de Gestión Hotelera se ha desarrollado siguiendo un enfoque centrado en el usuario, con interfaces específicas para cada rol que responden a sus necesidades particulares. La arquitectura visual implementa principios de diseño responsivo, garantizando la adaptabilidad a diferentes dispositivos y contextos de uso.

# Las interfaces propuestas cumplen con todos los requisitos funcionales establecidos (RF01-RF06), distribuyendo las funcionalidades de manera lógica entre los diferentes roles. Se ha priorizado la usabilidad y eficiencia operativa, reduciendo la curva de aprendizaje mediante un diseño intuitivo y consistente en todas las secciones del sistema.

# La implementación de estas interfaces en React permitirá aprovechar las ventajas de una arquitectura basada en componentes, facilitando el mantenimiento y la escalabilidad del sistema a medida que evolucionen las necesidades del negocio hotelero.

# Capítulo IV. Vista Lógica

## Introducción

## La Vista Lógica presenta la estructura organizativa del Sistema de Gestión Hotelera desde una perspectiva arquitectónica, describiendo cómo se distribuyen las responsabilidades entre los componentes de software. Esta vista es fundamental para entender las relaciones entre las distintas clases, subsistemas y patrones que conforman la solución.

## Nuestro sistema requiere una arquitectura sólida que facilite la mantenibilidad, escalabilidad y colaboración entre equipos, por lo que hemos adoptado un enfoque modular con una clara separación de responsabilidades.

## La Vista Lógica proporciona las bases para comprender cómo el sistema:

## Estructurará los datos de reservas, habitaciones, pagos e informes

## Establecerá las interacciones entre componentes

## Implementará patrones de diseño para resolver problemas recurrentes

## Garantizará la consistencia y coherencia de la información

## 4.1 MVC – Modelo Vista Controlador

## El Modelo-Vista-Controlador es el patrón arquitectónico central de nuestro sistema, implementado de la siguiente manera:

## Modelo (Backend - Django):

## Representa los datos y la lógica de negocio

## Implementado con Django ORM que facilita la interacción con PostgreSQL

## Contiene las entidades principales: Reserva, Habitación, Pago, Usuario

## Mantiene las reglas de negocio y validaciones

## Vista (Frontend - React):

## Interfaz de usuario adaptada a diferentes roles (huésped, recepcionista, gerente)

## Componentes React que presentan información y capturan acciones del usuario

## Diseño responsivo compatible con dispositivos móviles y escritorio

## Implementación mediante componentes reutilizables

## Controlador (Backend - Django REST Framework):

## Procesa las peticiones del cliente y orquesta el flujo de trabajo

## Coordina la comunicación entre la Vista y el Modelo

## Implementado como API RESTful usando Django REST Framework

## Maneja la autenticación, autorización y validación de datos

## Esta arquitectura permite una clara separación de responsabilidades, facilitando el desarrollo paralelo por equipos especializados y simplificando el mantenimiento futuro del sistema.

## 4.2 Diagramas de clases

**4.2.1 Diagrama de clases del subsistema o paquete <Paquete de Gestión de Reservas>**

**4.2.2 Diagrama de clases del subsistema o paquete <Paquete de Gestión de Habitaciones>**

**4.2.3 Diagrama de clases del subsistema o paquete <Paquete de Gestión de Pagos>**

**4.2.4 Diagrama de clases del subsistema o paquete <Paquete de Reportes e Informes>**

## 4.3 Patrones de diseño

### 4.3.1 Patrones GRASP

En nuestro Sistema de Gestión Hotelera implementamos los siguientes patrones GRASP para garantizar una asignación adecuada de responsabilidades:

**Experto en Información:**

* Asignamos responsabilidades a las clases que poseen la información necesaria
* Ejemplo: La clase Reserva calcula la duración de la estancia y el precio total porque contiene las fechas y detalles de habitaciones
* Beneficio: Mayor cohesión y encapsulamiento de datos con comportamiento relacionado

**Creador:**

* Las clases gestoras (GestorReservas, GestorHabitaciones, etc.) son responsables de crear instancias de los objetos que gestionan
* Ejemplo: GestorPagos crea instancias de Pago, Factura y Recibo
* Beneficio: Centralización de la lógica de creación y mejor control de ciclo de vida

**Controlador:**

* Los controladores coordinan las acciones entre la interfaz y el modelo
* Ejemplo: ControladorReservas maneja las peticiones de la UI y coordina con GestorReservas
* Beneficio: Descarga a la interfaz de usuario de responsabilidades de dominio

**Alta Cohesión:**

* Cada clase tiene responsabilidades estrechamente relacionadas
* Ejemplo: GestorHabitaciones solo maneja funcionalidad relacionada con habitaciones
* Beneficio: Mantenibilidad y comprensibilidad mejoradas

**Bajo Acoplamiento:**

* Reducimos dependencias entre clases utilizando interfaces y abstracciones
* Ejemplo: Los gestores interactúan con interfaces y no con implementaciones concretas
* Beneficio: Mayor flexibilidad y adaptabilidad ante cambios

**4.3.2 Patrones GOF**

Implementamos los siguientes patrones GOF (Gang of Four) para resolver problemas recurrentes de diseño:

**Singleton:**

* Aplicado a los gestores para garantizar una única instancia
* Ejemplo: GestorPagos es singleton para centralizar procesamiento de pagos
* Implementación: Constructor privado y método estático getInstance()

**Factory Method:**

* Utilizado para crear diferentes tipos de informes sin exponer la lógica de creación
* Ejemplo: GestorInformes.crearInforme(tipo, parametros) crea instancias específicas
* Beneficio: Desacoplamiento entre creación y uso de objetos

**Observer:**

* Implementado para notificar cambios en el estado de reservas o pagos
* Ejemplo: Cuando una reserva cambia a estado "CONFIRMADA", se notifica automáticamente
* Beneficio: Comunicación desacoplada entre componentes

**Strategy:**

* Aplicado para manejar diferentes métodos de pago
* Ejemplo: Estrategias distintas para procesar pagos con tarjeta, efectivo o transferencia
* Beneficio: Intercambiabilidad de algoritmos sin modificar el contexto

**Template Method:**

* Implementado en la clase abstracta Informe
* Ejemplo: Define el proceso general de generación de informes con pasos específicos implementados por subclases
* Beneficio: Reutilización de código común manteniendo flexibilidad

## 4.4 Diagramas de Interacción

### 4.4.1 Diagrama de secuencia del Caso de uso <Realizar nueva reserva>. <Flujo principal>

### 4.4.2 Diagrama de secuencia del Caso de uso <Modificar reserva existente>. <Flujo principal>

### 4.4.3 Diagrama de secuencia del Caso de uso <Cancelar reserva>. <Flujo principal>

### 4.4.4 Diagrama de secuencia del Caso de uso <Check-in de huésped>. <Flujo principal>

### 4.4.5 Diagrama de secuencia del Caso de uso <Check-out de huésped>. <Flujo principal>

### 4.4.6 Diagrama de secuencia del Caso de uso <Procesar pago>. <Flujo principal>

### 4.4.7 Diagrama de secuencia del Caso de uso <Generar informe de ocupación>. <Flujo principal>

## 4.5 Modelo de Datos

## Conclusiones parciales

# Capítulo V. Vista de Implementación

## Introducción

La Vista de Implementación constituye un componente esencial en la arquitectura del sistema de gestión hotelera, ya que permite visualizar la organización estructural del software a nivel de componentes y sus relaciones. Este capítulo presenta el conjunto de artefactos que conforman la implementación física del sistema, mostrando cómo se traducen los aspectos conceptuales y lógicos en elementos concretos de software.

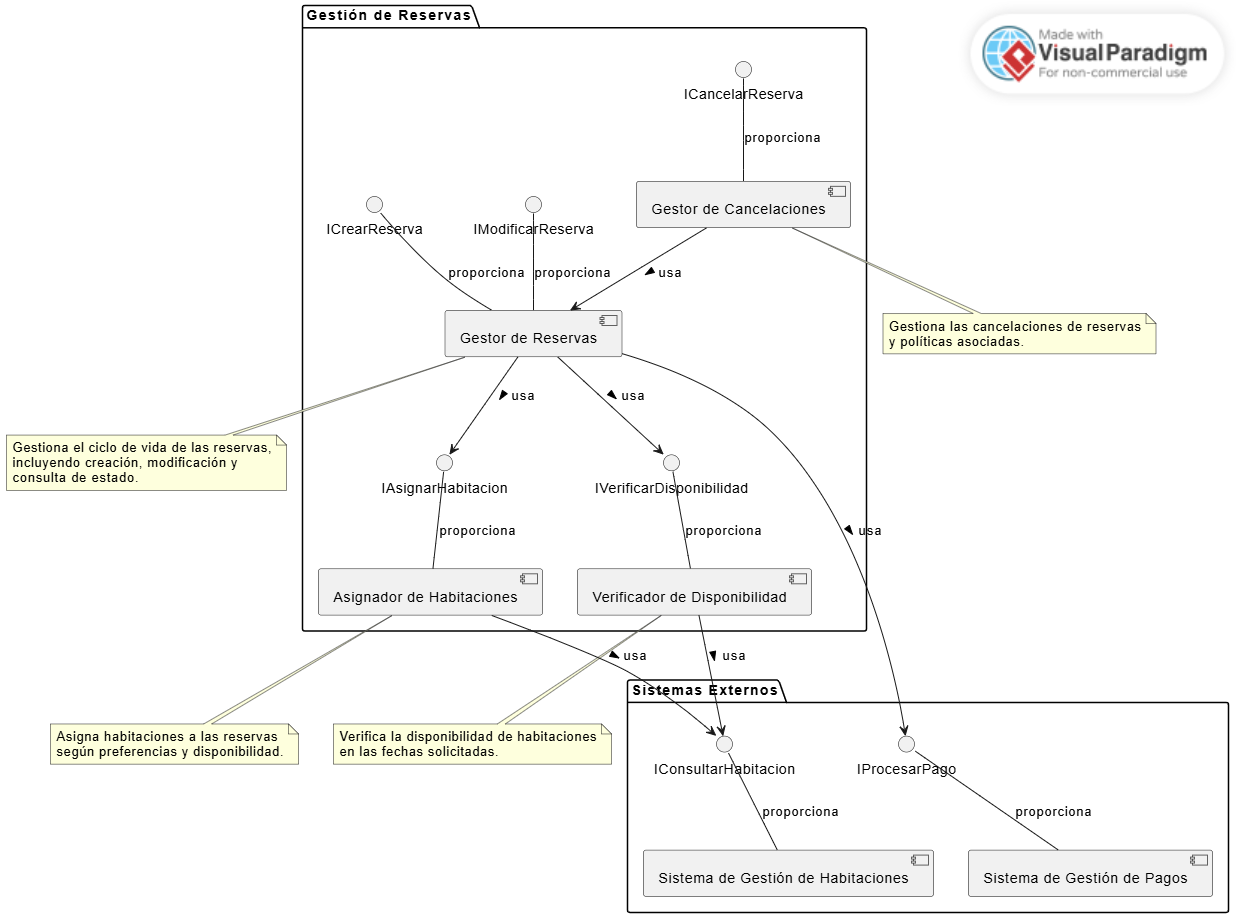
En esta vista se plasma la descomposición del sistema en componentes modulares, facilitando la comprensión de la estructura interna del software y sus dependencias. Los diagramas de componentes aquí presentados permiten visualizar la organización de los subsistemas desarrollados (Gestión de Reservas, Gestión de Habitaciones, Gestión de Pagos, y Reportes e Informes), mostrando las interfaces que exponen y los servicios que consumen.

La relevancia de esta vista radica en su capacidad para evidenciar aspectos críticos como la modularidad, la reutilización y el bajo acoplamiento entre los diferentes elementos del sistema. Mediante la identificación clara de los componentes y sus relaciones, se facilita no solo el desarrollo paralelo por equipos independientes, sino también el posterior mantenimiento y evolución del software ante nuevos requisitos o cambios tecnológicos.

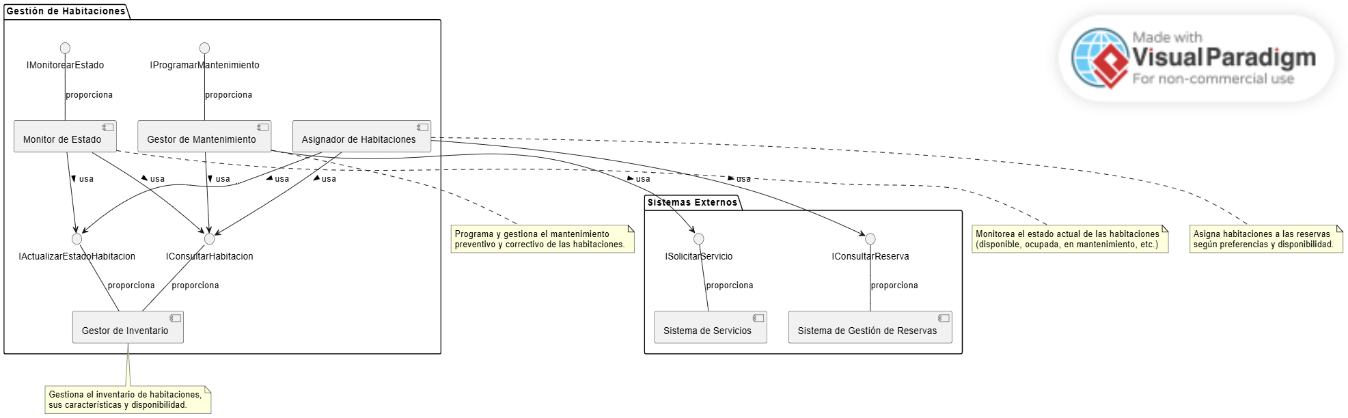
En el contexto específico de nuestro sistema de gestión hotelera, la Vista de Implementación resulta fundamental para garantizar la correcta integración entre el backend desarrollado en Django, el frontend en React y la base de datos PostgreSQL, asegurando que cada componente cumpla con sus responsabilidades de manera cohesiva y eficiente.

## Diagrama de Componentes

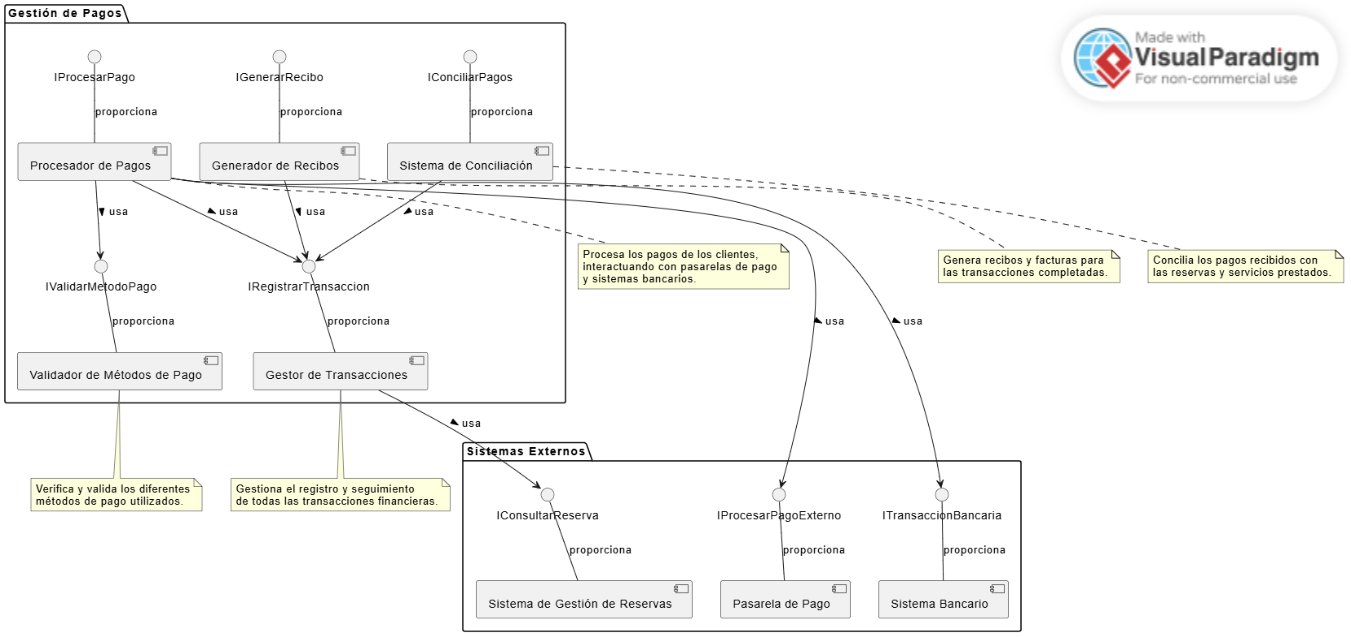
* + 1. **Diagrama de Componentes del subsistema o paquete < Paquete de Gestión de Reservas >**

****

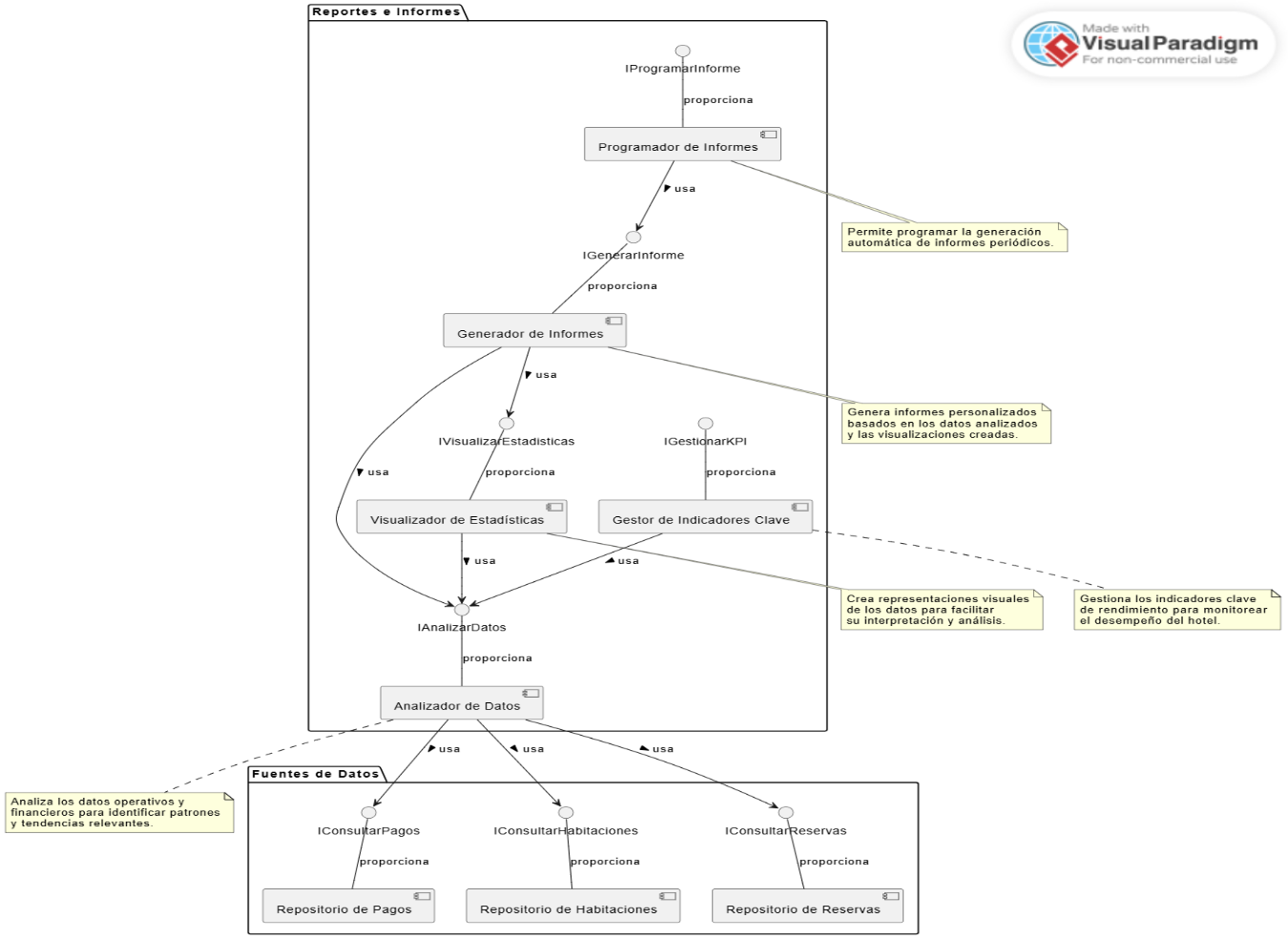
* + 1. **Diagrama de Componentes del subsistema o paquete < Paquete de Gestión de Habitaciones >**

****

* + 1. **Diagrama de Componentes del subsistema o paquete < Paquete de Gestión de Pagos >**

****

* + 1. **Diagrama de Componentes del subsistema o paquete < Paquete de Reportes e Informes>**

****

## Conclusiones parciales

Los diagramas de componentes desarrollados para los subsistemas de Gestión de Pagos y Reportes e Informes complementan la arquitectura del sistema de gestión hotelera, proporcionando una visión completa de su estructura modular.

El subsistema de **Gestión de Pagos** se estructura en torno a componentes especializados como el Procesador de Pagos, Gestor de Transacciones y Generador de Recibos, que trabajan coordinadamente para manejar todo el ciclo de vida de las transacciones económicas. La integración con sistemas externos bancarios y pasarelas de pago garantiza la seguridad y eficiencia en el procesamiento de pagos, cumpliendo con el requisito funcional RF04 (Procesamiento de Pagos).

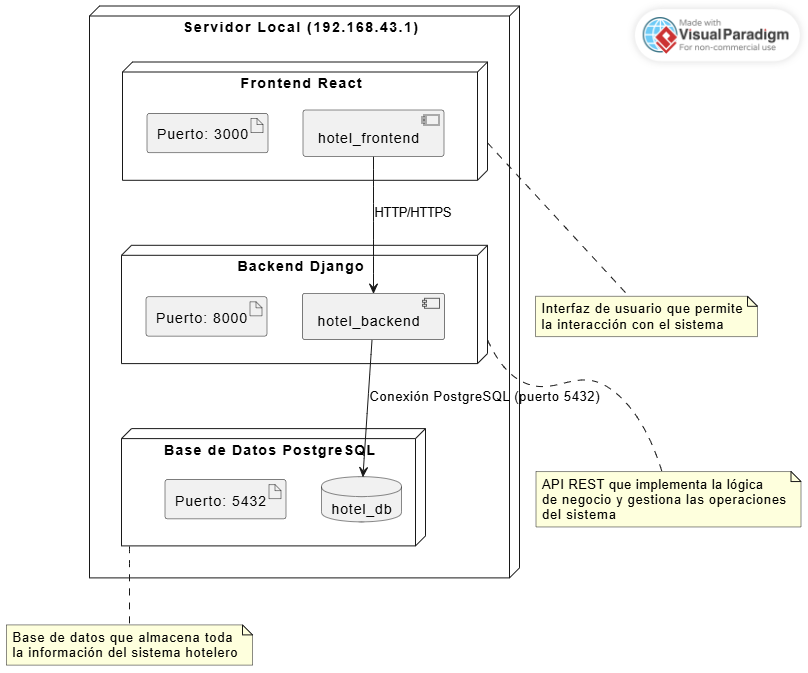
Por su parte, el subsistema de **Reportes e Informes** se organiza en componentes como el Generador de Informes, Analizador de Datos y Visualizador de Estadísticas, que permiten transformar los datos operativos en información valiosa para la toma de decisiones. Este subsistema responde directamente al requisito funcional RF05 (Generación de Informes) y facilita el análisis de indicadores clave de rendimiento del hotel.

La clara separación de responsabilidades en cada componente y la definición de interfaces estandarizadas favorecen un diseño de alta cohesión y bajo acoplamiento, siguiendo los principios GRASP establecidos en la arquitectura. Esta modularidad facilita no solo el desarrollo paralelo, sino también el mantenimiento y la evolución del sistema ante nuevos requisitos del negocio hotelero.

# Capítulo VI. Vista de Despliegue

## Introducción

## 6.1 Diagrama de Despliegue



### 6.1.1 Nombre de dispositivo: descripción de la capacidad que el dispositivo provee al sistema

### 6.1.2 Nombre del procesador: descripción de la funcionalidad y capacidad del nodo

### 6.1.3 <<Ethernet>>: Características físicas de la conexión

## Conclusiones parciales

# Capítulo VII. Prueba

## Introducción

## 7.1 Diseño de casos de prueba con caja negra

### 7.1.1 Descripción General

### 7.1.2 Condiciones de Ejecución

### 7.1.3 Secciones a probar en el Caso de Uso

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Id del escenario** | **Escenario** | **Variable 1**  **Nombre** | **Variable 2**  **Edad** | **Variable n**  **País** | **Respuesta del Sistema** | **Flujo central** |
| *EC 1* |  |  |  |  |  |  |
| *EC 1.1* |  |  |  |  |  |  |
| *EC 1.2* |  |  |  |  |  |  |
| *EC 1.3* |  |  |  |  |  |  |
| *EC 2* |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

## 7.2 Métodos a probar de las secciones

|  |  |
| --- | --- |
| **Método:** | **Grafo resultante:** |
|  |  |
| **Complejidad Ciclomática:** | |
|  | |

## 7.3 Diseño de casos de prueba con caja blanca

|  |
| --- |
| **Nombre del caso de uso:** |
|  |
| **Caso de prueba:** |
|  |
| **Camino:** |
|  |
| **Entrada:** |
|  |
| **Resultados:** |
|  |
| **Condiciones:** |
|  |

## 7.4 Registro de defectos y dificultades detectados

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Elemento** | **No** | **No conformidad** | **Aspecto correspondiente** | **Significativa** | **No Significativa** | **Recomendación** | **Estado NC** | **Respuesta del equipo de desarrollo** |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

## Conclusiones parcialesConclusiones generales

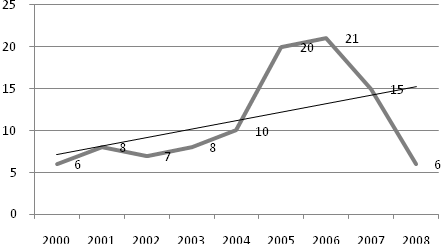
**Recomendaciones**

# Referencias Bibliográficas

# Bibliografía Consultada

# Anexos

*[Las imágenes a utilizar deben de estar centradas en el documento y con el formato como muestra el siguiente ejemplo:]*



**Figura 1.** El título de las figuras se colocará en la parte inferior, centrado, utilizando numeración secuencial según el orden en que aparecen en el trabajo (Arial a 10 puntos)

|  |  |  |  |
| --- | --- | --- | --- |
|  | Columna 1 | Columna 2 | Columna 3 |
| Fila 1 | x | z | y |
| Fila 2 | y | x | z |

**Tabla 1.** El título de las tablas en la parte inferior, centrado, utilizando numeración secuencial según el orden en que aparecen en el trabajo (Arial a 10 puntos)