

**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 Milán

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.

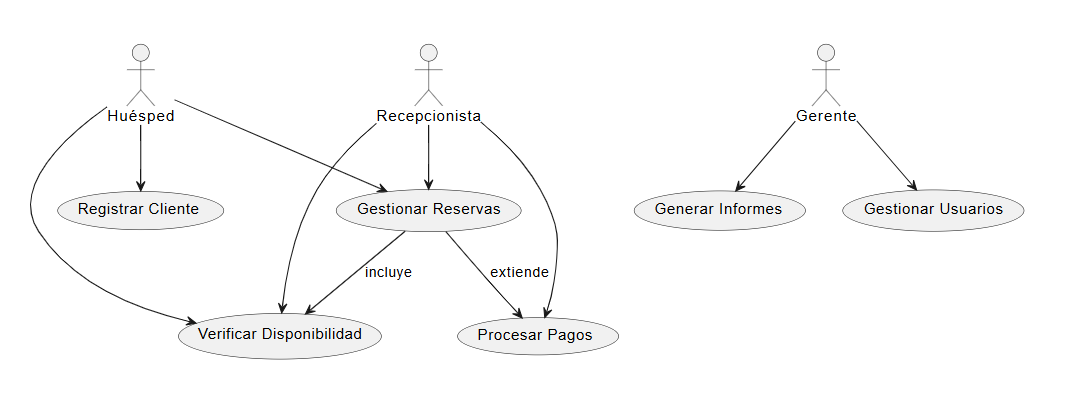
**1.3 Tecnologias Utilizadas**

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

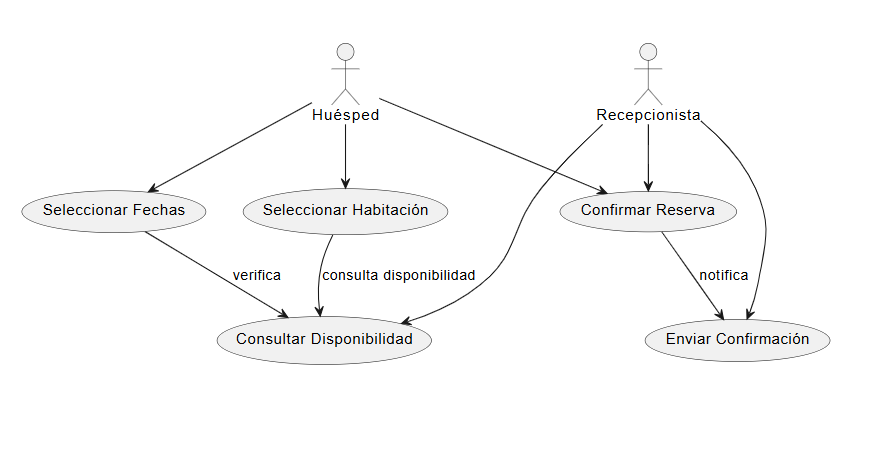
* Backend: Django (Python) por su robustez, escalabilidad y soporte para ORM, facilitando la integración con bases de datos.
* Frontend: React (Javascript) para una interfaz dinámica y responsiva, aprovechando su arquitectura modular y enlace de datos bidireccional.
* Base de datos: PostgreSQL, gestionado 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

## 3.2 Definición de contenidos pantalla de bienvenida

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

## 3.3 Definición de contenidos pantallas genéricas por roles

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

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

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

# Capítulo IV. Vista Lógica

## Introducción

## 4. **MVC (Modelo-Vista-Controlador)**

## 4.2 Diagramas de clases

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

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Objetivo** | |  | |
| **Actores** | |  | |
| **Resumen** | |  | |
| **Complejidad** | |  | |
| **Prioridad** | |  | |
| **Precondiciones** | |  | |
| **Postcondiciones** | |  | |
| **Flujo de eventos** | | | |
| **Flujo básico** | | | |
| **Actor** | | | **Sistema** |
|  |  | |  |
|  |  | |  |
|  |  | |  |
|  |  | |  |
| **Sección 1:** | | | |
| **Flujo básico** | | | **Sistema** |
| 1. **Actor** |  | |  |
|  |  | |  |
|  |  | |  |
|  |  | |  |
|  |  | |  |
|  | | | |
|  | | | |
|  | | | |

|  |  |  |
| --- | --- | --- |
| **Sección 2: “Eliminar Jugador”** | | |
| **Flujo básico Eliminar Jugador** | | |
| **Actor** | | **Sistema** |
| 1. | Presiona el botón “Eliminar Jugador”. |  |
| 2. |  | Muestra una ventana donde se coloca el ID del contrato a eliminar |
| 3. | Insertar el ID del jugador a eliminar y presionar el botón “Eliminar” |  |
| 4. |  | Verifica que todos los campos estén llenos. |
| 5. |  | Verifica que el contrato exista |
| 6. |  | Eliminar contrato con el ID seleccionado  Mandar el mensaje ”Contrato Eliminado”  *Finalizando caso de uso.* |
|  |  |  |
| Prototipo elemental de interfaz gráfica. | | |
|  | | |

## 4.3 Patrones de diseño

### 4.3.1 Patrones GRASP

### 4.3.2 Patrones GOF

## 4.4 Diagramas de Interacción

### 4.4.1 Diagrama de secuencia del Caso de uso <Gestión de Jugador>. <Agregar Jugador>

### 4.4.2 Diagrama de secuencia del Caso de uso <Gestión de Jugador>. <Eliminar Jugador>

### 4.4.2 Diagrama de secuencia del Caso de uso <Gestión de Jugador>. <Modificar Jugador>

## 4.5 Modelo de Datos

## Conclusiones parciales

# Capítulo V. Vista de Implementación

## Introducción

## 5.1 Diagrama de Componentes

**5.1.1 Diagrama de Componentes del subsistema o paquete <>**

**5.1.2 Diagrama de Componentes del subsistema o paquete <>**

## Conclusiones parciales

# Capítulo VI. Vista de Despliegue

## Introducción

## 4.1 Diagrama de Despliegue

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

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

### 4.1.3 <<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

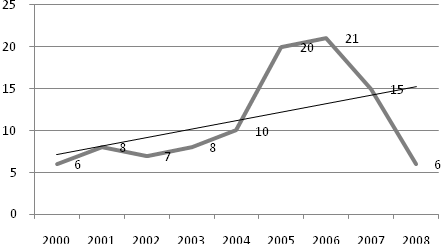
**Recomendaciones**

# 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)