SkyStay

**ÍNDICE (Provisional)**

**1. INTRODUCCIÓN**

1.1. Presentación del proyecto  
1.2. Motivación y justificación  
1.3. Objetivos generales y específicos  
1.4. Metodología de desarrollo y herramientas utilizadas

**2. PLANIFICACIÓN Y ANÁLISIS**

2.1. Definición del problema  
2.2. Requisitos del sistema  
2.2.1. Requisitos funcionales  
2.2.2. Requisitos no funcionales  
2.3. Análisis de viabilidad  
2.3.1. Estudio de tecnologías  
2.3.2. Recursos necesarios  
2.3.3. Planificación temporal del desarrollo

**3. DISEÑO DE LA ARQUITECTURA DEL SISTEMA**

3.1. Modelo de arquitectura (cliente-servidor)  
3.2. Diseño de la API REST  
3.2.1. Definición de endpoints  
3.2.2. Seguridad y autenticación (JWT, roles de usuario)  
3.3. Modelado de la base de datos  
3.3.1. Diagrama entidad-relación (ER)  
3.3.2. Estructura y normalización de tablas  
3.4. Diseño del frontend  
3.4.1. Wireframes y prototipos  
3.4.2. UX/UI: principios aplicados

**4. IMPLEMENTACIÓN DEL BACKEND**

4.1. Configuración del entorno y dependencias (Spring Boot, Maven)  
4.2. Creación del modelo de datos en MariaDB  
4.3. Implementación de controladores y servicios  
4.4. Seguridad y autenticación con Spring Security y JWT  
4.5. Gestión de reservas y usuarios  
4.6. Envío de correos electrónicos con notificaciones  
4.7. Generación de informes y estadísticas

**5. IMPLEMENTACIÓN DEL FRONTEND**

5.1. Configuración del entorno y dependencias (Angular)  
5.2. Desarrollo de la estructura de componentes  
5.3. Conexión con la API REST  
5.4. Gestión de sesiones y autenticación  
5.5. Diseño de las vistas y experiencia de usuario  
5.6. Implementación de funcionalidades principales  
5.6.1. Consulta de disponibilidad y reservas  
5.6.2. Sistema de fidelización (puntos y recompensas)  
5.6.3. Gestión de reservas y cancelaciones

**6. PRUEBAS Y DEPURACIÓN**

6.1. Pruebas unitarias (JUnit)  
6.2. Pruebas de integración (Postman, Swagger)  
6.3. Pruebas de carga y rendimiento  
6.4. Pruebas de usabilidad y accesibilidad  
6.5. Estrategias de depuración y resolución de errores

**7. DESPLIEGUE Y MANTENIMIENTO**

7.1. Contenedorización con Docker  
7.2. Despliegue en servidores y configuración  
7.3. Monitorización y gestión de logs  
7.4. Estrategia de mantenimiento y actualizaciones

**8. RESULTADOS Y CONCLUSIONES**

8.1. Comparación entre objetivos iniciales y resultados obtenidos  
8.2. Problemas encontrados y soluciones aplicadas  
8.3. Valoración del rendimiento del sistema  
8.4. Posibles mejoras y ampliaciones futuras

**9. BIBLIOGRAFÍA Y REFERENCIAS**

**10. ANEXOS**

10.1. Código fuente relevante  
10.2. Documentación técnica de la API (Swagger)  
10.3. Manual de usuario y administrador  
10.4. Planificación temporal detallada

Aquí voy a realizar la memoria del proyecto,

**1. INTRODUCCIÓN**

**1.1. Presentación del Proyecto**

**SkyStay** es una plataforma web desarrollada para gestionar reservas en una cadena hotelera. El sistema se divide en dos partes:

* **Frontend:** Interfaz web intuitiva para que los usuarios puedan consultar disponibilidad, gestionar reservas y acceder a funcionalidades como el sistema de fidelización.
* **Backend:** Encargado de procesar la lógica del negocio, gestionar los datos y mantener la comunicación con la base de datos mediante una API REST desarrollada en Spring Boot.

El sistema contempla tanto la gestión de reservas por parte de los clientes como funcionalidades administrativas destinadas a los gestores de los hoteles.

**1.2. Motivación y Justificación**

En un entorno cada vez más digitalizado, las empresas hoteleras demandan soluciones modernas que agilicen la gestión de sus servicios y mejoren la experiencia del cliente. SkyStay surge para cubrir esta necesidad ofreciendo:

* Una plataforma unificada y centralizada para cadenas hoteleras.
* Interfaz amigable para el cliente y funcionalidades avanzadas para administradores.
* Sistema de recompensas que mejora la fidelización.

El objetivo es crear un sistema eficaz, seguro y moderno que responda a las exigencias del sector hotelero actual.

**1.3. Objetivos Generales y Específicos**

**Objetivo General:** Desarrollar una plataforma completa de gestión de reservas hoteleras que integre frontend, backend y base de datos.

**Objetivos Específicos:**

1. Crear una API REST con Spring Boot para la gestión de reservas.
2. Implementar autenticación y autorización diferenciada para clientes y administradores.
3. Permitir reservas en tiempo real y gestión de recompensas.
4. Desarrollar un frontend responsive e intuitivo en Angular.
5. Incluir funcionalidades de cancelación y modificación de reservas.
6. Incorporar notificaciones por correo para eventos clave.
7. Asegurar la protección de los datos personales y transacciones.
8. Generar estadísticas de ocupación para los administradores.

**1.4. Metodología de Desarrollo y Herramientas Utilizadas**

**Metodología:**  
Se empleará una metodología ágil basada en **iteraciones semanales** con entregables funcionales, revisiones constantes y pruebas continuas.

**Herramientas de Desarrollo:**

* **Backend:** Spring Boot 3.x
* **Frontend:** Angular
* **Base de Datos:** MariaDB
* **Control de versiones:** Git + GitHub
* **Pruebas:** JUnit (pruebas unitarias) + Postman (pruebas de API)
* **Documentación API:** Swagger
* **Despliegue:** Docker
* **IDE:** IntelliJ IDEA y Visual Studio Code

**2. PLANIFICACIÓN Y ANÁLISIS**

**2.1. Definición del Problema**

Las cadenas hoteleras que operan en múltiples localizaciones enfrentan dificultades al gestionar reservas de forma centralizada. Los sistemas tradicionales suelen carecer de integración, personalización o escalabilidad. SkyStay pretende solventar estas deficiencias proporcionando una solución web moderna y eficiente.

**2.2. Requisitos del Sistema**

**2.2.1. Requisitos Funcionales**

* Registro y autenticación de usuarios (clientes y administradores).
* Visualización de disponibilidad de habitaciones.
* Realización, cancelación y modificación de reservas.
* Canjeo de recompensas.
* Gestión de habitaciones, servicios y reservas por parte de los administradores.
* Envío automático de correos de confirmación y recordatorios.
* Generación de informes estadísticos para administradores.

**2.2.2. Requisitos No Funcionales**

* Sistema seguro con cifrado de datos sensibles.
* Accesibilidad desde distintos dispositivos (responsive design).
* Alta disponibilidad y rendimiento.
* Trazabilidad y registro de actividades.
* Despliegue en contenedores para facilitar la escalabilidad.

**2.3. Análisis de Viabilidad**

**2.3.1. Estudio de Tecnologías**

**Software:**

* Backend: Spring Boot
* Frontend: Angular
* Base de datos: MariaDB
* Despliegue: Docker
* Pruebas: JUnit, Postman
* Control de versiones: Git + GitHub
* Documentación: Swagger

**Hardware:**

* CPU: 4 núcleos o superior
* RAM: mínimo 8 GB
* Disco: mínimo 50 GB SSD
* Sistema operativo servidor: Windows

**2.3.2. Recursos Necesarios**

* Ordenadores de desarrollo con capacidad adecuada para ejecución local.
* Acceso a servidores para pruebas de despliegue.
* Licencias y entornos de desarrollo (IDE, herramientas de documentación, etc.).

**2.3.3. Planificación Temporal del Desarrollo**

| **Semana** | **Tarea** |
| --- | --- |
| 1 | Análisis de requisitos y diseño de arquitectura |
| 2-4 | Desarrollo del backend (Spring Boot) |
| 5-6 | Desarrollo del frontend (Angular) |
| 7-8 | Integración del backend con el frontend |
| 9-10 | Pruebas funcionales y corrección de errores |
| 11 | Despliegue en entorno real (Docker) |
| 12 | Documentación técnica y preparación de presentación final |