📄Documento de Especificación de Requerimientos del Software (ERS)

📆Sistema de Reservas de Hotel

---

1. Introducción

1.1 Propósito

Este documento describe los requisitos funcionales y no funcionales para el desarrollo de un sistema de reservas de hotel. El sistema permitirá a los clientes reservar habitaciones en línea, y proporcionará a recepcionistas y administradores herramientas para gestionar reservas, habitaciones y usuarios de forma eficiente.

1.2 Alcance

El sistema está orientado a un **solo hotel** y se desarrollará como una aplicación web.

- **Clientes** podrán registrarse, iniciar sesión, explorar habitaciones y realizar reservas.
- **Recepcionistas** gestionarán reservas y realizarán check-in/check-out.
- **Administradores** gestionarán usuarios, habitaciones y reportes básicos.

1.3 Definiciones, siglas y abreviaturas

- **MVP**: Producto Mínimo Viable.
- **JWT**: JSON Web Token.
- **BDD**: Base de Datos.
- **ERS**: Especificación de Requerimientos del Software.

1.4 Referencias

- Sprint 0 – Planeación y diseño del sistema.
- Diagramas de flujos de uso y modelo entidad-relación (ERD).

---

2. Descripción general

2.1 Perspectiva del producto

El sistema tendrá tres módulos principales:

1. **Módulo Cliente**
2. **Módulo Recepcionista**
3. **Módulo Administrador**

La arquitectura seguirá el patrón **MVC (Model-View-Controller)** para mayor claridad y mantenibilidad.

2.2 Funcionalidades del sistema

- Gestión de usuarios con roles y permisos.
- CRUD de habitaciones.
- Gestión de reservas con validación de disponibilidad.
- Envío de correos electrónicos de confirmación.
- Autenticación y autorización basada en JWT.

2.3 Restricciones

- La aplicación será desarrollada únicamente para entorno web.
- Solo soporte para idioma español en el MVP.

---

3. Requerimientos funcionales

3.1 Cliente

- **RF1.** El cliente podrá registrarse e iniciar sesión.
- **RF2.** El cliente podrá ver habitaciones disponibles con filtros por fecha, precio y capacidad.
- **RF3.** El cliente podrá realizar reservas seleccionando fechas y habitación.
- **RF4.** El cliente podrá cancelar o modificar sus reservas antes de la fecha de entrada.
- **RF5.** El cliente recibirá correos electrónicos de confirmación al realizar una reserva.

3.2 Recepcionista

- **RF6.** El recepcionista podrá iniciar sesión.
- **RF7.** El recepcionista podrá ver todas las reservas activas y pasadas.
- **RF8.** El recepcionista podrá crear, modificar y cancelar reservas a nombre de clientes.
- **RF9.** El recepcionista podrá gestionar el check-in y check-out de clientes.

3.3 Administrador

- **RF10.** El administrador podrá crear, editar y eliminar habitaciones.
- **RF11.** El administrador podrá crear y desactivar cuentas de recepcionistas.
- **RF12.** El administrador podrá ver reportes básicos de ocupación y reservas.

---

4. Requerimientos no funcionales

- **RNF1.** El sistema debe ser accesible desde los navegadores modernos.
- **RNF2.** La autenticación debe implementar JWT para proteger endpoints.
- **RNF3.** Contraseñas deben almacenarse usando hash bcrypt.
- **RNF4.** El sistema debe tener un tiempo de respuesta menor a 2 segundos por petición.
- **RNF5.** La base de datos debe registrar timestamps de creación y actualización en todas las tablas.

---

5. Modelo de datos

5.1 Tabla usuarios

Campo	Tipo	Nota
`id_usuario`	UUID (PK)	Clave primaria
`nombre_completo`	VARCHAR(100)	Obligatorio
`email`	VARCHAR(100)	Único, validado, requiere verificación
`password_hash`	TEXT	Hash bcrypt
`telefono`	VARCHAR(20)	Opcional
`documento_identidad`	VARCHAR(20)	Opcional
`fecha_nacimiento`	DATE	Opcional
`rol`	ENUM	('admin', 'recepcionista', 'cliente')
`estado`	ENUM	('activo', 'inactivo')
`created_at`	TIMESTAMP	Fecha de creación
`updated_at`	TIMESTAMP	Fecha de última actualización

5.2 Tabla habitaciones

Campo	Tipo	Nota
`id_habitacion`	UUID (PK)	Clave primaria
`numero`	INT	Número de habitación
`descripcion`	TEXT	Opcional
`precio`	DECIMAL(10,2)	Precio por noche
`estado`	ENUM	('libre', 'ocupada', 'mantenimiento')
`created_at`	TIMESTAMP	Fecha de creación
`updated_at`	TIMESTAMP	Fecha de última actualización

5.3 Tabla reservas

Campo	Tipo	Nota
`id_reserva`	UUID (PK)	Clave primaria
`id_usuario`	UUID (FK)	Relación con cliente
`id_habitacion`	UUID (FK)	Relación con habitación
`fecha_inicio`	DATE	Fecha de entrada

Campo	Tipo	Nota
`fecha_fin`	DATE	Fecha de salida
`estado`	ENUM	('pendiente', 'confirmada', 'en curso', 'completada', 'cancelada')
`metodo_pago`	ENUM	('tarjeta', 'efectivo', 'transferencia', 'pendiente')
`estado_pago`	ENUM	('pagado', 'pendiente', 'rechazado')
`comentarios`	TEXT	Notas especiales
`created_at`	TIMESTAMP	Fecha de creación
`updated_at`	TIMESTAMP	Fecha de última actualización

---

6. Diagrama de arquitectura

El sistema seguirá un modelo cliente-servidor con:

- **Frontend:** Angular (consumiendo APIs RESTful)
- **Backend:** Node.js + Express
- **Base de datos:** PostgreSQL (ORM Prisma)
- **Autenticación:** JWT
- **Infraestructura:** Docker, Railway (Backend), Vercel (Frontend)

---

7. Criterios de aceptación

- Todas las funcionalidades deben estar probadas con Postman.
- El sistema debe cumplir con los casos de uso definidos.
- El código debe pasar revisiones de calidad (linting, pruebas unitarias).

---

8. Análisis de riesgos

Riesgo	Mitigación
Errores en reservas simultáneas	Validación backend para evitar solapamientos de fechas.
Acceso no autorizado a datos sensibles	Implementar middlewares de roles y JWT correctamente.
Falta de escalabilidad en el MVP	Modularizar código para futuras ampliaciones.

---

9. Entregables Sprint 0

✅ Documento ERS (este documento).\ ✅Diagramas de flujos de uso (Mermaid).\ ✅Diagrama entidad-relación (ERD).\ ✅Tablero Trello con tareas iniciales.\ ✅Repositorio Git inicial con estructura básica.\ ✅Planeacion SCRUM

🔗 Aprobación

- Fecha: [por definir]
- Versión: 1.0
- Aprobado por: [nombre del responsable]