INFORME TÉCNICO: PLATAFORMA DE RESERVA DE HOSPEDAJE

Equipo LAST PUSH
Juan Felipe Gómez (98265)
Susana Cai (106991)
Luciano Desimone (114055)
Alexis Herrera (111127)
Juan Francisco Solís (112796)
Stefano Diaz (113909)

November 24, 2025

1. Introducción

El presente informe detalla la arquitectura, desarrollo e implementación de una plataforma integral de reserva de hospedaje. Este proyecto representa una solución completa que integra tecnologías modernas de *frontend* y *backend* para proporcionar a los usuarios una experiencia fluida y eficiente en la búsqueda y reserva de alojamientos.

1.1. Descripción General del Proyecto

La plataforma "Hotel IDS" es una aplicación web diseñada para facilitar la reserva de alojamientos (hoteles, cabañas, departamentos) con los siguientes objetivos:

- Proporcionar una interfaz intuitiva para navegar disponibilidad de habitaciones
- Permitir reservas fáciles y seguras
- Mostrar información detallada con fotografías de alojamientos
- Facilitar contacto entre usuarios y la plataforma
- Mantener registro de historial de reservas
- Integrar herramientas interactivas como mapas

1.2. Objetivos del Proyecto

1. Crear una plataforma web moderna y responsiva
2. Implementar un sistema robusto de gestión de reservas
3. Desarrollar un backend seguro con autenticación de usuarios

4. Proporcionar una base de datos eficiente para almacenar información
5. Garantizar la escalabilidad y mantenibilidad del código

2. Tecnologías Utilizadas

2.1. Frontend

Tecnología	Descripción
Python/Flask	Framework web para renderizar templates HTML
HTML5	Estructura y semántica de las páginas
CSS3	Estilos, diseño responsivo y animaciones
Bootstrap	Framework CSS para componentes prediseñados
JavaScript	Interactividad del lado del cliente
Leaflet.js	Integración de mapas interactivos

2.2. Backend

Tecnología	Descripción
Python	Lenguaje de programación principal
Flask	Microframework web para APIs REST
Flask-CORS	Gestión de CORS entre frontend y backend
Flask-Mail	Sistema de notificaciones por correo
SQLAlchemy	ORM para interacción con base de datos
MySQL	Sistema de gestión de base de datos relacional

2.3. Herramientas de Desarrollo

- **Control de Versiones:** Git y GitHub
- **Documentación API:** OpenAPI/Swagger
- **Entorno Virtual:** Python venv
- **Gestor de Dependencias:** pip

3. Arquitectura del Sistema

3.1. Estructura General

La arquitectura del proyecto sigue un patrón de separación cliente-servidor:

- **Frontend:** Aplicación Flask que renderiza vistas HTML
- **Backend:** API REST desarrollada con Flask para gestionar datos
- **Base de Datos:** MYSQL para almacenamiento persistente

3.2. Estructura de Carpetas

```
IDS-TPFINAL/
          backend/
                app.py                # Punto de entrada del
   backend
                db.py                 # Configuraci n de base de
   datos
                requirements.txt      # Dependencias Python
                routes/
                      habitaciones.py  # API de habitaciones
                      reservas.py      # API de reservas
                      usuarios.py      # API de usuarios
                openapi.yaml          # Documentaci n de API
                init_db.py            # Scripts de inicializaci n
          frontend/
                app.py                # Aplicaci n Flask frontend
                requirements.txt
                static/
                      css/              # Estilos CSS
                      js/               # Scripts JavaScript
                      img/              # Im genes
                template/
                    base.html          # Template base
                    index.html         # P gina principal
                    rooms.html         # Listado de habitaciones
                    reservar.html      # P gina de reserva
                    login.html         # Login
                    register.html      # Registro
                    user.html          # Perfil de usuario
          README.md
```

4. Componentes Principales

4.1. Backend - API REST

4.1.1. Autenticación y Usuarios

La ruta `/usuarios` implementa:

- Registro de nuevos usuarios
- Login seguro con sesiones
- Gestión de perfiles de usuario
- Recuperación de contraseña

```
# Ejemplo de estructura de usuario
{
```

```
    "id": 1,
    "nombre": "Juan P rez",
    "email": "juan@example.com",
    "telefono": "+598 2 1234 5678",
    "historial_reservas": [...]
}
```

4.1.2. Gestión de Habitaciones

La ruta `/habitaciones` implementa:

- Listado de habitaciones disponibles
- Búsqueda y filtrado por características
- Consulta de disponibilidad por fechas
- Detalles completos con fotografías
- Información de precios y servicios

4.1.3. Sistema de Reservas

La ruta `/reservas` implementa:

- Creación de nuevas reservas
- Validación de disponibilidad
- Confirmación y generación de comprobantes
- Cancelación de reservas
- Historial de reservas por usuario

4.2. Frontend - Interfaz de Usuario

4.2.1. Páginas Principales

index.html Página principal con descripción del servicio y destacados

rooms.html Catálogo de habitaciones con filtros y búsqueda

reservar.html Formulario de reserva con selección de fechas

login.html Formulario de inicio de sesión

register.html Formulario de registro de nuevos usuarios

user.html Perfil de usuario y historial de reservas

4.2.2. Características Tecnológicas

- Diseño responsivo compatible con dispositivos móviles
- Interfaz intuitiva y accesible
- Animaciones CSS suaves
- Validación de formularios en cliente y servidor
- Integración de mapa interactivo con Leaflet.js

5. Base de Datos

5.1. Modelo de Datos

5.1.1. Tabla: Usuarios

Campo	Tipo	Descripción
id	INTEGER	Identificador único (PK)
nombre	VARCHAR	Nombre completo
email	VARCHAR	Correo electrónico (UNIQUE)
contraseña	VARCHAR	Contraseña hasheada
telefono	VARCHAR	Número de teléfono
$\text{fecha}_r egistro$	DATETIME	Fecha de registro

5.1.2. Tabla: Habitaciones

Campo	Tipo	Descripción
id	INTEGER	Identificador único (PK)
numero	VARCHAR	Número de habitación
tipo	VARCHAR	Tipo (individual, doble, suite)
capacidad	INTEGER	Número de huéspedes
precio	DECIMAL	Precio por noche
descripcion	TEXT	Descripción detallada
imagen	VARCHAR	URL de imagen
disponible	BOOLEAN	Estado de disponibilidad
servicios	TEXT	Servicios incluidos

5.1.3. Tabla: Reservas

Campo	Tipo	Descripción
id	INTEGER	Identificador único (PK)
usuario_id	INTEGER	Referencia a usuario (FK)
habitacion_id	INTEGER	Referencia a habitación (FK)
fecha$_i$*nicio*	DATE	Fecha de entrada
fecha$_f$*in*	DATE	Fecha de salida
estado	VARCHAR	Estado (pendiente, confirmada, cancelada)
precio$_t$*otal*	DECIMAL	Precio total de la reserva
fecha$_r$*eserva*	DATETIME	Fecha de creación

6. Funcionalidades Implementadas

6.1. Módulo de Usuarios

1. **Registro:** Formulario con validación de datos
2. **Login:** Autenticación segura con sesiones
3. **Perfil:** Visualización y edición de información personal
4. **Historial:** Vista de reservas anteriores
5. **Recuperación de Contraseña:** Enlace de recuperación por correo

6.2. Módulo de Habitaciones

1. **Catálogo:** Visualización de todas las habitaciones
2. **Búsqueda:** Filtrado por tipo, capacidad y precio
3. **Detalles:** Información completa con fotos y servicios
4. **Disponibilidad:** Verificación de fechas disponibles
5. **Calificaciones:** Sistema de puntuación de usuarios

6.3. Módulo de Reservas

1. **Creación:** Selección de fechas y habitación
2. **Validación:** Verificación de disponibilidad en tiempo real
3. **Confirmación:** Envío de correo de confirmación
4. **Gestión:** Modificación y cancelación de reservas
5. **Comprobante:** Generación de PDF con detalles

6.4. Características Adicionales

- **Mapa Interactivo:** Ubicación del hospedaje
- **Sistema de Contacto:** Formulario de consultas
- **Panel de Opiniones:** Comentarios de clientes
- **Notificaciones:** Recordatorios por correo
- **Soporte 24/7:** Chat o email de contacto

7. API REST

7.1. Endpoints Principales

7.1.1. Usuarios

Método	Endpoint	Descripción
POST	/usuarios/registro	Registrar nuevo usuario
POST	/usuarios/login	Iniciar sesión
GET	/usuarios/{id}	Obtener datos de usuario
PUT	/usuarios/{id}	Actualizar usuario
POST	/usuarios/logout	Cerrar sesión

7.1.2. Habitaciones

Método	Endpoint	Descripción
GET	/habitaciones	Listar todas las habitaciones
GET	/habitaciones/{id}	Obtener detalles de habitación
GET	/habitaciones/disponibles	Habitaciones disponibles
POST	/habitaciones/buscar	Buscar por criterios

7.1.3. Reservas

Método	Endpoint	Descripción
POST	/reservas	Crear nueva reserva
GET	/reservas/{id}	Obtener detalles de reserva
GET	/reservas/usuario/{id}	Historial de usuario
PUT	/reservas/{id}	Modificar reserva
DELETE	/reservas/{id}	Cancelar reserva

8. Seguridad

8.1. Medidas Implementadas

- **Autenticación:** Sistema de login con sesiones Flask
- **Contraseñas:** Hash seguro de contraseñas

- **CORS:** Configuración de CORS en backend
- **Validación:** Validación en servidor de todos los datos
- **Sanitización:** Protección contra inyecciones SQL
- **HTTPS:** (Recomendado en producción)

9. Instalación y Configuración

9.1. Requisitos Previos

- Python 3.8 o superior
- PostgreSQL 12 o superior
- Node.js (opcional, para herramientas de desarrollo)
- Git para control de versiones

9.2. Pasos de Instalación

9.2.1. Backend

```
# Clonar repositorio
git clone <repository-url>
cd IDS-TPFINAL/backend

# Crear entorno virtual
python3 -m venv venv
source venv/bin/activate  # En Windows: venv\Scripts\activate

# Instalar dependencias
pip install -r requirements.txt

# Crear archivo .env con variables
cp .env.example .env

# Inicializar base de datos
python init_db.py

# Ejecutar servidor
python app.py
```

9.2.2. Frontend

```
# Navegar a carpeta frontend
cd ../frontend

# Crear entorno virtual
```

```
python3 -m venv venv
source venv/bin/activate

# Instalar dependencias
pip install -r requirements.txt

# Ejecutar servidor
python app.py
```

9.3. Variables de Entorno

```
# .env archivo
FLASK_ENV=development
FLASK_SECRET_KEY=your_secret_key
DATABASE_URL=postgresql://user:password@localhost:5432/hotel_db
MAIL_USERNAME=your_email@gmail.com
MAIL_PASSWORD=your_app_password
```

10. Documentación API

10.1. OpenAPI/Swagger

La documentación interactiva de la API está disponible en:

`http://localhost:5010/api/docs`

Ver archivo `openapi.yaml` para especificación completa.

10.2. Ejemplo de Solicitud

```
# Crear una reserva
curl -X POST http://localhost:5010/reservas \
  -H "Content-Type: application/json" \
  -d '{
    "usuario_id": 1,
    "habitacion_id": 5,
    "fecha_inicio": "2025-12-01",
    "fecha_fin": "2025-12-05"
  }'
```

11. Gestión del Proyecto

11.1. Metodología

Se utilizó metodología Agile con sprints de una semana, permitiendo:

- Iteración rápida sobre funcionalidades

- Feedback constante del equipo
- Adaptación a cambios de requisitos
- Entrega incremental de valor

11.2. Herramientas de Colaboración

- **GitHub:** Control de versiones y colaboración
- **GitHub Issues:** Gestión de tareas
- **GitHub Projects:** Tablero Kanban
- **Slack/Discord:** Comunicación del equipo

12. Conclusiones

Este proyecto de Trabajo Práctico Final demuestra la capacidad del equipo LAST PUSH para:

- Diseñar y arquitectar una aplicación web moderna
- Implementar funcionalidades complejas en frontend y backend
- Trabajar colaborativamente en un proyecto de escala real
- Aplicar buenas prácticas de desarrollo y seguridad
- Documentar adecuadamente el trabajo realizado

La plataforma Hotel IDS proporciona una solución robusta y escalable para la gestión de reservas de hospedaje, con potencial para ser expandida y mejorada en el futuro. El código está bien estructurado, documentado y listo para ser mantenido y mejorado por otros desarrolladores.

13. Referencias

- Flask Documentation: https://flask.palletsprojects.com/
- PostgreSQL Documentation: https://www.postgresql.org/docs/
- Bootstrap: https://getbootstrap.com/
- Leaflet.js: https://leafletjs.com/
- OpenAPI Specification: https://spec.openapis.org/