POLIDEPORTIVO TORRELODONES

Especificación de Proyecto
Aplicación Web de Reservas

Versión: 1.0

Fecha: Diciembre 2025

Autor: Alejandro Diaz Bravo

Índice

1. Descripción general del proyecto

El objetivo de este proyecto es desarrollar una aplicación web sencilla que permita gestionar las reservas de diferentes instalaciones deportivas de un polideportivo ficticio. La aplicación estará centrada en ofrecer una experiencia clara y directa, sin necesidad de registro o autenticación. El usuario simplemente accede a la web, consulta la disponibilidad de las instalaciones para los próximos 30 días, selecciona un horario libre y formaliza la reserva mediante un pequeño formulario.

El sistema mostrará de manera visual y clara los huecos disponibles y ocupados para cada tipo de instalación, con una tarifa fija de 20 euros por hora. Al completar una reserva, el sistema actualizará automáticamente la disponibilidad, mostrando el hueco como ocupado.

2. Objetivo principal

Proporcionar una herramienta web funcional y sencilla que permita gestionar reservas deportivas de manera intuitiva, mostrando disponibilidad real en un calendario básico de 30 días y registrando reservas sin necesidad de inicio de sesión.

3. Usuarios previstos

La aplicación está dirigida a:

Personas usuarias del polideportivo ficticio que quieran reservar una pista.

Personal administrativo que quiera visualizar reservas.

Estudiantes del curso que utilizan esta aplicación como práctica académica y ejemplo de interacción con IA.

No será necesario crear distintos roles ni autenticación.

4. Funcionalidades esenciales (MVP)

4.1. Selección del tipo de instalación

El usuario podrá elegir entre:

Campo de fútbol

Pista de tenis

Pista de pádel

Pista de baloncesto

4.2. Consulta de disponibilidad

La aplicación mostrará los huecos disponibles y ocupados durante los próximos 30 días.

Los huecos estarán divididos por horas (por ejemplo: 09:00–10:00, 10:00–11:00, etc.).

Se mostrarán claramente ocupados (rojo) o disponibles (verde).
(El estilo será básico, delegando en Bootstrap si es requerido por la IA.)

4.3. Realización de una reserva

El usuario podrá:

Seleccionar un hueco disponible.

Rellenar un formulario con:

Nombre

Correo electrónico

Confirmar la reserva.

Ver cómo el hueco cambia automáticamente a "ocupado".

4.4. Actualización automática del calendario

Tras la reserva, la disponibilidad se actualiza en tiempo real.

No habrá cancelaciones ni ediciones ni registros de usuarios para mantener la simplicidad.

5. Requisitos funcionales

RF1 – Gestión de instalaciones

La aplicación permitirá seleccionar diferentes tipos de instalaciones disponibles.

RF2 – Visualización de disponibilidad

Mostrar para cada día y hora si el hueco está libre u ocupado.

RF3 – Registrar reserva

El sistema permitirá rellenar un formulario simple para completar una reserva.

RF4 – Almacenar y recuperar datos de reservas

Las reservas se almacenarán de manera persistente.

6. Requisitos no funcionales

RNF1 – Simplicidad

La interfaz debe ser clara y minimalista.

RNF2 – Acceso local

La aplicación funcionará exclusivamente en entorno local para evitar problemas de despliegue.

RNF3 – Robustez

El sistema debe evitar errores comunes generados por entradas incorrectas.

RNF4 – Rendimiento

La aplicación debe mostrar la disponibilidad de 30 días de forma fluida.

RNF5 – Trazabilidad del desarrollo

Todas las modificaciones del código deben ser generadas mediante asistencia de IA (no programación manual).

7. Alcance del proyecto

El proyecto comprende:

Una interfaz web sencilla con routes básicas de Flask.

Un sistema de almacenamiento local (SQLite) gestionado automáticamente por la IA.

La lógica de reservas sin autenticación.

Generación de todo el código mediante interacción con un asistente IA, incluyendo correcciones y mejoras.

Documentación del proceso y presentación de capturas.

No incluye:

Autenticación de usuarios.

Gestión de pagos o integración con pasarelas.

Edición o cancelación de reservas.

Panel de administración.

Despliegue en la nube.

8. Tecnologías previstas

Lenguaje: Python

Framework web: Flask

Base de datos: SQLite

Frontend: HTML + CSS básico (opcional Bootstrap)

Ejecución: Localhost

9. Dependencias previstas

Estas serán generadas por IA, pero previsiblemente incluirán:

Flask

SQLite (módulo estándar: sqlite3)

Posteriormente se generará un archivo requirements.txt automáticamente con la IA.

10. Limitaciones previstas

El sistema no valida solapamientos complejos: una reserva ocupa exactamente un hueco de una hora.

No tiene soporte multiusuario simultáneo avanzado (suficiente para una práctica local).

La apariencia visual será sencilla, priorizando funcionalidad sobre diseño.

Si la base de datos se borra o se reinicia, las reservas desaparecerán (propio de SQLite local).

11. Plan de trabajo propuesto

Definir prompts iniciales para la IA.

Solicitar estructura base del proyecto en Flask.

Pedir generación de modelos y base de datos.

Pedir rutas de:

Selección de instalación

Visualización de disponibilidad

Formulario de reserva

Iterar errores y corregirlos mediante IA.

Testear el flujo completo.

Documentar la interacción con el asistente IA.

Capturar pantallas para el informe final.

Preparar PDF final.

Ajustes aplicados: Requisitos y especificaciones

1. Requisitos Funcionales

RF1: Registro e inicio de sesión de usuarios con email y contraseña.

RF2: Gestión de perfiles de usuario (ver/editar datos).

RF3: Listado de instalaciones con detalle (nombre, descripción, capacidad, precio).

RF4: Crear, modificar y anular reservas por parte del usuario.

RF5: Gestión de instalaciones por administradores (alta, baja, edición).

RF6: Visualización de reservas del usuario y estado (confirmada/cancelada).

2. Requisitos No Funcionales

RNF1: La aplicación deberá ejecutarse localmente y responder en < 500ms para operaciones comunes.

RNF2: Persistencia en base de datos SQLite para el MVP.

RNF3: Autenticación segura usando JWT y contraseñas cifradas.

RNF4: Interfaz usable con accesibilidad básica y responsive para móvil.

RNF5: Documentación mínima para despliegue local y pruebas.

3. Historias de Usuario (priorizadas)

HU1: Como usuario, quiero registrarme con email y contraseña para poder reservar instalaciones.

HU2: Como usuario registrado, quiero ver el listado de instalaciones y sus detalles para elegir una.

HU3: Como usuario, quiero reservar una instalación indicando fecha/hora para asegurarla.

HU4: Como administrador, quiero dar de alta/baja instalaciones para mantener catálogo actualizado.

HU5: Como usuario, quiero ver y cancelar mis reservas para gestionarlas fácilmente.

4. Criterios de Aceptación (ejemplos)

CA1: Registro: al enviar email/username/password válidos, se crea usuario y se devuelve 201.

CA2: Reserva: no se deben crear reservas con solapamiento para la misma instalación (comprobación en backend).

CA3: Seguridad: las contraseñas se almacenan hashed y el token JWT expira tras un periodo configurado.

5. Modelo de Datos (resumen)

Entidades principales:

- Usuario: id, username, email, password_hash, role
- Instalación (Facility): id, name, description, capacity, price
- Reserva: id, user_id, facility_id, start, end, status

6. Autenticación y Autorización (decisión)

Se adopta autenticación con JWT y almacenamiento de contraseñas con bcrypt. Roles: admin/usuario.

7. Entregables y MVP

- API REST local con endpoints de autenticación, instalaciones y reservas.
- Frontend mínimo para registro/login, listado y creación de reservas.
- Documentación de despliegue local y criterios de aceptación completos.