Tarea

Hugo Javier Ledesma

Institución universitaria Antonio José Camacho

Ingeniería en sistemas

2025

# **Instalación y ejecutar la aplicación**

* **Instalación y configuración del frontend (React + Vite)**

**1.1 Crear el proyecto con Vite**

Ejecuta el siguiente comando en la terminal para crear un nuevo proyecto con Vite:

npm create vite@latest loscielos --template react

Esto creará la carpeta **loscielos** con la estructura básica de un proyecto React con Vite.

**1.2 Instalar dependencias**

Accede a la carpeta del proyecto y ejecuta:

cd loscielos

npm install

Luego, instala las dependencias necesarias:

npm install @mui/material @mui/icons-material @reduxjs/toolkit react-redux react-router-dom react-toastify formik yup styled-components

**1.3 Ejecutar el frontend**

Para iniciar el servidor de desarrollo de Vite, usa:

npm run dev

Esto abrirá la aplicación en el navegador en http://localhost:5173.

# Configuración de proyecto

* **Instalación y configuración del backend (Express + MySQL)**

**2.1 Crear la carpeta del backend**

Desde la raíz del proyecto, crea una carpeta para el backend y accede a ella:

cd services

**2.2 Inicializar Node.js y Express**

Ejecuta el siguiente comando para crear un archivo package.json:

npm init -y

Luego, instala Express, CORS y MySQL:

npm install express cors mysql2

**2.3 Crear el servidor (server.js)**

Crea un archivo llamado server.js y copia el siguiente código:

import express from "express";

import cors from "cors"; // Importa cors

import mysql from "mysql2";

import parapenteData from "./parapente.json" assert { type: "json" };

const app = express();

const port = 3000;

// Configuración de la base de datos MySQL

const db = mysql.createConnection({

  host: "localhost",

  user: "root",

  password: "",

  database: "parapentebd",

});

// Conexión a la base de datos

db.connect((err) => {

  if (err) {

    console.error("Error de conexión a la base de datos:", err);

    return;

  }

  console.log("Conectado a la base de datos MySQL");

});

// Habilitar CORS

app.use(cors());

// Middleware para parsear el cuerpo de las solicitudes como JSON

app.use(express.json());

// Ruta raíz para verificar que el servidor funciona

app.get("/", (req, res) => {

  res.send("Bienvenido a la API de Parapente");

});

// Iniciar el servidor

app.listen(port, () => {

  console.log(`API corriendo en http://localhost:${port}`);

});

**2.4 Configurar la base de datos en MySQL**

Abre MySQL y ejecuta los siguientes comandos para crear la base de datos y la tabla de usuarios:

CREATE TABLE `credenciales` (

  `rol` varchar(30) NOT NULL,

  `email` text NOT NULL,

  `password` varchar(50) NOT NULL,

  `id` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

**2.5 Ejecutar el backend**

Desde la carpeta backend, inicia el servidor:

node server.js

Si deseas que se reinicie automáticamente al hacer cambios, instala nodemon:

npm install -g nodemon

Y luego ejecuta:

nodemon server.js

# Descripción de la estructura del proyecto y decisiones técnicas

El proyecto está organizado de manera modular para facilitar su mantenimiento y escalabilidad. Se trata de una aplicación web desarrollada con React y Vite en el frontend, mientras que en el backend se utiliza Node.js con Express para la API y MySQL como base de datos para la autenticación de usuarios. Además, se emplea un archivo JSON para almacenar información sobre los tipos de parapente disponibles.

**Estructura del Proyecto**

* src/assets/Files: Contiene recursos estáticos como imágenes, iconos y fuentes.
* src/assets /components: Almacena componentes reutilizables como la barra de navegación (AppBar.jsx), el pie de página (Footer.jsx), el encabezado (Header.jsx) y las tarjetas de servicios (ServiciosCard.jsx).
* src/assets /layouts: Define la estructura de las diferentes páginas o secciones de la aplicación (Admin.jsx, Instructor.jsx, Landing.jsx, Login.jsx, Servicios.jsx).
* src/assets /routes: Contiene la configuración de las rutas principales (App.jsx) y un archivo imagenes.js que probablemente maneja imágenes dinámicamente.
* src/assets /services: Aquí se gestiona la lógica del backend, incluyendo:
* parapente.json: Archivo que almacena información sobre los diferentes tipos de parapente.
* parapentebd.sql: Script para la creación de la base de datos en MySQL.
* server.js: Archivo donde se configura el servidor Express y Endpoint de SQL y se definen las rutas.
* src/style: Contiene archivos relacionados con el estilo de la aplicación.

**Otros archivos importantes:**

* main.jsx: Punto de entrada del frontend en React.
* vite.config.js: Configuración de Vite para optimizar el desarrollo y la compilación.
* .gitignore: Para evitar la inclusión de archivos innecesarios en el control de versiones.

# **Rutas CRUD para la API**

# En el archivo server.js, se definen las rutas para interactuar con la base de datos y el archivo JSON. Estas son:

# 1. Gestión de usuarios (credenciales) con MySQL

# Base de datos: parapentebd

# Tabla: credenciales

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Método** | | |  | | --- | | **Ruta** | | |  | | --- | | **Descripción** | |
| POST | /api/register | Registra un nuevo usuario. |
| POST | /api/login | Autentica a un usuario. |

**2. Gestión de tipos de parapente con JSON**

Archivo: parapente.json

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | Método | | |  | | --- | | Ruta | | |  | | --- | | Descripción | |
| GET | /api/parapentes | Obtiene la lista de tipos de parapente. |
| POST | /api/parapentes | Agrega un nuevo tipo de parapente. |
| PUT | /api/parapentes/:id | Modifica un tipo de parapente existente. |
| DELETE | /api/parapentes/:id | Elimina un tipo de parapente. |

# **Decisiones Técnicas**

# Uso de MySQL para credenciales

# Se utilizó MySQL para almacenar credenciales de usuario debido a la necesidad de contar con una base de datos relacional segura y confiable para la autenticación.

# Se permite la gestión de usuarios con contraseñas cifradas, lo que mejora la seguridad.

# Facilita la escalabilidad al integrar más información de usuarios, como roles y permisos.

# Uso de JSON para los tipos de parapente

# El archivo parapente.json almacena información estática sobre los tipos de parapente, ya que estos datos no cambian con frecuencia.

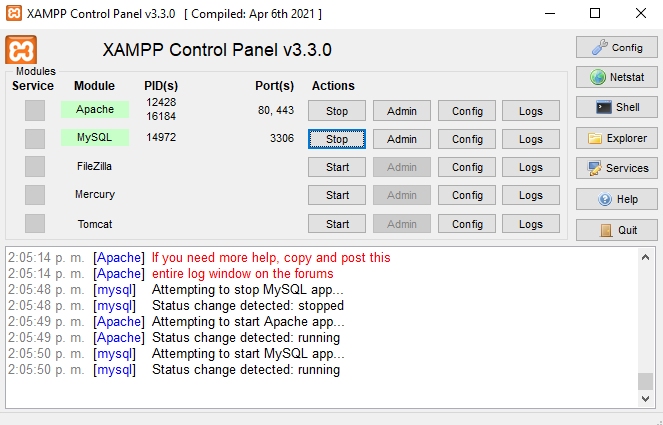
# Se eligió JSON porque es ligero y rápido de leer/escribir, evitando la sobrecarga de consultas a una base de datos para datos poco dinámicos.

# Esto permite que el frontend pueda acceder a los datos sin necesidad de una consulta SQL, optimizando el rendimiento.

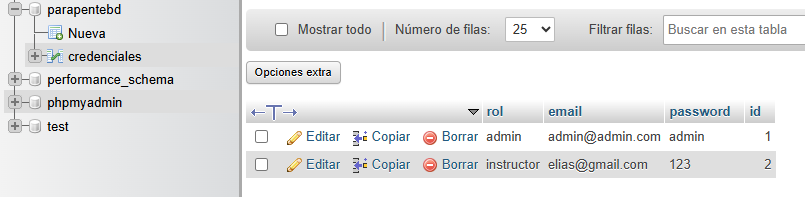
# Evidencias de las pruebas al Backend y al Frontend

# **Backend**

Xamp:

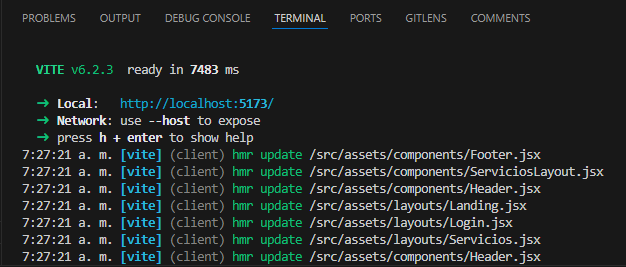


Database:

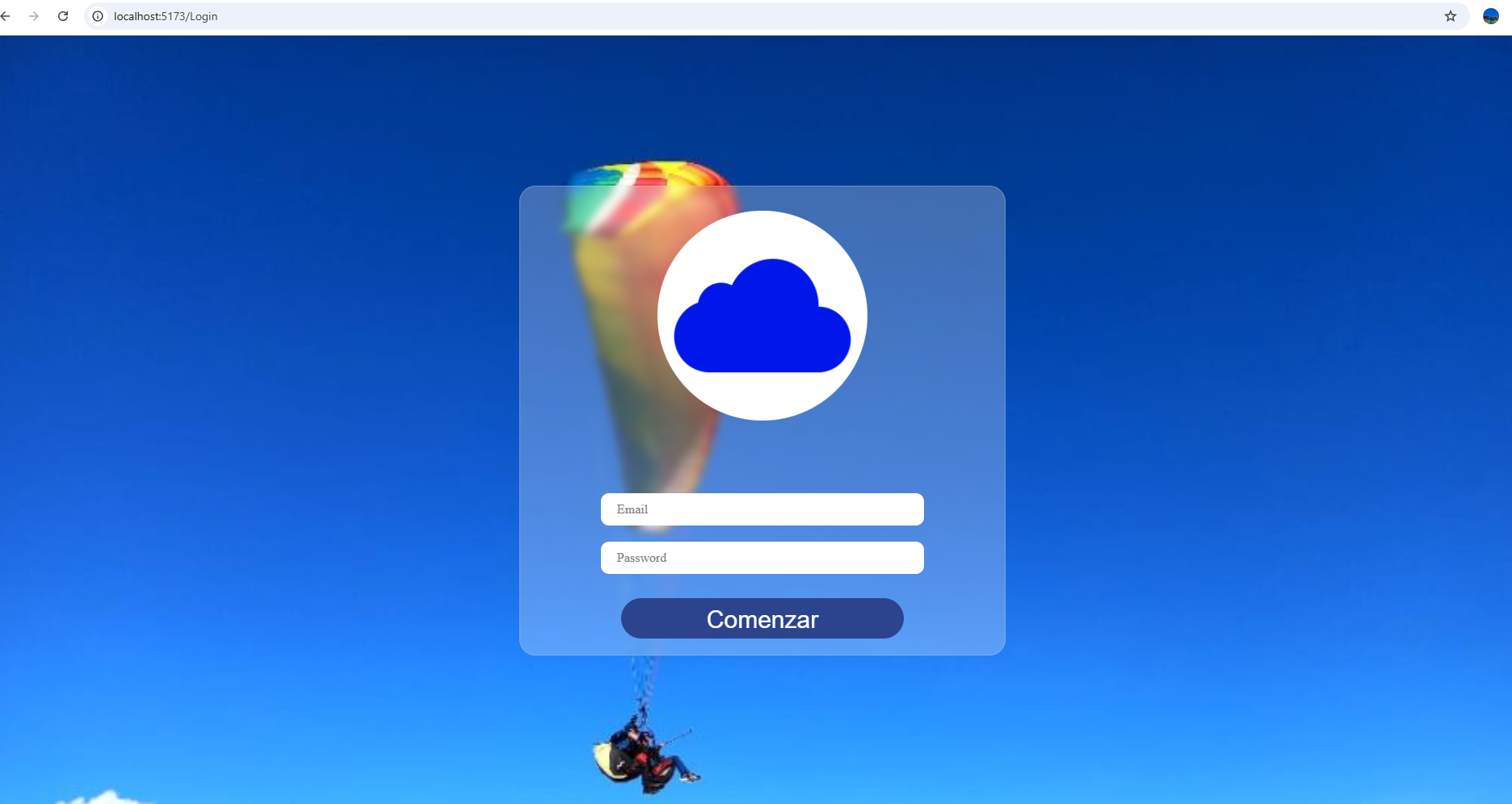


# **Frontend:**

Servidor Front:



Login:



Landing:

