**Documentación del Proyecto TP\_1**

**Sistema para Gestión de Inscripciones**

**TUP - UTN**

**Materia: Programación 4.**

**Grupo 6: Integrantes:**

**Vocos, Alfredo**

**Gonzales, Juan**

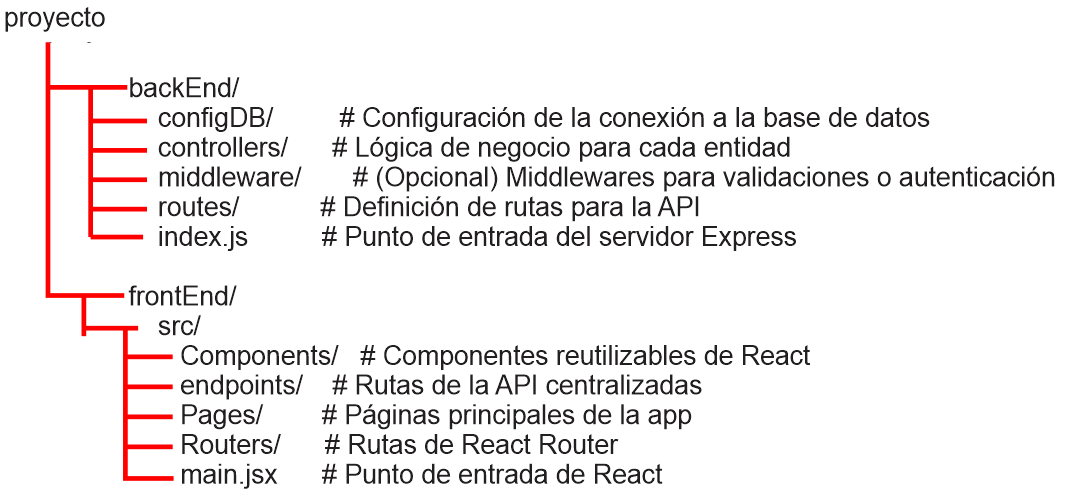
**Ortiz, Martín**

**Ponce, Cristian**

**Descripción**

Este proyecto es una aplicación web para la gestión de cursos, estudiantes e inscripciones. Utiliza un backend en Node.js con Express y MySQL, y un frontend en React. Permite visualizar, crear, editar y eliminar cursos, estudiantes e inscripciones.

**Estructura de árbol del Proyecto**



**Proceso de Creación**

**Primero inicializamos en la terminal y posicionados en la carpeta raiz, ejecutamos git init**

**Creamos 2 carpetas: Front y Back**

**1. Inicialización del Backend**

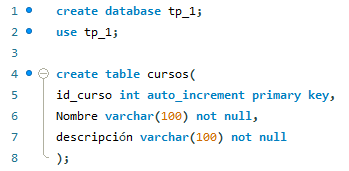
* Se creó la carpeta [backEnd](vscode-file://vscode-app/c:/Users/alfe-/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)
* Entramos a la carpeta [backEnd](vscode-file://vscode-app/c:/Users/alfe-/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

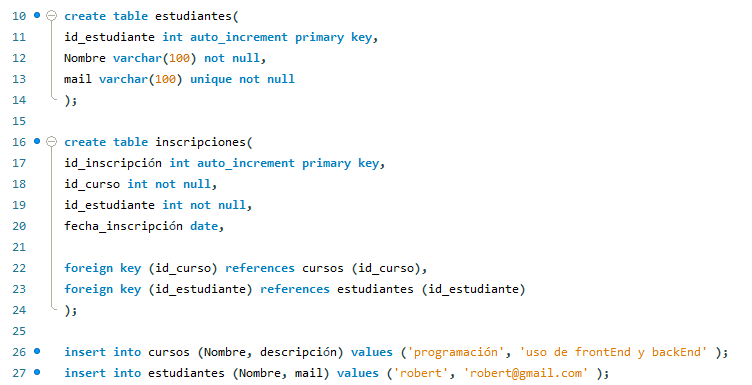
cd backEnd

* y se inicializó con npm init -y.
* Se instalaron las dependencias principales:
* npm install express
* npm i mysql2 cors nodemon
* npm i cors nodemon
* npm i nodemon
* Se creó la estructura de carpetas para separar la lógica:
  + ConfigDB: para la configuración de la base de datos.
  + Controllers: para la lógica de cada entidad (cursos, estudiantes, inscripciones).
  + Routes: para definir las rutas de la API.
  + index.js: configura el servidor Express y conecta las rutas y arranca el servidor.
* El backend se encuentra disponible en el puerto http://localhost:8000.

**2. Conexión a la Base de Datos MySQL (Workbench)**

* En configDB/dataBase.js se configura la conexión a MySQL usando los datos del servidor local.
* en nuestra bases de datos, vamos a crear una BD.





**3. Inicialización del Frontend**

* Se creó la carpeta [frontEnd](vscode-file://vscode-app/c:/Users/alfe-/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) y se inicializó con Vite
* cd frontEnd
* npm create vite@latest
* Se instalaron las dependencias necesarias:
* npm i axios
* npm i react-bootstrap
* npm i react-router-dom
* El frontend se encuentra disponible en http://localhost:5173.

**Explicación de Carpetas y Archivos**

* **Components**: Componentes reutilizables de React (tablas, formularios, etc).
* **Pages**: Páginas principales de la app (por ejemplo, Home).
* **endpoints**: Centraliza las URLs de la API para fácil mantenimiento.
* **Routers**: Configuración de rutas de React Router.

**Fragmentos de códigos importantes**

1. Conexión a la base de datos MySQL
2. const mysql = require('mysql2');
3. const connection = mysql.createConnection({
4. host: 'localhost',
5. user: 'root',
6. password: 'tu\_contraseña',
7. database: 'nombre\_base\_de\_datos'
8. });
9. module.exports = { connection };

**Explicación:**  
Este archivo guarda toda la configuración para conectarse a la base de datos MySQL (usuario, contraseña, nombre de la base, etc.).

En lugar de escribir esa configuración en cada parte del proyecto, se hace una sola vez ahí.  
Después, cuando otro archivo necesita consultar o modificar datos en la base, importa esa conexión y la usa directamente.

**2. Controladores (Ejemplo: Cursos)**

const mostrarCursos = (req, res) => {

  connection.query('SELECT \* FROM cursos', (error, results) => {

    if (error) { /\* Manejo de error \*/ }

    res.json(results);

  });

};

**Explicación:**  
Los controladores contienen la lógica para interactuar con la base de datos. Por ejemplo, mostrarCursos obtiene todos los cursos y los devuelve como JSON.

**3. Rutas (Ejemplo: Cursos)**

const express = require('express');

const router = express.Router();

const { mostrarCursos, crearCurso, editarCurso, eliminarCurso } = require('../controllers/cursos');

router.get('/cursos', mostrarCursos);

router.post('/cursos', crearCurso);

router.put('/cursos/:id', editarCurso);

router.delete('/cursos/:id', eliminarCurso);

module.exports = router;

**Explicación:**  
Las rutas definen los endpoints de la API y conectan cada ruta con su controlador correspondiente.

**4. Punto de entrada del backend**

const express = require("express");

const cors = require('cors');

const cursos = require("./routes/cursos");

const estudiantes = require("./routes/estudiantes");

const inscripciones = require("./routes/inscripciones");

const app = express();

app.use(express.json());

app.use(cors());

app.use("/", cursos);

app.use("/", estudiantes);

app.use("/", inscripciones);

app.listen(8000, () => {

  console.log("Servidor corriendo en el puerto 8000");

});

**Explicación:**  
Aquí se configura Express, se importan las rutas. El servidor escucha en el puerto 8000.

**5. API en el Frontend**

useEffect(() => {

  getCursos();

  getEstudiantes();

  getInscripciones();

}, []);

**Explicación:**  
Cuando el componente se monta, se hacen peticiones a la API para obtener los datos y mostrarlos en tablas.

**6. Renderizado de tablas con datos**

<Table>

  <thead>...</thead>

  <tbody>

    {cursos.map((curso) => (

      <tr key={curso.id\_curso}>

        <td>{curso.id\_curso}</td>

        <td>{curso.Nombre}</td>

        <td>{curso.descripción}</td>

      </tr>

    ))}

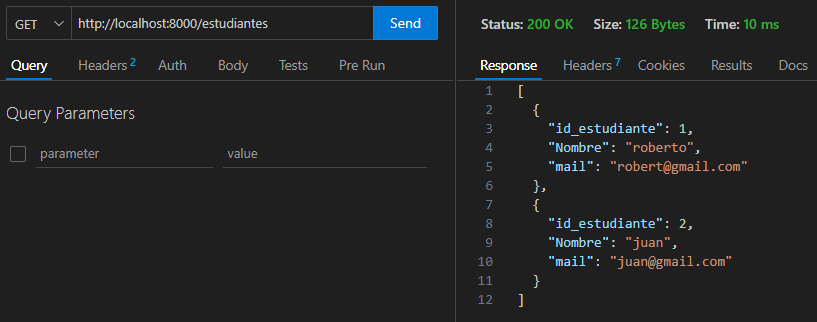
  </tbody>

</Table>

**Explicación:**  
Se recorre el array de cursos y se renderiza una fila por cada curso, usando el id como key única.

**7. Pruebas con Thunder Client**

* Se utilizó la extensión Thunder Client en VS Code para probar los endpoints del backend.
* Se probaron los métodos GET, POST, PUT y DELETE para cursos, estudiantes e inscripciones.
* Ejemplo de prueba para mostrar los estudiantes:
  + Método: GET
  + URL: http://localhost:8000/estudiantes



**8. abrimos el navegador con**

Cd frontEnd

npm run dev

**9 . subimos el proyecto a GITHUB**

* git push -u origin main
* Hacemos commits frecuentemente para registrar el avance.