Guia basica para backend

Samuel Benitez Mesa

samuelbenitezmesa@gmail.com

INTRODUCCIÓN

Esta guía tiene como objetivo dar paso a paso para crear la base funcional del Backend para una página web, utilizando los controles CRUD para la obtención, implementación, actualización y eliminación de datos dentro de una base datos, comunicándose entre sí de forma consistente

DETALLE DEL INFORME

Dividiré las secciones por medio de pasos de inicio a fin, junto a pruebas de funcionamiento para asegurar el funcionamiento del código

TABLA DE ILUSTRACIONES

[Ilustración 1 Carpetas Backend y Frontend 2](#_Toc206373414)

[Ilustración 2 Creación de archivo HTML y JS 3](#_Toc206373415)

[Ilustración 3 estructura HTML5 mas identificación del tema 3](#_Toc206373416)

[Ilustración 4 Formulario para el usuario 3](#_Toc206373417)

[Ilustración 5 Uso de Live Server del proceso 4](#_Toc206373418)

[Ilustración 6 Sección para los datos y anclar JS 4](#_Toc206373419)

[Ilustración 7 Anclar URL API 5](#_Toc206373420)

[Ilustración 8 Funciones CRUD 5](#_Toc206373421)

[Ilustración 9 Referencias DOM 6](#_Toc206373422)

[Ilustración 10 Función para mostrar los equipos 6](#_Toc206373423)

[Ilustración 11 Funciones para añadir y cancelar los nuevos datos 7](#_Toc206373424)

[Ilustración 12 Funciones para editar y eliminar los nuevos datos 7](#_Toc206373425)

[Ilustración 13 Carpetas src en Backend 8](#_Toc206373426)

[Ilustración 14 archivos externos del src 8](#_Toc206373427)

[Ilustración 15 carpeta Backend sin ubicar 8](#_Toc206373428)

[Ilustración 16Carpeta Backend ubicada 9](#_Toc206373429)

[Ilustración 17 package.json ingresado con éxito 9](#_Toc206373430)

[Ilustración 18 ingresar función dev en package.json 9](#_Toc206373431)

[Ilustración 19 nodemon instalado exitosamente 10](#_Toc206373432)

[Ilustración 20 carpeta node\_modules instalado correctamente 10](#_Toc206373433)

[Ilustración 21 nodemon instalado en devDependencias instalado correctamente 10](#_Toc206373434)

[Ilustración 22 modulos instalados correctamente 10](#_Toc206373435)

[Ilustración 23 dependencias instalas correctamente en el package.json 11](#_Toc206373436)

[Ilustración 24 Implementación de variables de entorno 11](#_Toc206373437)

[Ilustración 25 archivo conexión de la base de datos para usar las variables de entorno 11](#_Toc206373438)

[Ilustración 26 archivo uploads 12](#_Toc206373439)

[Ilustración 27 Código gitignore 12](#_Toc206373440)

[Ilustración 28 Implementación del código CRUD 12](#_Toc206373441)

[Ilustración 29 CRUD crear y actualizar 13](#_Toc206373442)

[Ilustración 30 CRUD eliminar y exportación de la clase CrudController 13](#_Toc206373443)

[Ilustración 31 Rutas para la base de datos y el CRUD obtener (GET) 14](#_Toc206373444)

[Ilustración 32 Rutas (POST) (PUT) (DELETE) 14](#_Toc206373445)

[Ilustración 33 Código para app.js 15](#_Toc206373446)

[Ilustración 34 Código para Server.js 15](#_Toc206373447)

[Ilustración 35 Implementar archivo BD.sql y añadir base de datos, tablas y datos 15](#_Toc206373448)

[Ilustración 36 Crear servidor MySQL 16](#_Toc206373449)

[Ilustración 37 Creación de bd, tabla y datos en MySQL Workbench 16](#_Toc206373450)

[Ilustración 38 Mensaje terminal, servidor corriendo exitosamente 17](#_Toc206373451)

[Ilustración 39 Extensión Thunder Client 17](#_Toc206373452)

[Ilustración 40 Extensión Thunder Cliente ingresar prueba de BD 17](#_Toc206373453)

[Ilustración 41 Ingresar URL de nuestro servidor local 18](#_Toc206373454)

[Ilustración 42 Funcionamiento GET 18](#_Toc206373455)

[Ilustración 43 Funcionamiento POST 19](#_Toc206373456)

[Ilustración 44 Funcionamiento PUT 19](#_Toc206373457)

[Ilustración 45 Funcionamiento DELETE 19](#_Toc206373458)

[Ilustración 46 Extensión Live Server 20](#_Toc206373459)

[Ilustración 47 Prueba con Live Server GET 20](#_Toc206373460)

[Ilustración 48 Prueba POST 20](#_Toc206373461)

[Ilustración 49 Prueba PUT 21](#_Toc206373462)

[Ilustración 50 Prueba DELETE 21](#_Toc206373463)

[Ilustración 51Repositorio GitHub creado 22](#_Toc206373464)

[Ilustración 52 Usar Git bash en la carpeta contenedora 22](#_Toc206373465)

[Ilustración 53 proceso para subir proyecto a repositorio GitHub 23](#_Toc206373466)

[Ilustración 54 uso de git status 24](#_Toc206373467)

[Ilustración 55 Proceso para actualizar proyecto en el repositorio GitHub 24](#_Toc206373468)

[Ilustración 56 Creación de archivo CSS 25](#_Toc206373469)

[Ilustración 57 Vincular archivo CSS a la cabeza del HTML 25](#_Toc206373470)

[Ilustración 58 Estilo Body 25](#_Toc206373471)

[Ilustración 59 Estilo Nav 25](#_Toc206373472)

[Ilustración 60 Estilos títulos 26](#_Toc206373473)

[Ilustración 61 Estilo Formulario 26](#_Toc206373474)

[Ilustración 62 Estilos para botones y cuadro de texto 26](#_Toc206373475)

[Ilustración 63 Estilo para los Cards 27](#_Toc206373476)

etapa frontend

1N. PASO (Creacion carpetas y archivos)

Empezamos por crear dos carpetas, lo más sencillo y obvio es crear una llamada **Backend** y **Frontend**

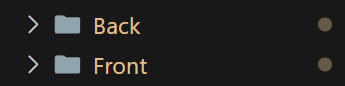


Ilustración 1 Carpetas Backend y Frontend

2n. PASO

Después dentro de la carpeta **Frontend** (Conocido por ser la parte interactuable por el usuario) crearemos dos archivos, ***index.html*** (el index es conocido por ser un nombre el cual el sistema reconoce de forma mas sencilla) y index.js, que será el funcionamiento para el usuario *(SUPER MEGA IMPORTANTE PARA ANCLAR LA CARPETA BACKEND Y LA BASE DE DATOS AL FRONTEND).*

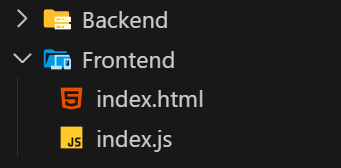
**

Ilustración 2 Creación de archivo HTML y JS

3n. paso html

Usaremos el simbolo **!** para hacer un **DOCTYPE** basico, podremos introducir el nombre de la pagina en la etiqueta t**itle**, la tematica sera sobre equipos de futbol (sera importante a la hora de crear la base de datos) asi que añadiremos un titulo con **nav**, **h1/h2** para el titulo y **p** para los parrafos.

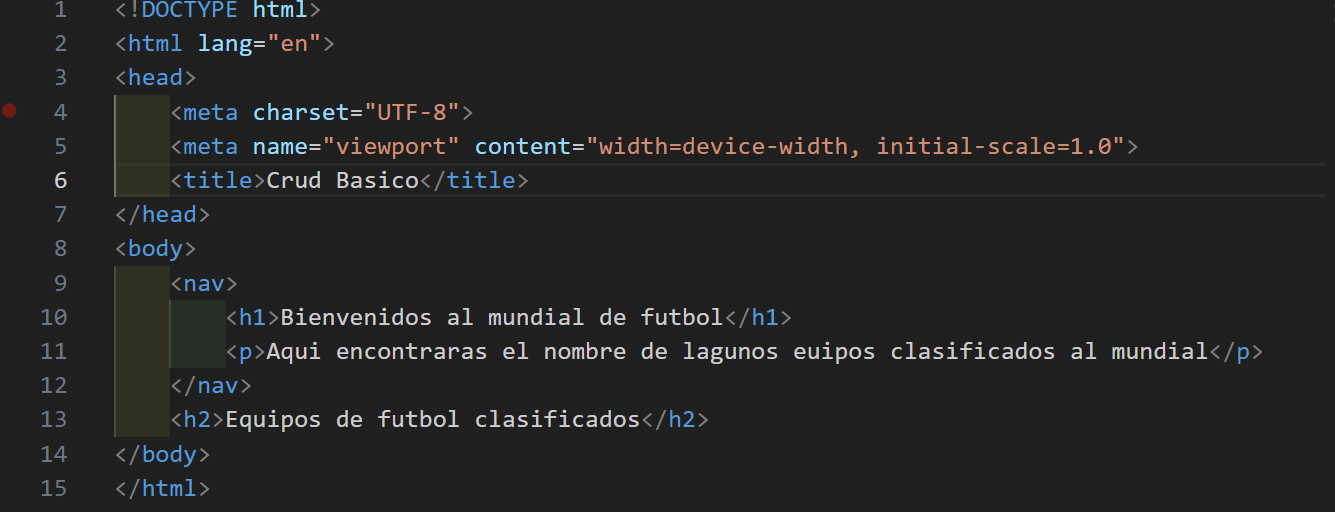


Ilustración 3 estructura HTML5 mas identificación del tema

4n. paso (interaccion para el usuario)

Añadiremos etiqueta **form** para hacer un formulario, junto a una ***id*** para identificarlo *(y para usar en funciones js)* dentro de estos tendrá un **input** con tipo **hidden** y también su propia id, después usaremos div (para dividir la sección) usaremos un **label** y un input tipo texto para que identifique e ingrese el equipo de futbol que desee.  
Junto a esto botones de guardar e ingresar, muy importante guardar ya que gracias a este se almacenara los nuevos datos.

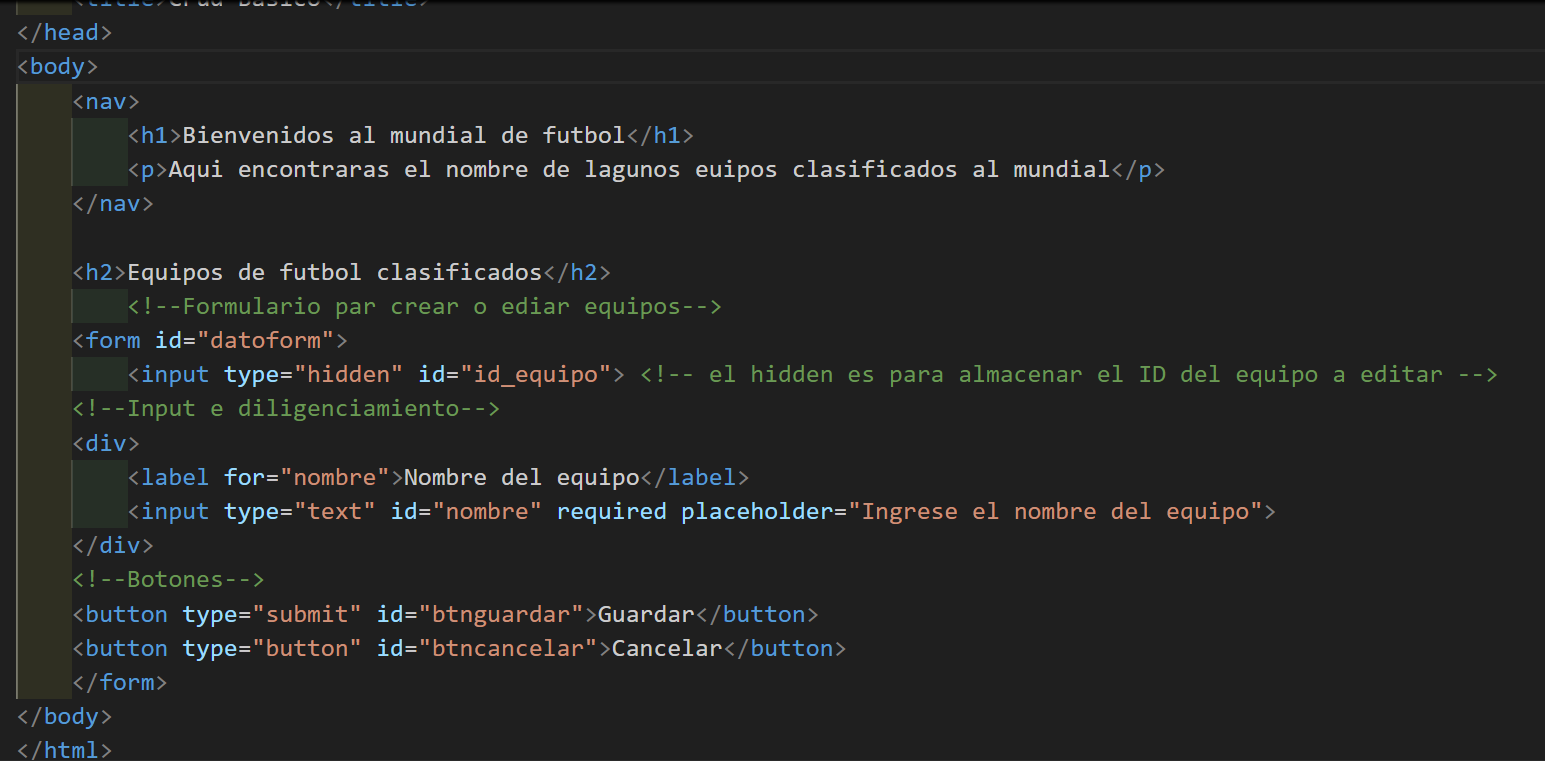


Ilustración 4 Formulario para el usuario



Ilustración 5 Uso de Live Server del proceso

5n. paso (formulario basico)

Después de esto le daremos el espacio la información que vendrá por parte del **Backend**, que sería **cards**, usaremos un **div** para identificarlo más un **témplate** que tendrán botones para eliminar y editar (Parte crucial para los controles **CRUD**, por ultimo por parte del HTML relacionamos un script el archivo js que creamos.

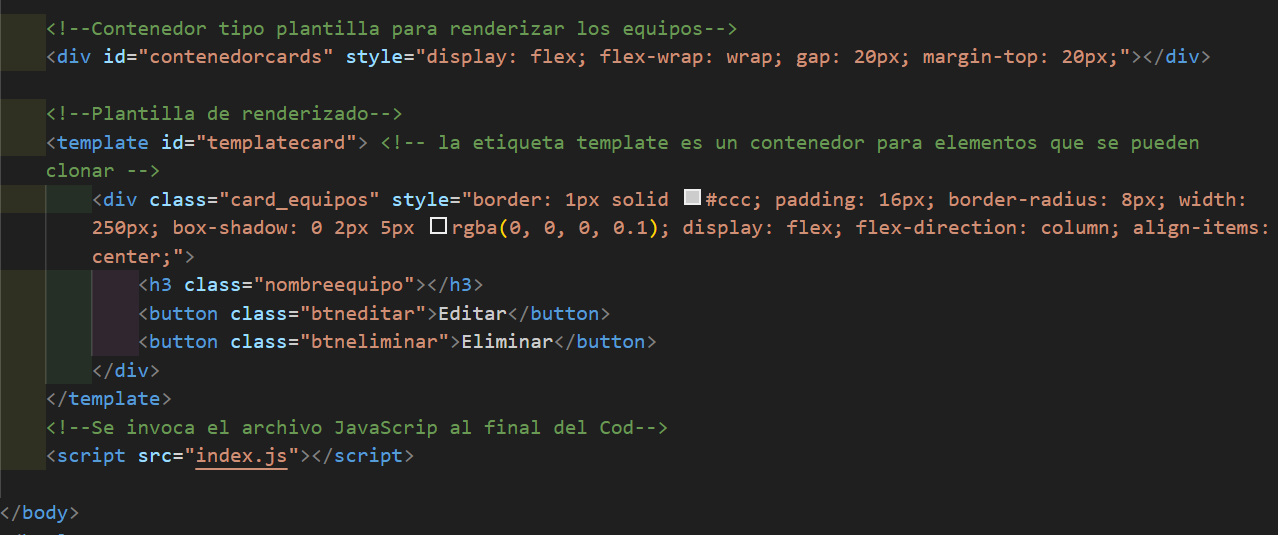


Ilustración 6 Sección para los datos y anclar JS

6n. paso javascript (funcionabilidad para el usuario y conexión con el backend)

a continuación, sigue ***JavaScript*** (**Muy importante** relacionar los **id** que creamos con anterioridad para que las funciones se activen correctamente), crearemos una variable cons con el nombre de ***API\_URL***, usaremos la siguiente **url *(***[***http://localhost:3000/api/equipos***](http://localhost:3000/api/equipos)***)*** esto es porque ya estamos relacionando el puerto y su api aunque no llegamos a la sección de Backend.

Después creamos las siguientes funciones conocidas como **CRUD**, *obtener, enviar, actualizar y eliminar.*

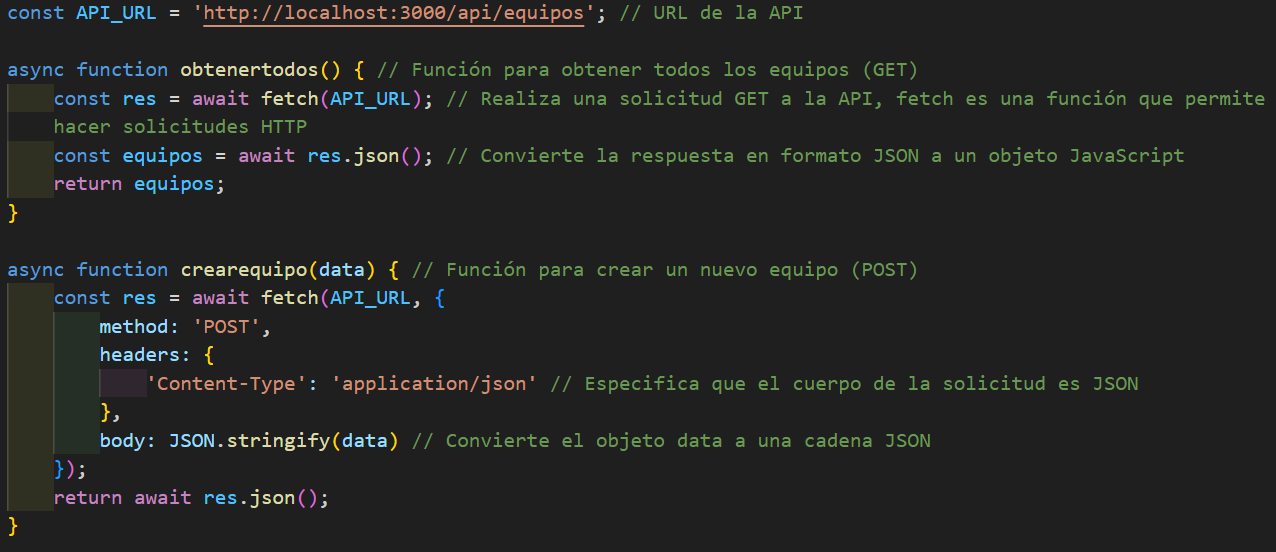


Ilustración 7 Anclar URL API

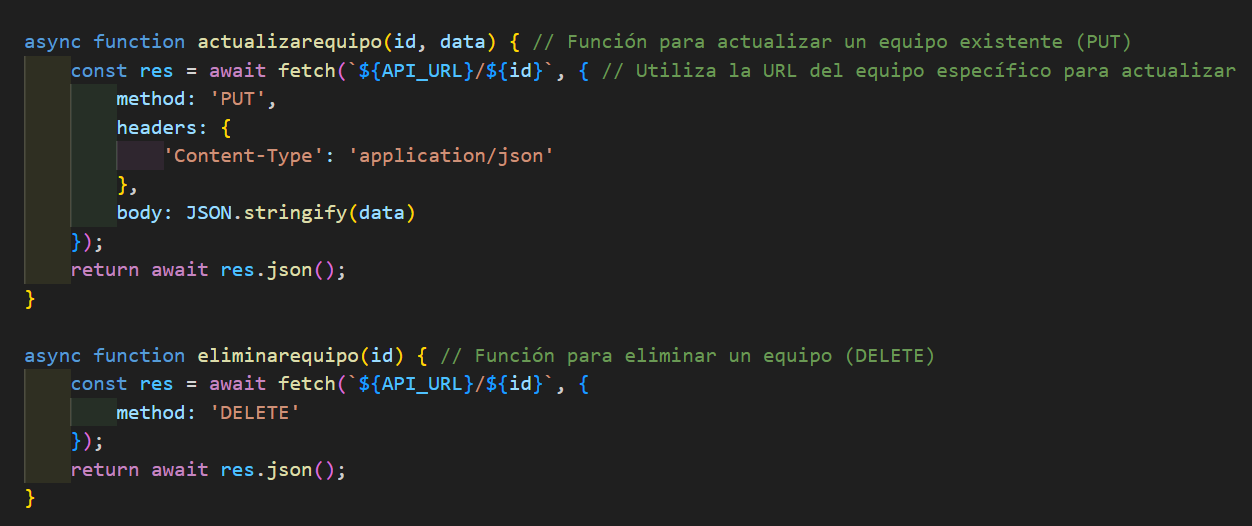


Ilustración 8 Funciones CRUD

6n paso (referencias doom)

Crearemos referencias ***DOM (Document Object Model)*** que en pocas palabras estamos trayendo los id de las etiquetas que añadimos con anterioridad para darle cuyo funcionamiento en las funciones **Javascript** y que así tengan un funcionamiento a la hora de interactuar en la página **(Importante ingresar los nombres exacto del HTML)**.

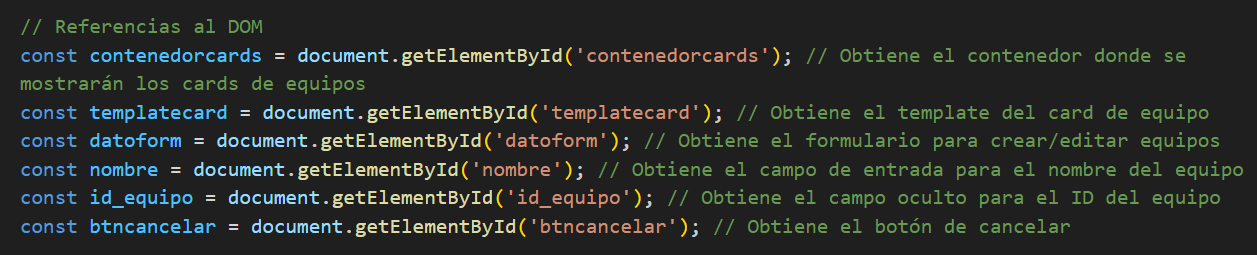


Ilustración 9 Referencias DOM

7n paso (Funciones asincronicas para cards)

Este paso es bastante importante, en resumen, la función nos dará la posibilidad de ver los datos de la base de datos que vendrán del **api** que añadimos al principio del **JS** el cuales llamado **mostrarequipos()**

La siguiente función se encargará de una parte bastante importante, después de esta será la que añada el nuevo dato que el usuario va a ingresar, esta ira para **dataform** como una función asincrónica ***(esta espera a que se complete una tarea para que funcione).***

Las demás le dan funcionamiento al botón cancelar, el botón editar y eliminar dentro de los cards, y por último y muy importante llamamos la función **mostrarequipos()** que creamos arriba para que siempre cargue los equipos que están añadidos y añadiremos.

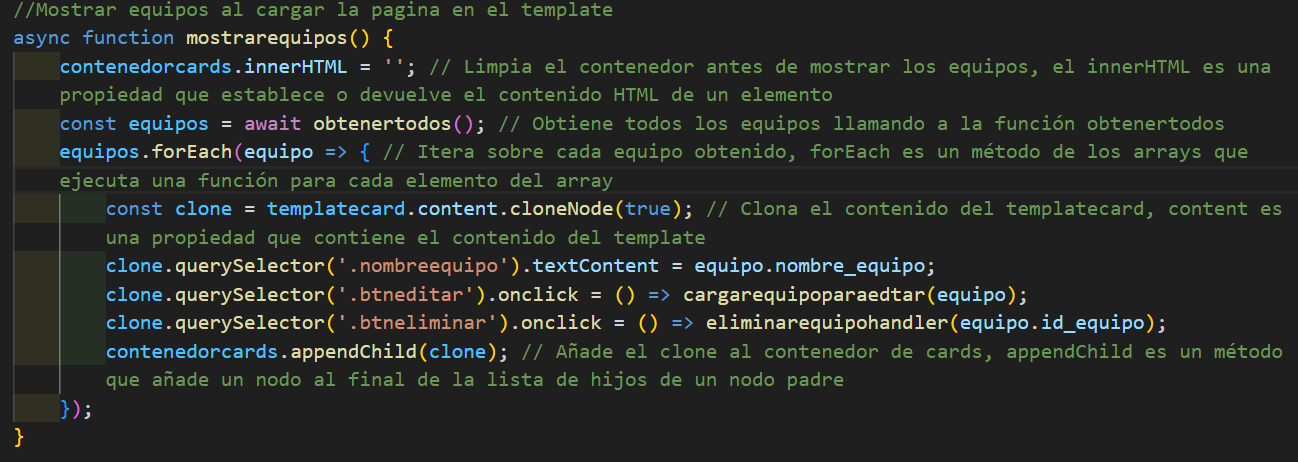


Ilustración 10 Función para mostrar los equipos

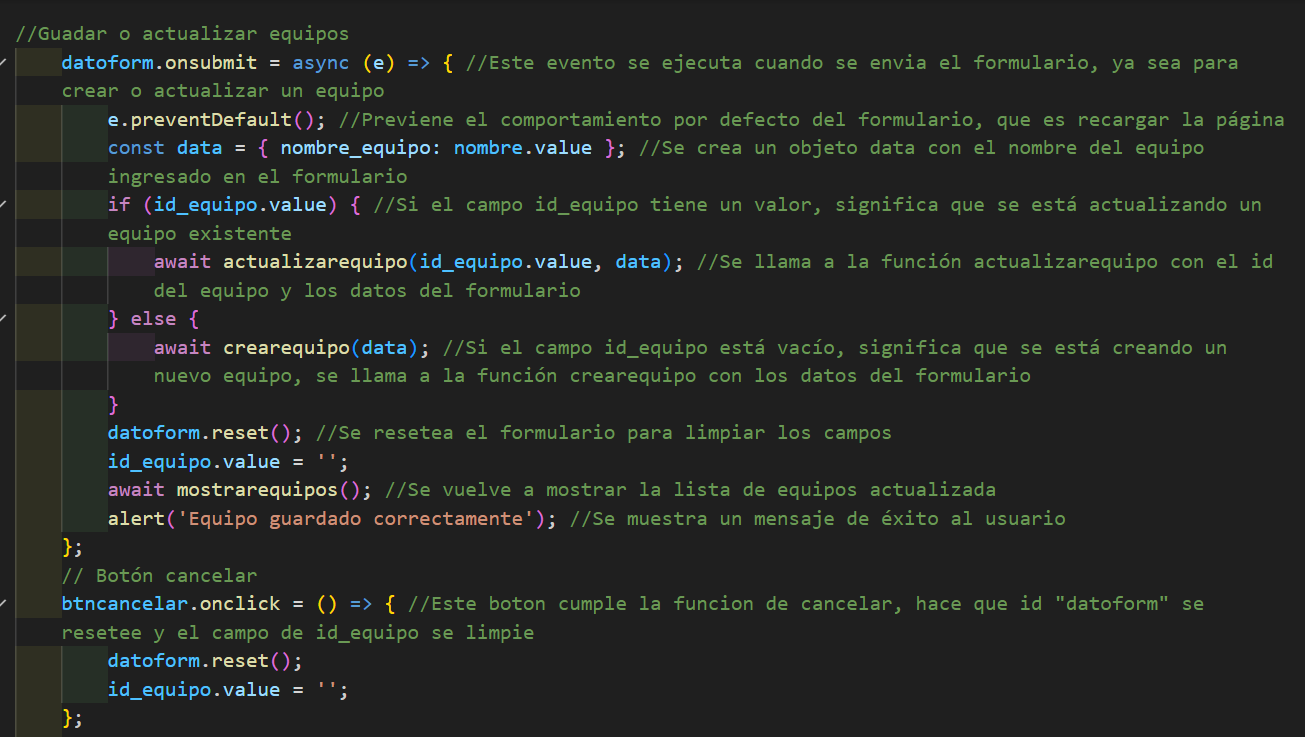


Ilustración 11 Funciones para añadir y cancelar los nuevos datos

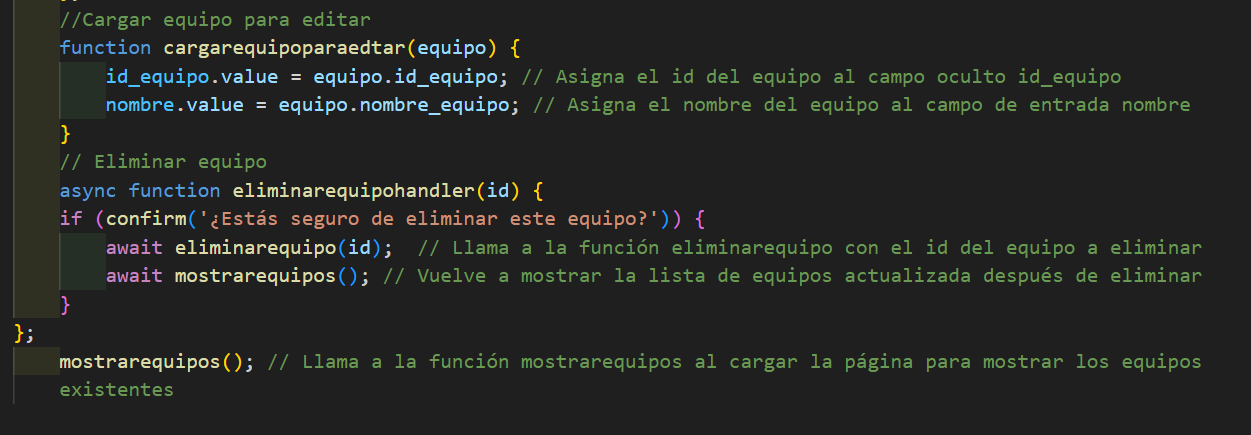


Ilustración 12 Funciones para editar y eliminar los nuevos datos

Etapa backend

1n. paso (carpetas backend)

Comenzaremos creando la carpeta principal llamada **src**, esta contendrá las siguientes carpetas.  
la primera se llama **config** su función será la configuración de la base de datos, la siguiente es **controllers** la cual nos ayudara a crear los controles **crud,** después sigue **db** *(data base)* donde estará nuestro código **Mysql** de preferencia, después la carpeta **middlewares** la cual funciona entre la petición del usuario y la respuesta del servidor, y por ultimo **routes** el cual funcionara para enlazar nuestros datos al **crud** que crearemos en *controllers,* y por ultimo añadiremos **app.js**que es el corazón de la aplicación y definirá las rutas y el middleware.*(asegurarse que todas las carpetas se encuentren dentro de* ***routes*** *al igual que la JS app).*

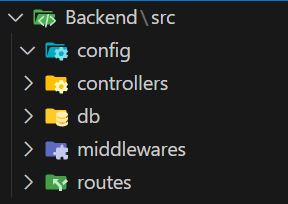


Ilustración 13 Carpetas src en Backend

2n. paso (archivos backend)

Fuera de la carpeta **src** crearemos el .***env*** el cual contendrá variables de entorno donde se encontrará información sensible.

Después creamos el archivo **,gitignore** (este tendrá un código que eliminará módulos que instalaremos para server).

Creamos el archivo **server.js**, su funcionamiento es importante ya que arranca el server, importa el **app.js**, define el puerto y puede manejar errores globales.

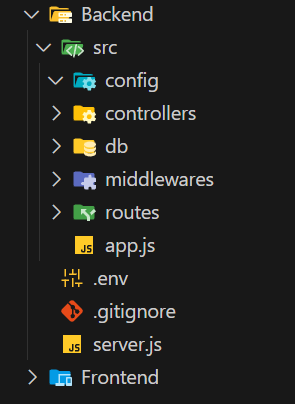


Ilustración 14 archivos externos del src

3n. paso (terminal)

Este paso es crucial para instalar módulos dentro de la carpeta **Backend**

Oprimimos *(Control+ñ)* para abrir la terminal, debemos asegurarnos de que estamos dentro de la carpeta **Backend**



Ilustración 15 carpeta Backend sin ubicar

Para entrar a una carpeta en específico usamos el comando ***(cd Backend)*** dependiendo del nombre que le hemos asignado.

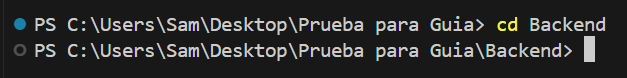


Ilustración 16Carpeta Backend ubicada

El próximo paso y crucial será crear el archivo **package.json**, para hacer esto usaremos la terminar y escribiremos ***(npm init –y)*** o ***(npm –g),*** el **–y** es para un archivo rápido y el **–g** para un archivo personalizado *(ES IMPORTANTE RECALCAR QUE EN ESTE PUNTO DEBEMOS DE TENER INSTALADO NODE.JS).*

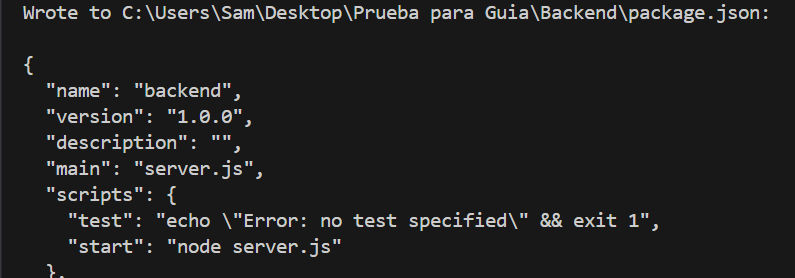


Ilustración 17 package.json ingresado con éxito

4n. paso

Ingresamos a **package.json** que se creó en nuestra carpeta **Backend,** ubicamos la línea “***start”: “node server.js”*** para ingresar debajo de este ***“dev” : “nodemon server.js”*** *(importante añadir coma dentro de “script”).*

*(****nodemon*** *tiene como función reiniciar el servidor a la entrada, actualización, eliminación de archivos, en pocas palabras es el que nos ahorra la molestia de detener el server y volver a iniciarlo para ver los cambios reflejados)*

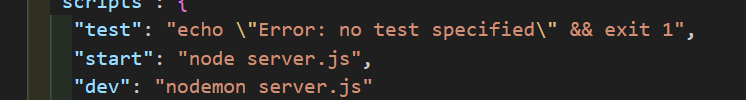


Ilustración 18 ingresar función dev en package.json

5n. paso (instalacion nodemon y modulos)

Comenzaremos a instalar los **módulos** *(este paso es crucial para el funcionamiento del servidor y toda la carpeta* ***Backend****)*

Volvemos a la terminal e ingresamos **npm install --save-dev nodemon**, esto significa que se va a instalar **nodemon** en nuestro **package.json** y se almacenara en **devDependencias.**

Después de instalar *nodemon* exitosamente, ingresaremos el siguiente código **npm i bcrypt cors dotenv express multer mysql2 jsonwebtoken**.

Estos son modulos básicos para que funcione correctamente nuestro servidor y aplicativo

**Bycrypt=** Sirve para **encriptar contraseñas.**

**Cors=** **Ejemplo:** tu **Frontend** corre en ***localhost:3000*** y tu backend en ***localhost:5000***; sin CORS no se podrían comunicar fácilmente.

* **Dotenv=** Permite cargar *variables de entorno* desde un archivo **.env.**
* **Express=** Sirve para crear servidores y manejar *rutas HTTP* **(GET, POST, etc.)** de manera sencilla.
* **Multer=** Middleware para manejar subidas de archivos **(imágenes, PDFs, etc.).**
* **Mysql2=** Driver para conectarse a **MySQL** desde Node.js.
* **Jsonwebtoken=** Permite emitir un token al iniciar sesión y verificarlo en cada petición protegida.

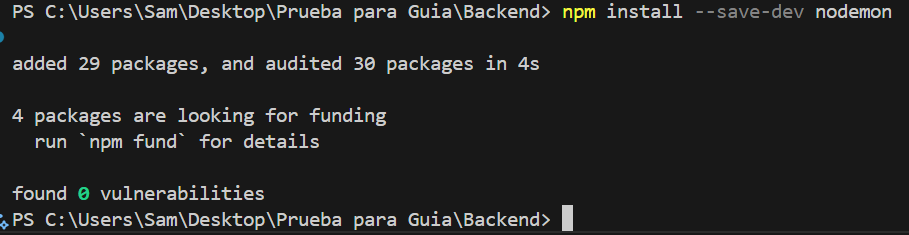


Ilustración 19 nodemon instalado exitosamente



Ilustración 20 carpeta node\_modules instalado correctamente

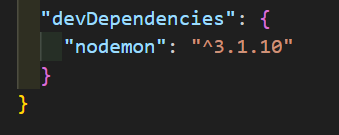


Ilustración 21 nodemon instalado en devDependencias instalado correctamente

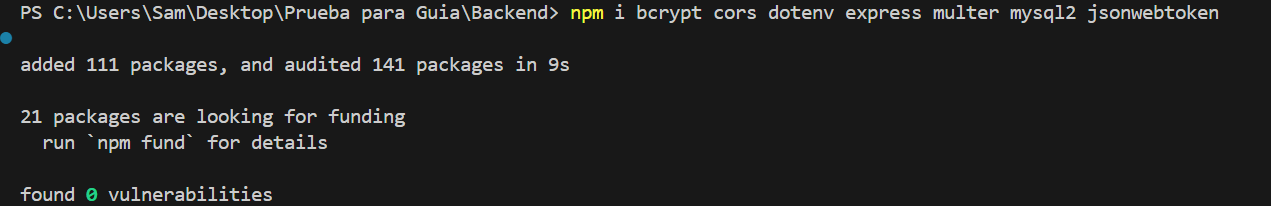


Ilustración 22 modulos instalados correctamente



Ilustración 23 dependencias instalas correctamente en el package.json

6n. paso (introduccion de funciones para las carpetas y archivos backend)

Comenzaremos a añadir el funcionamiento a las carpetas y archivos que creamos en nuestra carpeta Backend

Empezaremos por .env y crearemos el **PORT**, **DB\_HOST,** **DB\_USER**, **DB\_PASSWORD,** **DB\_NAME** *(Es importante escribir* ***DB*** *y no BD, no confundir con base de datos, ya que el sistema solo reconoce* ***Data Base****).*

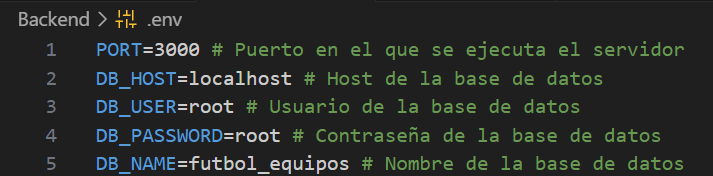


Ilustración 24 Implementación de variables de entorno

Entraremos a la carpeta **src** y nos dirigimos a la carpeta **config** y creamos un archivo JS llamado **conexion\_DB.js**, e implementaremos el siguiente código.

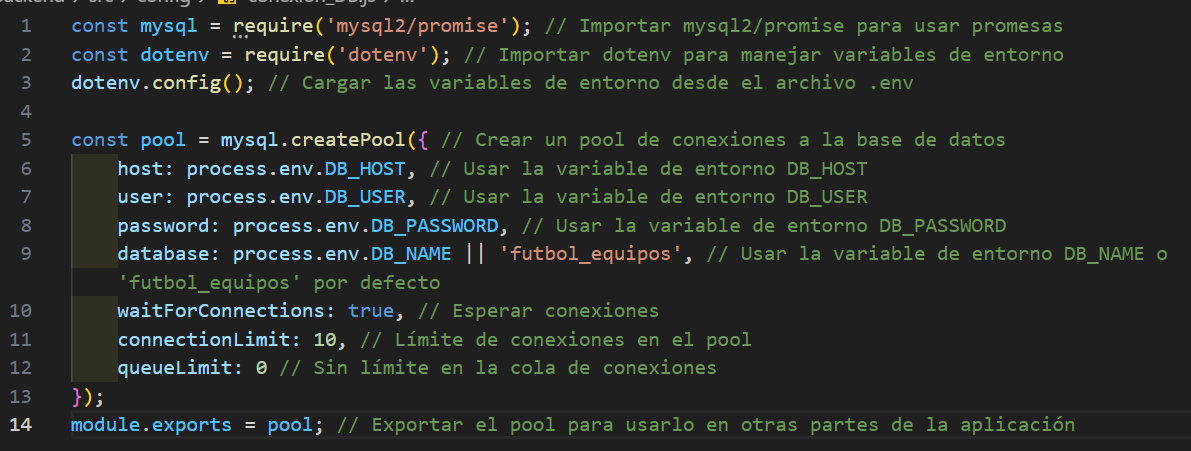


Ilustración 25 archivo conexión de la base de datos para usar las variables de entorno

Ingresamos al a carpeta middlewares y añadimos el archivo **uploads.js,** y añadimos el siguiente código.

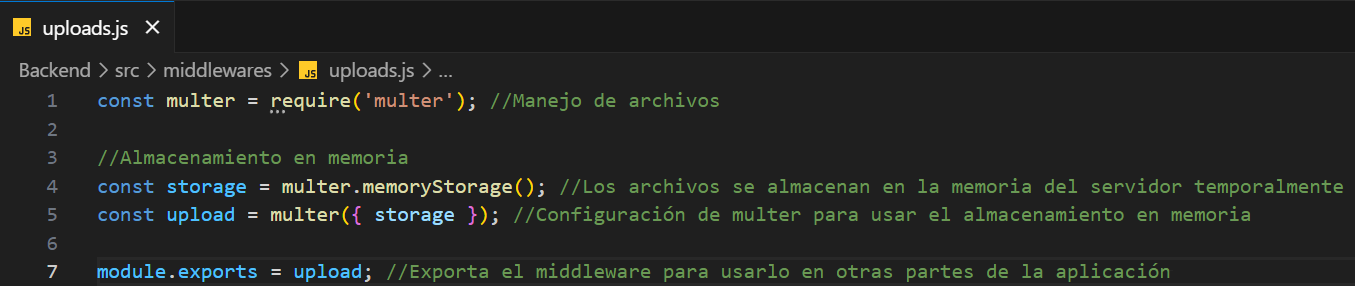


Ilustración 26 archivo uploads

Salimos del src y vamos al archivo .**gitignore** que creamos anteriormente, esta cumplirá la función de ignorar los repositorios innecesarios a la hora de subir nuestro proyecto a ***github.***

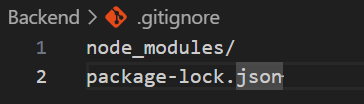


Ilustración 27 Código gitignore

Después volvemos src e ingresamos a la carpeta **controllers** y crearemos el *archivo JS* llamado **crud.controller.js** *(esta parte es muy importante ya que creamos los controles* ***CRUD*** *para el funcionamiento de la base de datos con el Backend y que se muestre correctamente al Frontend).*

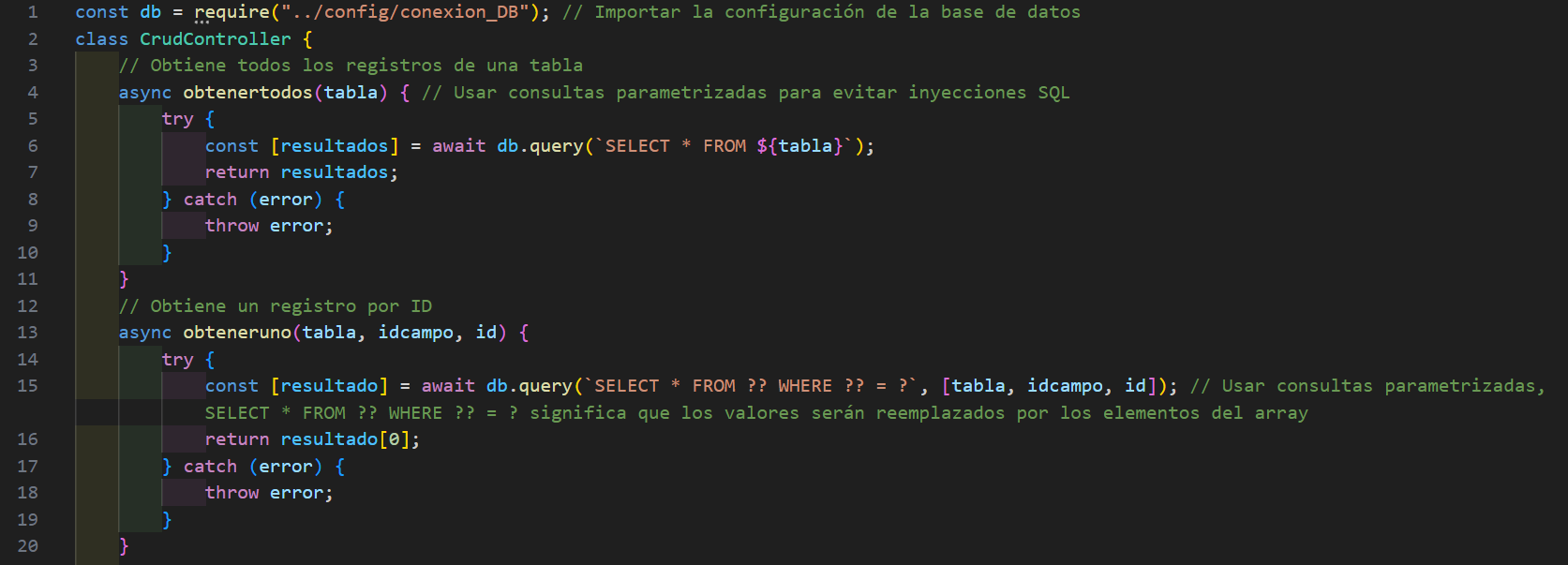
**

Ilustración 28 Implementación del código CRUD

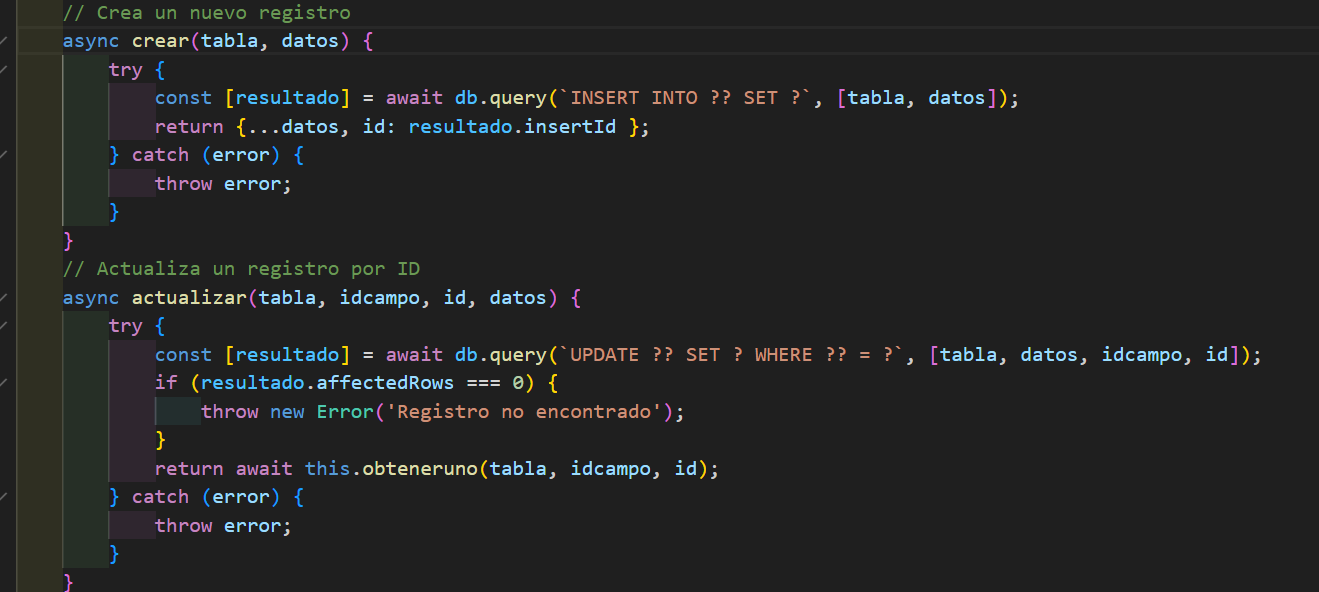


Ilustración 29 CRUD crear y actualizar

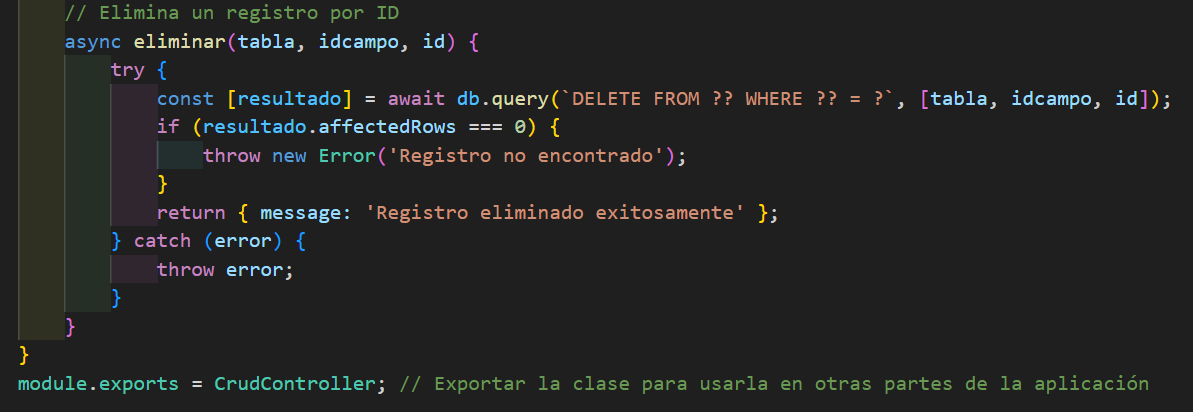


Ilustración 30 CRUD eliminar y exportación de la clase CrudController

Iremos después a la carpeta **routes** y crearemos un archivo **JS** como por ejemplo **equipos.routes.js.**

Este cumplirá la función de ubicar la tabla al **crud** que creamos en **controllers**, básicamente le dará el funcionamiento **CRUD** a la tabla que creamos en la base de datos, así que se podrían crear muchos más **routes** para diferentes tablas.

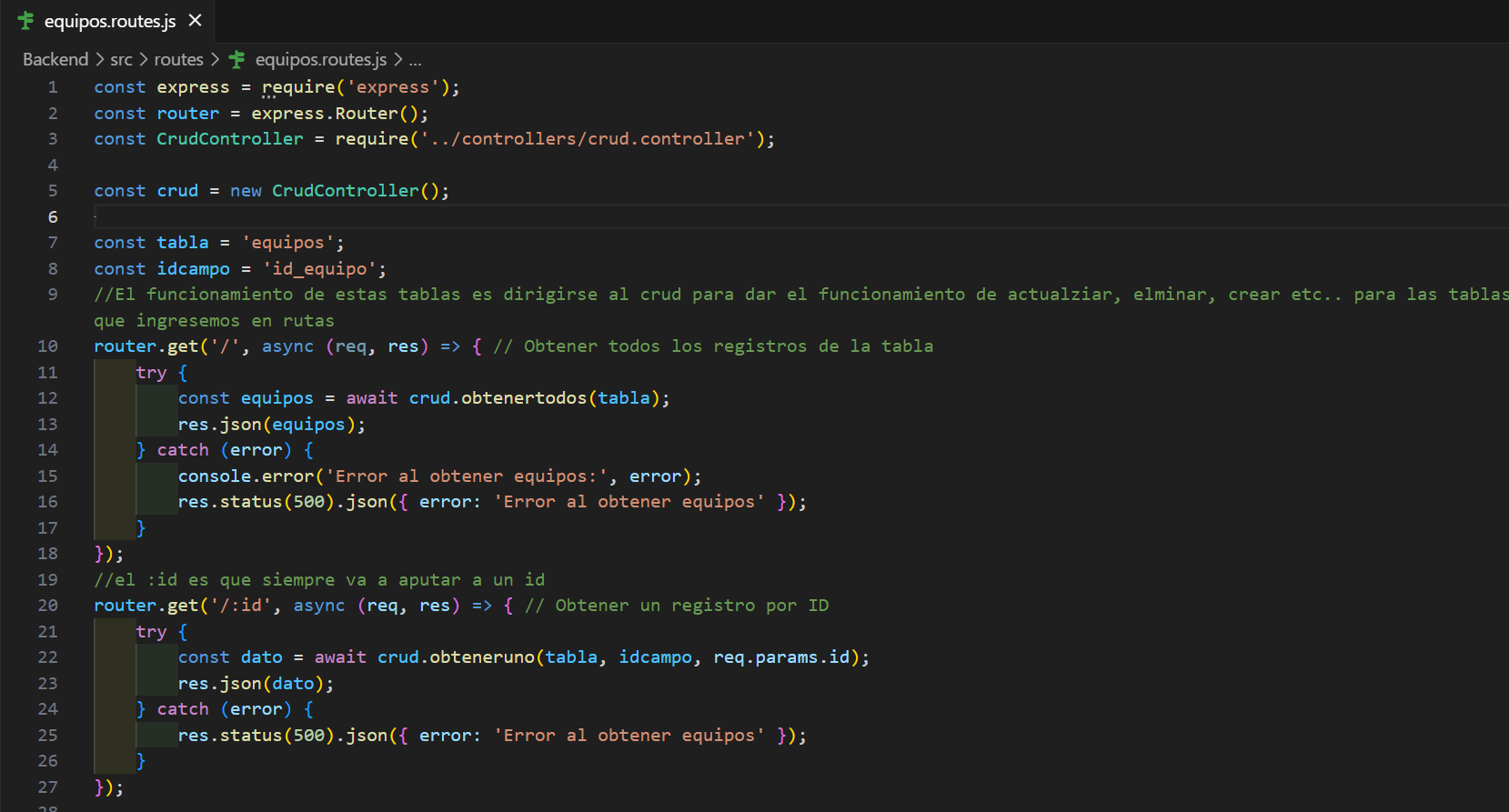


Ilustración 31 Rutas para la base de datos y el CRUD obtener (GET)

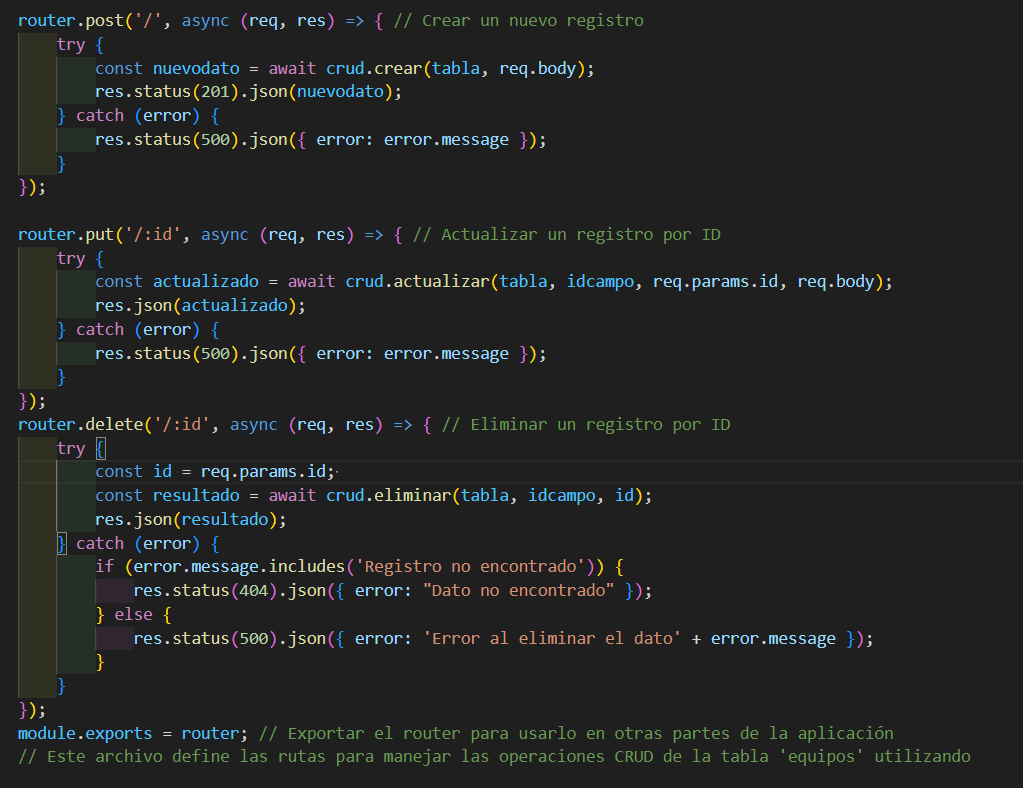


Ilustración 32 Rutas (POST) (PUT) (DELETE)

Ahora nos dirigimos a la carpeta app.js e ingresamos el siguiente código, esto es para asegurar que la dirección de **routes** funcione correctamente.

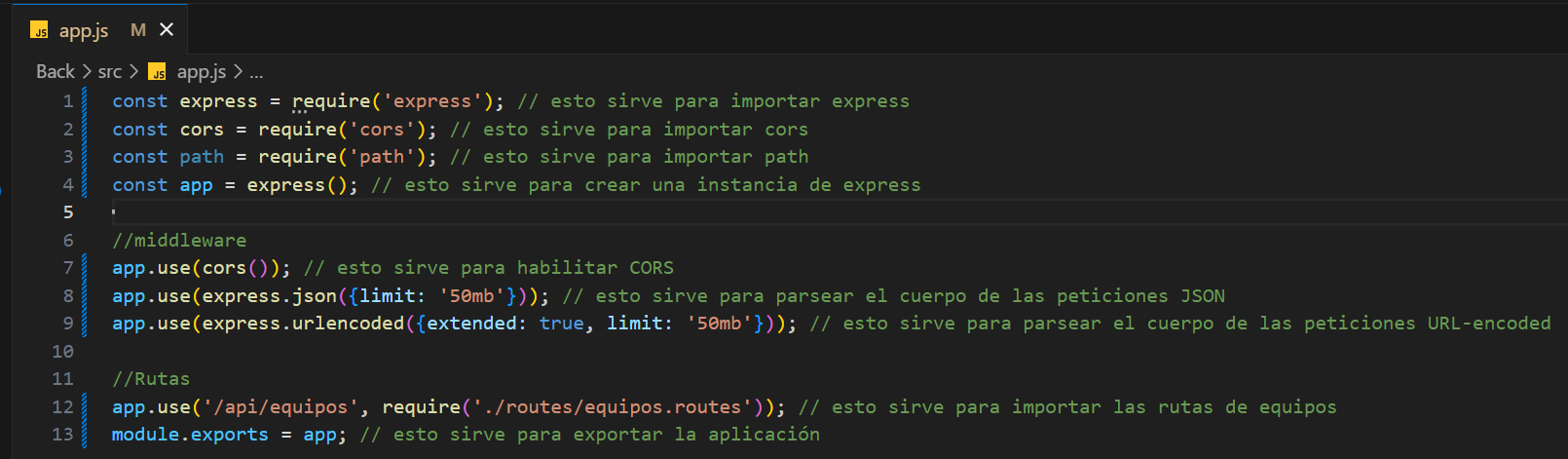


Ilustración 33 Código para app.js

Después vamos a **server.js** e ingresamos el siguiente código, este es como el interruptor de encendido para el **Backend.**

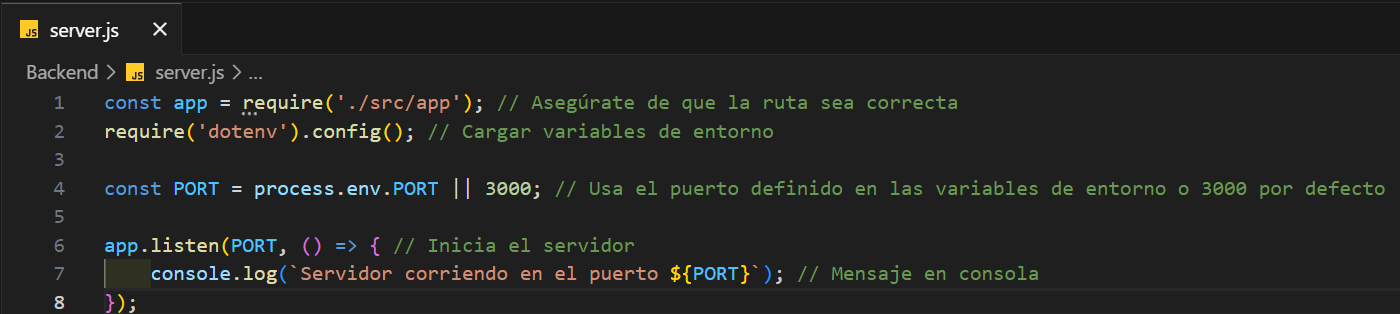


Ilustración 34 Código para Server.js

Por ultimo nos dirigimos a la carpeta **db** y creamos el archivo **DB.sql,** en este podremos usar la sintaxis de ***mysql*** donde crearemos una base de datos con CREATE, se usará esta base de datos con USE, se creará una tabla con **CREATE TABLE**, se ingresarán datos con **INSERT INTO VALUES**, y podremos ver nuestra tabla y datos con **SELECT \* FROM** *(todo esto para verlo dentro* ***MySQL Workbench****).*

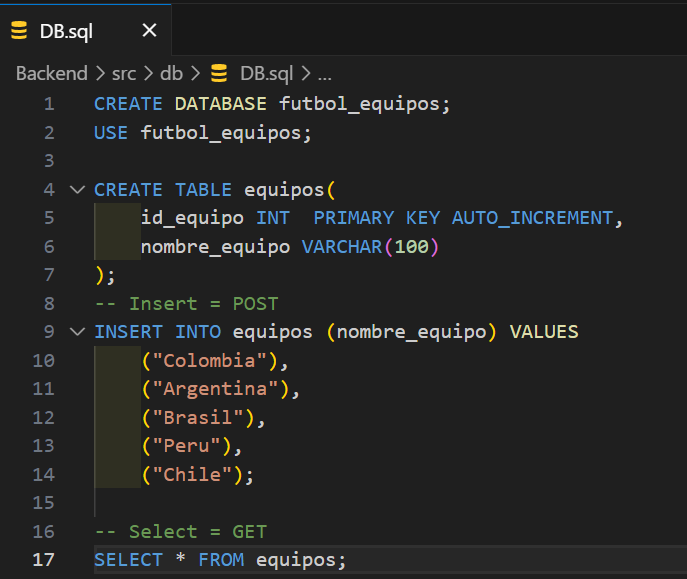


Ilustración 35 Implementar archivo BD.sql y añadir base de datos, tablas y datos

Y muy importante asegurarse que dentro de nuestra base de datos este conectada, crearemos un servidor dentro de ***MySQL Workbench*** (tener presente que este debe de contar con el usuario **root** con la misma contraseña)

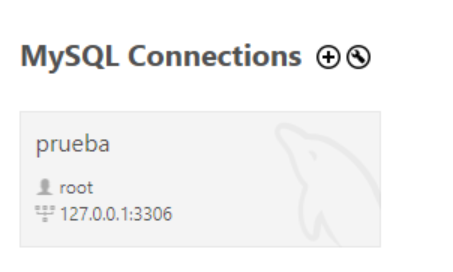


Ilustración 36 Crear servidor MySQL

Ingresamos, copiamos y pegamos el código que hicimos en **Visual estudio** y lo ejecutamos en el icono de rayo del programa.

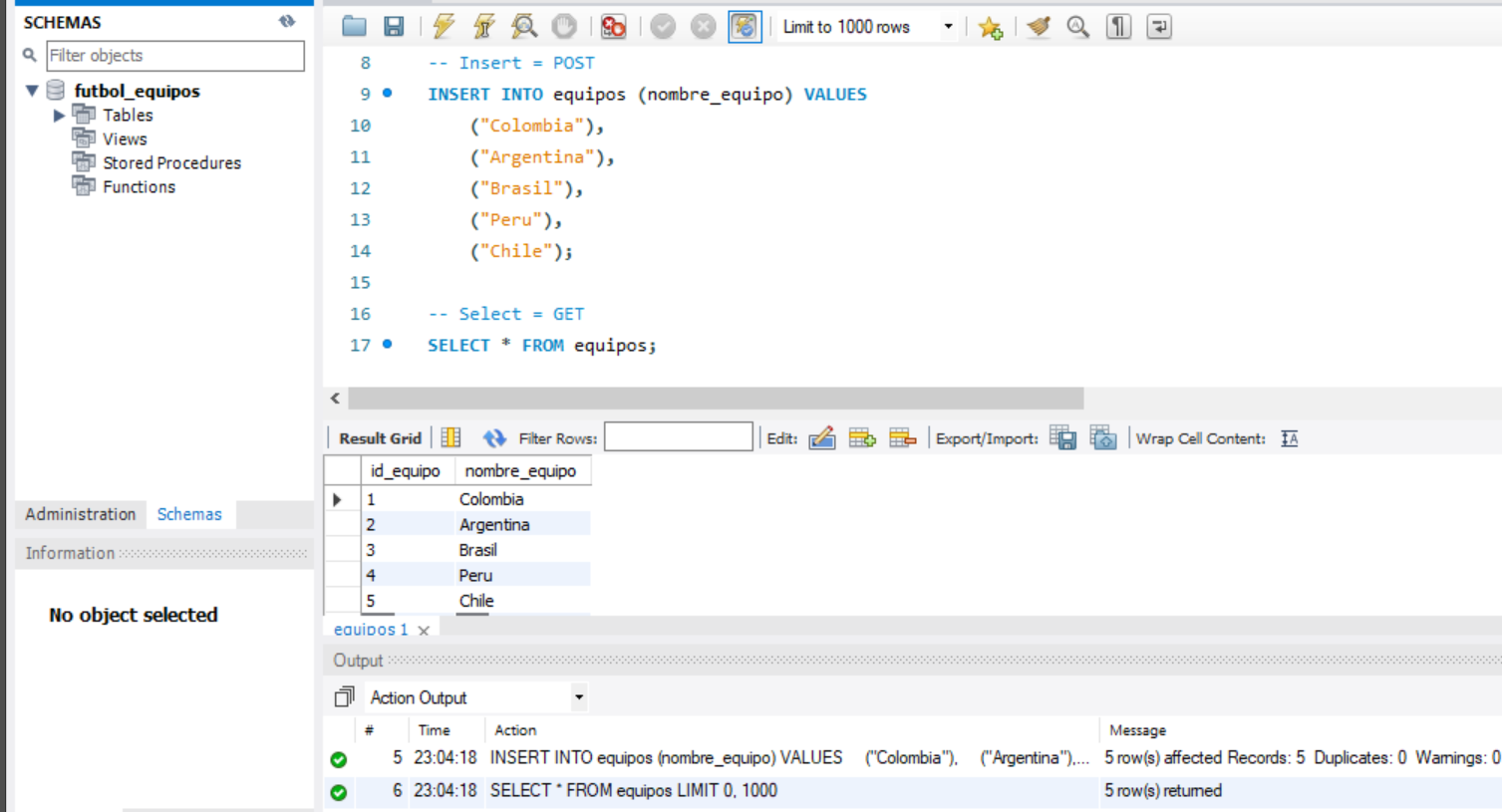


Ilustración 37 Creación de bd, tabla y datos en MySQL Workbench

7n. paso (pruebas de funcionamiento

Para comprobar que toda nuestra carpeta Backend funciona correctamente volveremos a la terminal e ingresaremos lo siguiente **npm run dev,** si quieres parar su funcionamiento basta con oprimir *(control+c).*

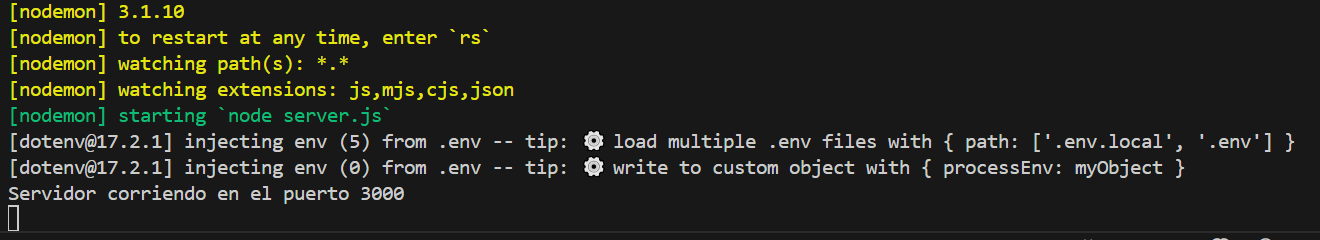


Ilustración 38 Mensaje terminal, servidor corriendo exitosamente

Ahora para comprobar que los controles **CRUD** funcionan correctamente tendremos que instalar la extensión **Thunder Cliente.**

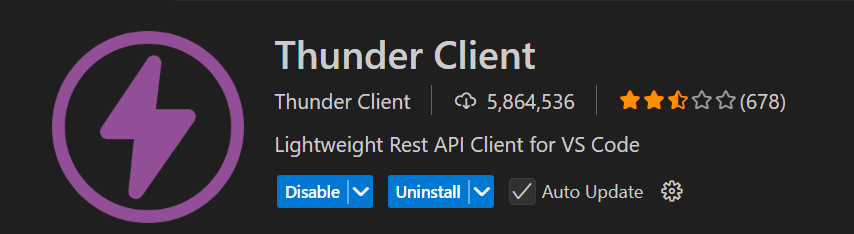


Ilustración 39 Extensión Thunder Client

Nos dirigimos a la extensión y le damos a **New Request,** y dentro del bloque de texto cambiamos la URL por nuestro api, que con el procedimiento que realizamos debería de ser **localhost:3000/api/equipos,** dependiendo del caso este puede cambiar. *(puedes copiar esta ruta correctamente yendo directamente a* ***app.js '/api/equipos****')*

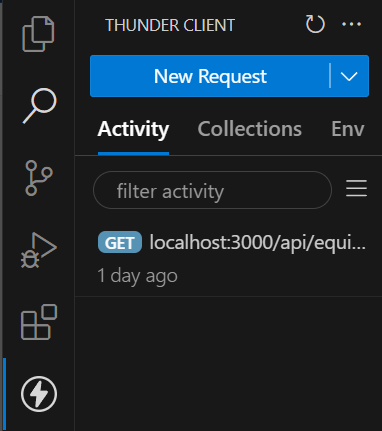


Ilustración 40 Extensión Thunder Cliente ingresar prueba de BD

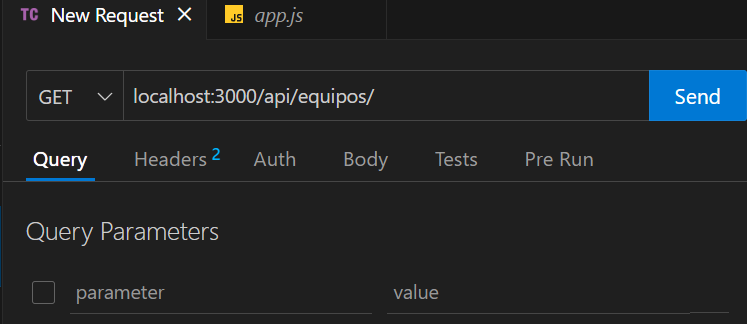


Ilustración 41 Ingresar URL de nuestro servidor local

Si hasta el momento haz seguido los pasos correctamente, al presionar el botón **send** debería de traerte los datos que ingresaste en la tabla *(Importante iniciar servidor para que esto funcione)*

El primero que obtendremos es el **GET**, por supuesto nos dará como respuesta los datos que tenemos en nuestra base de datos.

El segundo que obtendremos es el **POST**, para probarlo copia uno de los datos, dirígete a **Body** y cópialo, cámbiale el nombre si gustas. (No es necesario añadir una id, se implementará gracias **auto\_increment** dentro de la base de datos)

El tercero que obtendremos es el **PUT,** para probarlo debes de escoger una id dentro de la **URL**, usare de ejemplo el nuevo dato que ingrese gracias al **POST.**

El cuarto y último que obtendremos es el **DELETE**, para probarlo solo tendremos que ingresar en la **URL** el numero id dentro de la tabla.

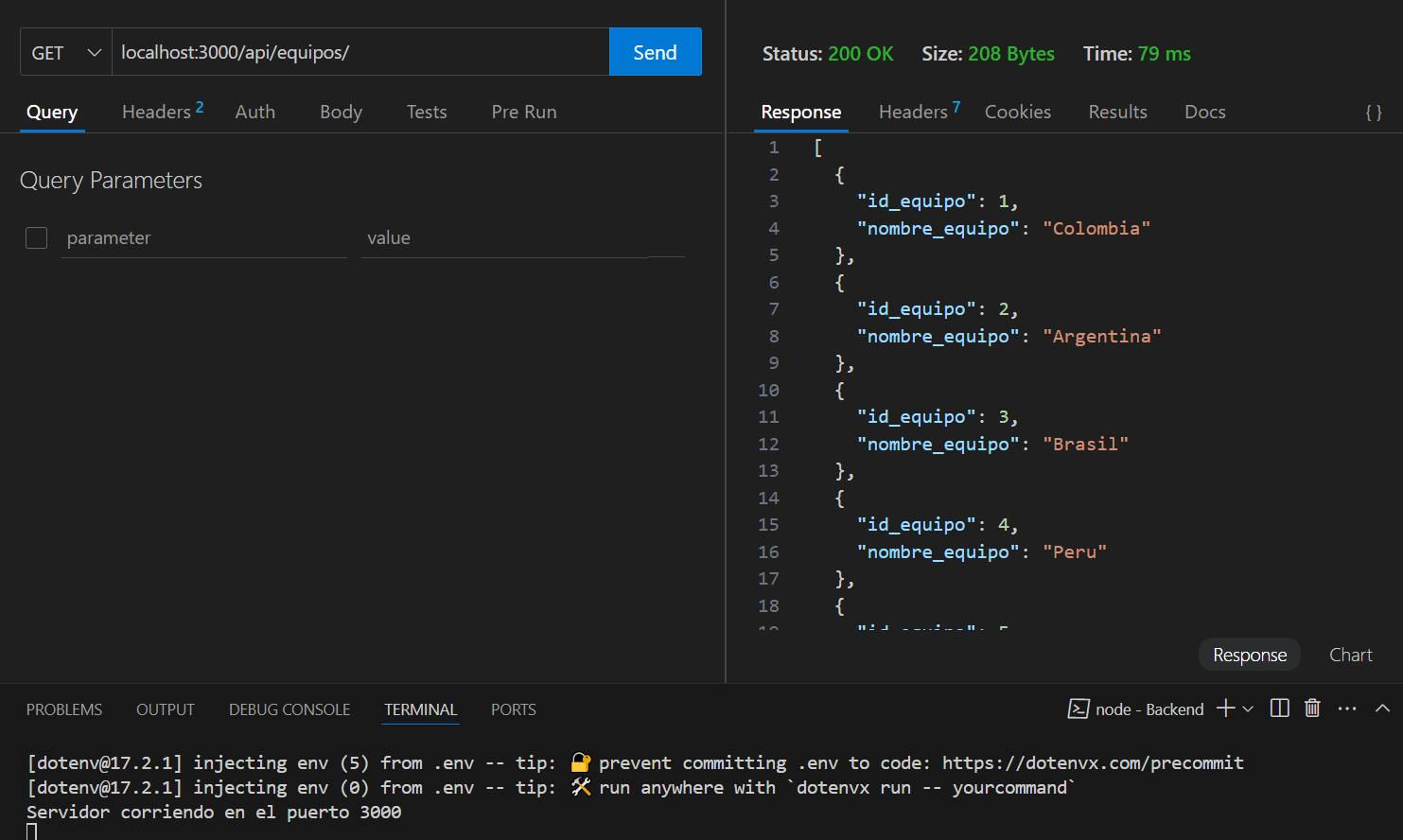


Ilustración 42 Funcionamiento GET

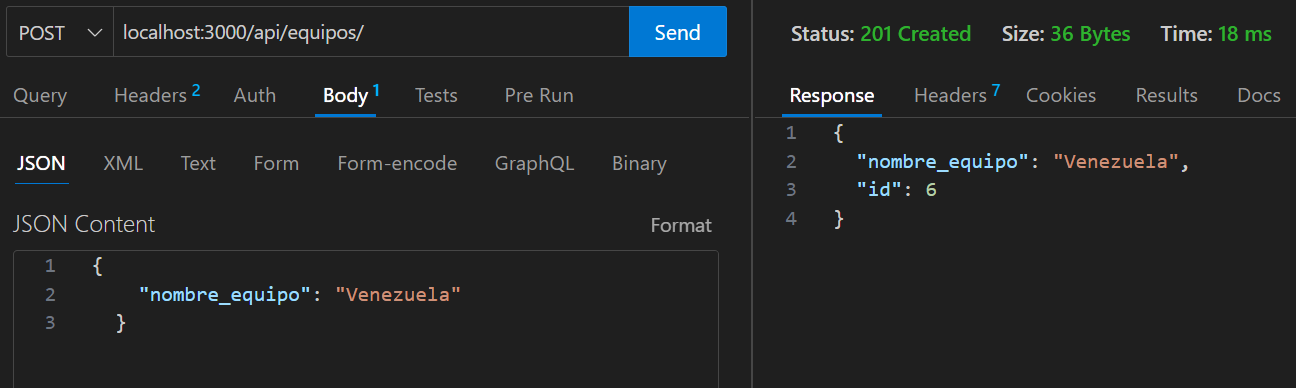


Ilustración 43 Funcionamiento POST

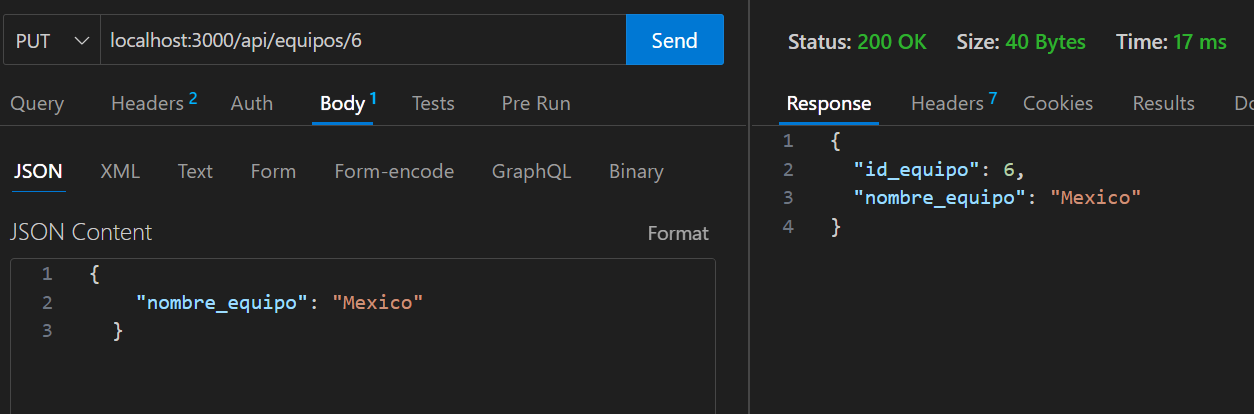


Ilustración 44 Funcionamiento PUT

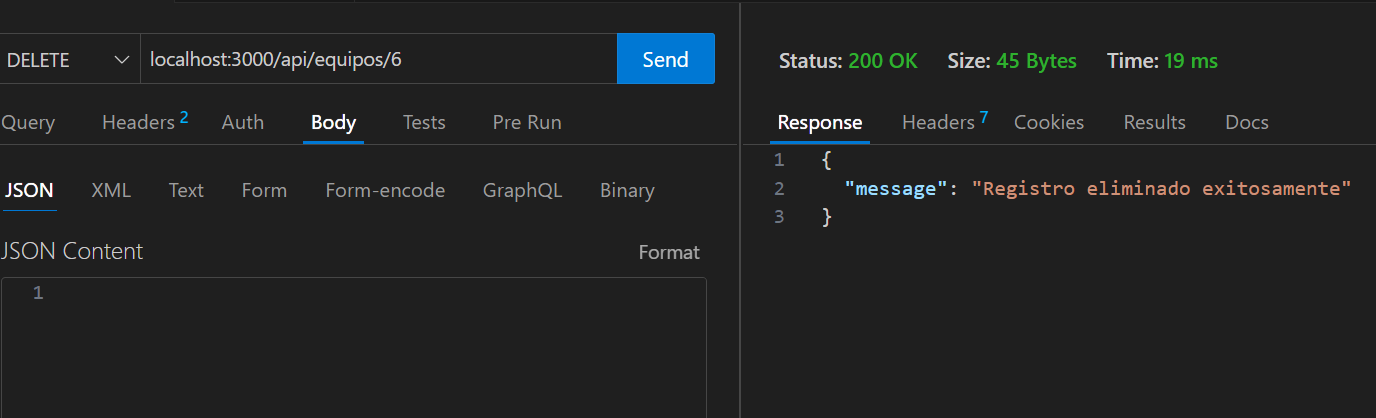


Ilustración 45 Funcionamiento DELETE

8n. paso (prueba en pagina web)

Llegamos al final, para comprobar el satisfactorio comportamiento de una base de datos en una página web, para esto tienes que tener la extensión **Live Server.**

Cuando lo obtengas dirígete al archivo **HTML** dentro de la carpeta **Frontend**, presiona Click Derecho y dale click a ***Open With Live Server.***

Si todo salió bien deberías de ver los equipos de futbol que ingresaste en la base de datos.

Ten en cuenta consola por si tienes algún error para poder solucionarlo.

Ya puedes ver, ingresar, editar y eliminar los datos que quieras.

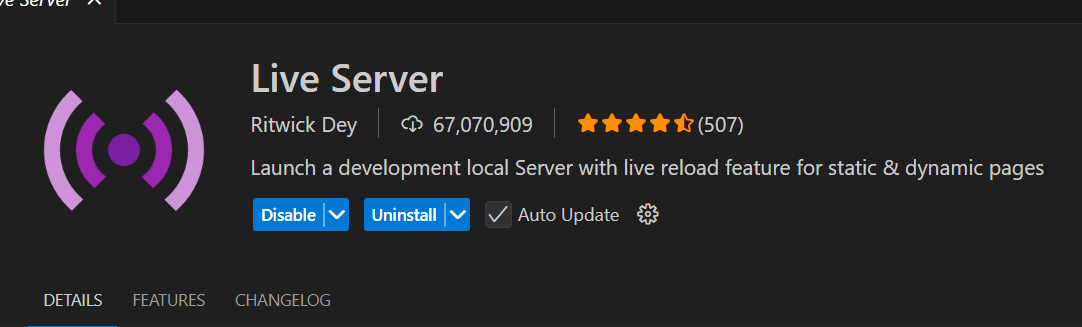


Ilustración 46 Extensión Live Server



Ilustración 47 Prueba con Live Server GET

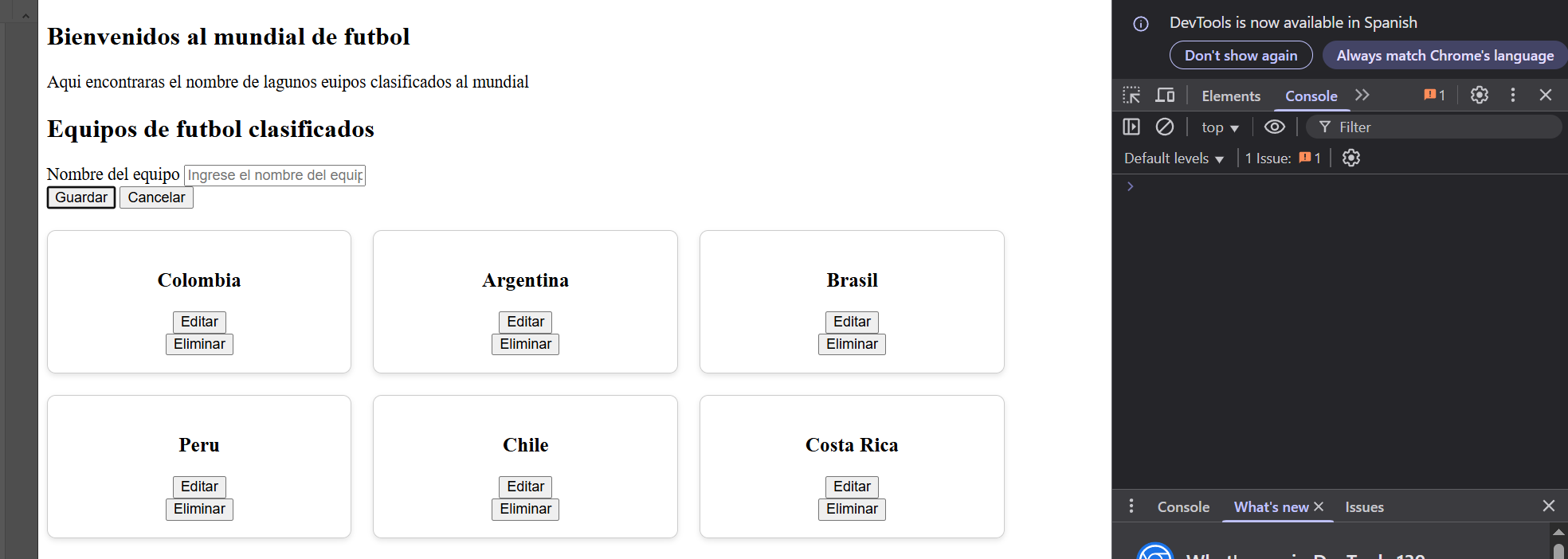


Ilustración 48 Prueba POST

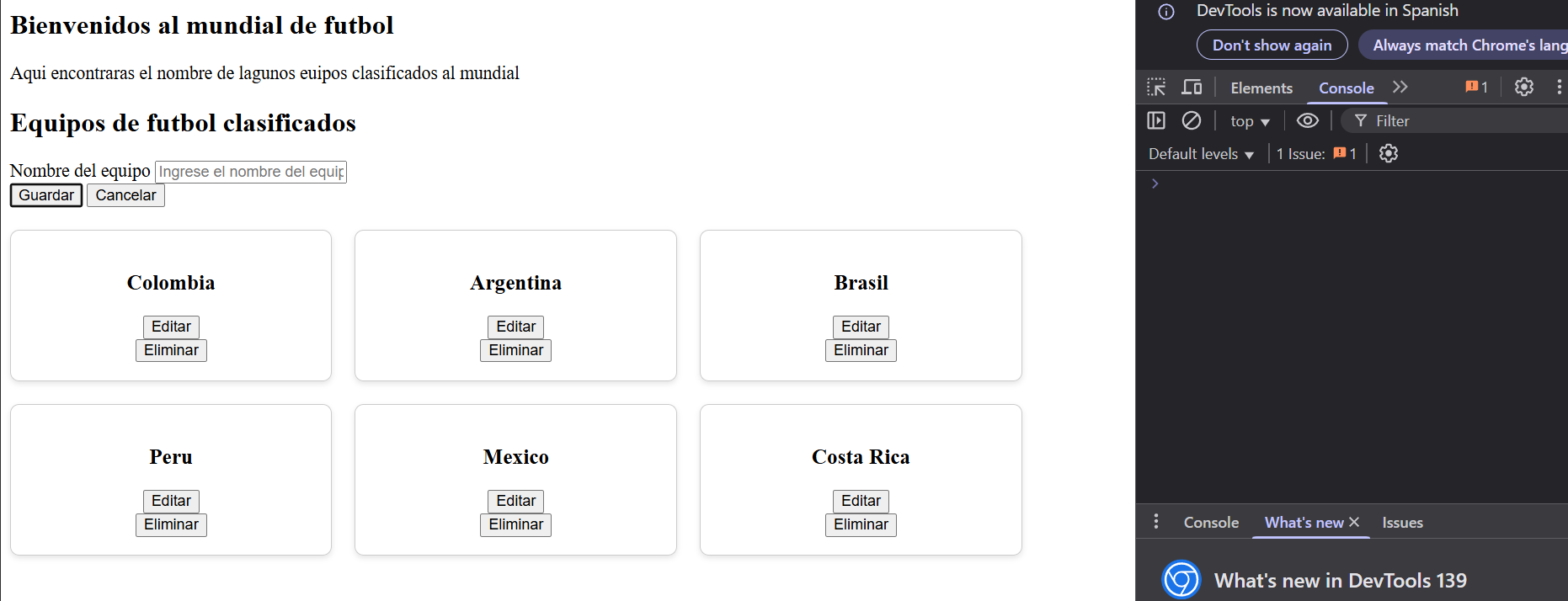


Ilustración 49 Prueba PUT

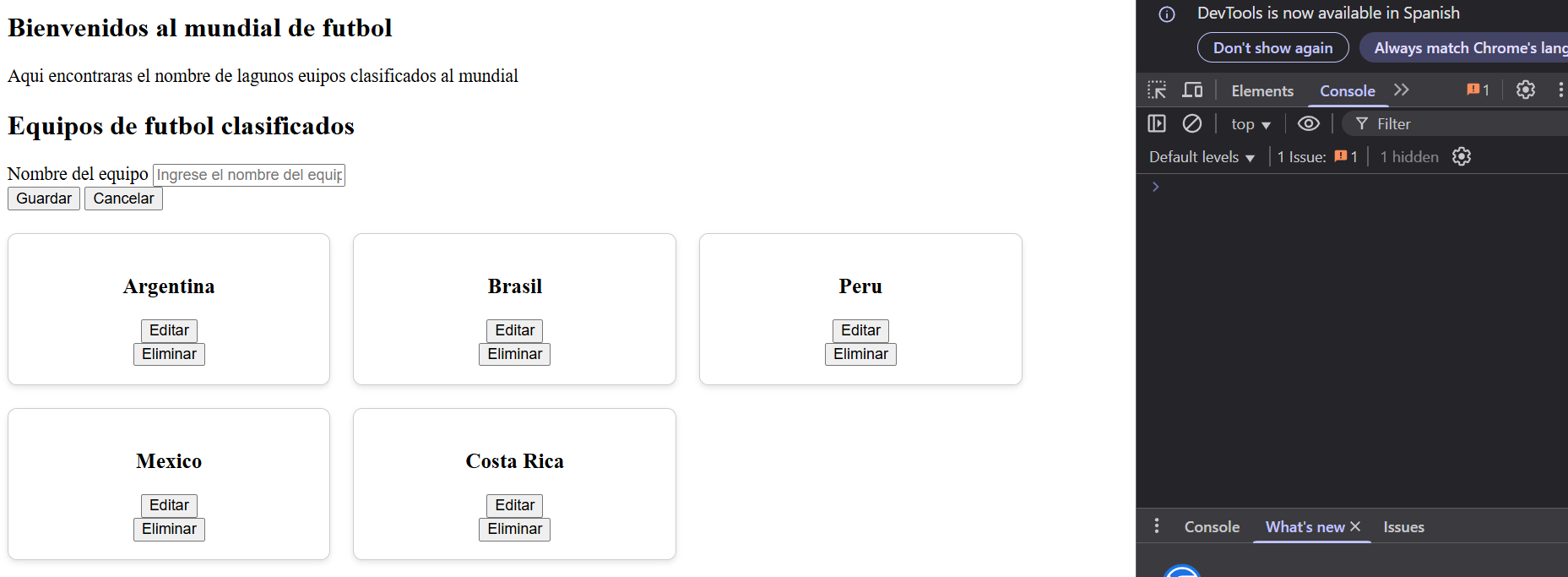


Ilustración 50 Prueba DELETE

¡PASO EXTRA! (guardar proyecto en repositorio con git y github)

Supongo que después de un gran trabajo en tu proyecto deseas guardarlo, pues lo que necesitas es instalar **git** de escritorio y tener una cuenta ***GitHub*** para almacenar tu proyecto en repositorio.

Cuando tengas los dos, ve a **GitHub** y crea un nuevo repositorio, cuando tengas el repositorio disponible podrás ir a tu carpeta que contiene el **Backend** y el **Frontend**, usa el click derecho y dale a **Git Bash Here**.

(recuerda configurar tu usuario y correo electrónico de GitHub en tu git con ***git config --global user.name "Tu Nombre" git config --global user.email*** [***tu\_correo@example.com***](mailto:tu_correo@example.com)***)***

Los siguientes códigos que usaremos dentro de **git** serán **git init,** esta tiene como objetivo inicializar un nuevo repositorio y dará paso a los siguientes códigos.

Usaremos **git add .** para añadir todos los archivos para el **commit.**

Después usamos el **git commit -m "el texto que tu quieras",** el cual es como un punto de guardado en nuestro proyecto.

Usamos el **git branch -M main,** el *main* es para seguir el estándar actual de **GitHub**

Por ultimo usamos estos dos códigos **git remote add origin “url del repositorio”** y **git push -u origin main,** el primero es para decirle al **git** que use este repo *(repositorio)* de **GitHub** como destino y el segundo comando sube tu rama **main** a **GitHub** y la vincula para que los siguientes **git push** y **git pull** funcionen sin que tengas que volver a especificar todo.

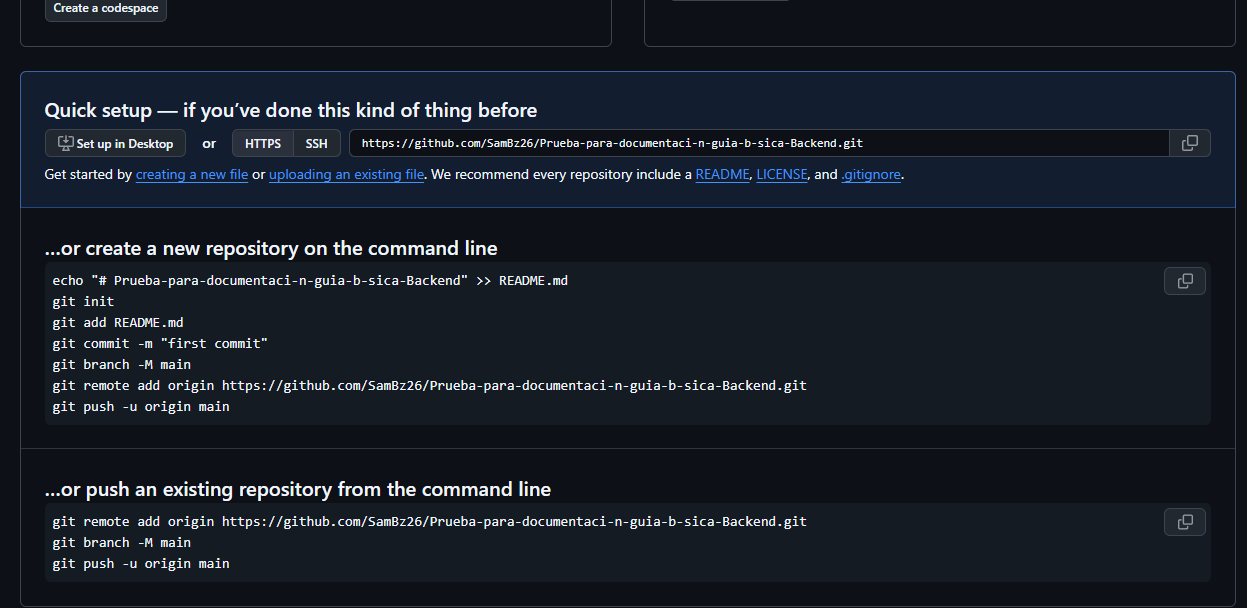


Ilustración 51Repositorio GitHub creado

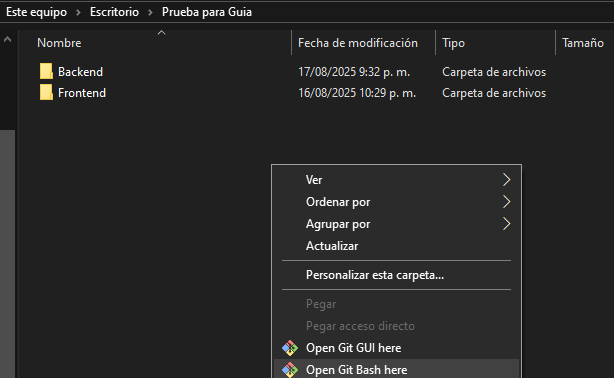


Ilustración 52 Usar Git bash en la carpeta contenedora

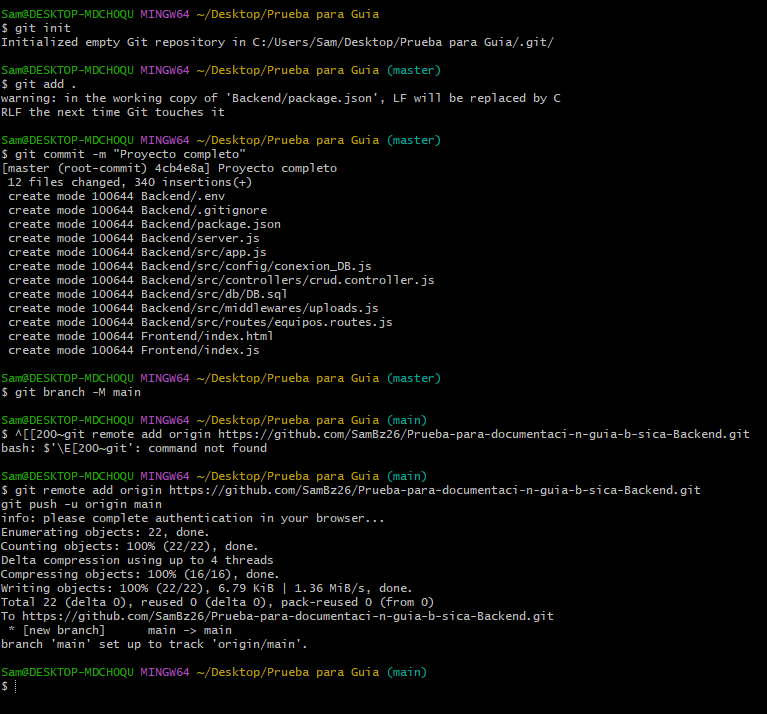
**

Ilustración 53 proceso para subir proyecto a repositorio GitHub

2n paso extra (actualizar respositorio git)

Si en tal caso quieres actualizar el repositorio que almacenaste anteriormente.

Verifica el estado de tu repositorio con **git status *(después de haber hecho git bash here),*** este te mostrara los cambios que realizaste con anterioridad

Después de verificar el estado del repositorio usa **git add .** para guardar los cambios también puedes usar ***git add archivo.js*** para usar guardar un archivo en específico.

Confirma los cambios con nuevo commit con ***git commit -m "Fix: corregido bug en login"*** *(Recuerda que el commit es como un punto de guardado, como en los videojuegos).*

Y por último sube los cambios al repositorio remoto con **git push origin main,** si usas otra rama puedes cambiar el **main** por la rama que usas.

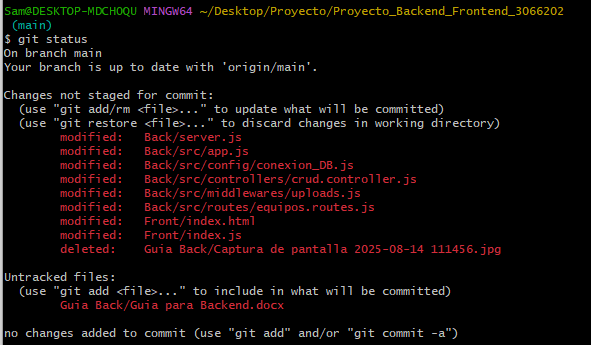


Ilustración 54 uso de git status

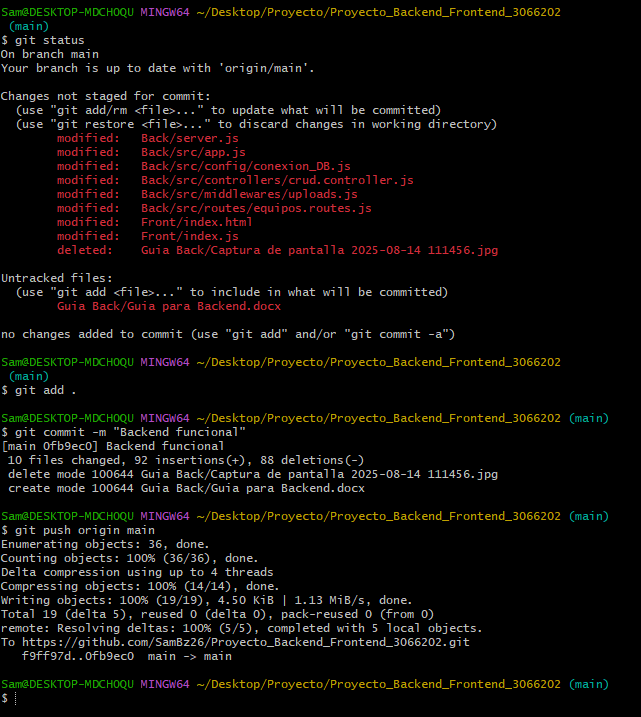


Ilustración 55 Proceso para actualizar proyecto en el repositorio GitHub

3n. paso extra (aplicar estilos con css)

Si después de todo esto aún te quedas con ganas, pues realizaremos algunos pequeños cambios en el diseño de nuestra página web.

Haremos esto con **CSS,** vamos a nuestra carpeta Frontend y creamos un nuevo archivo llamado **style.css**.

Después añadiremos con una etiqueta **link** la ruta **CSS** que creamos.

En este caso no crearemos **clases**, será un **CSS** básico usando las etiquetas principales de nuestro archivo **HTML,** así que vamos directamente a nuestro archivo **style.css** y construiremos un estilo básico para el **body, nav, h1, h2,form, input, button, .btneliminar, button:hover, .card\_equipos** *(la explicación de cada etiqueta* ***css*** *estará en las imágenes).*



Ilustración 56 Creación de archivo CSS

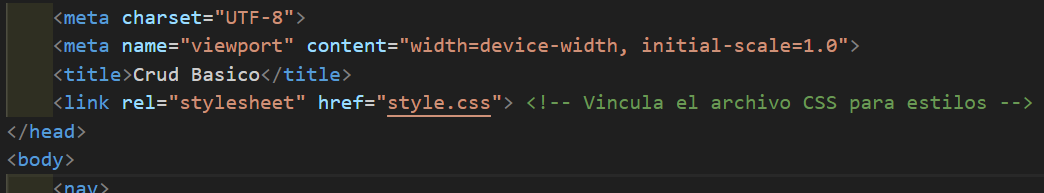


Ilustración 57 Vincular archivo CSS a la cabeza del HTML

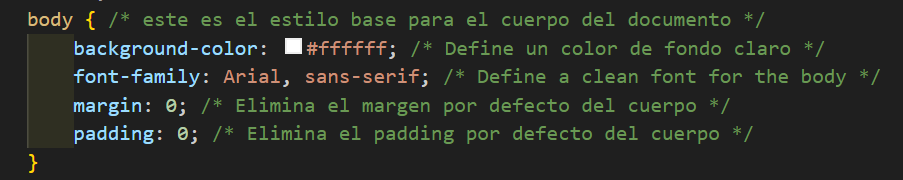


Ilustración 58 Estilo Body

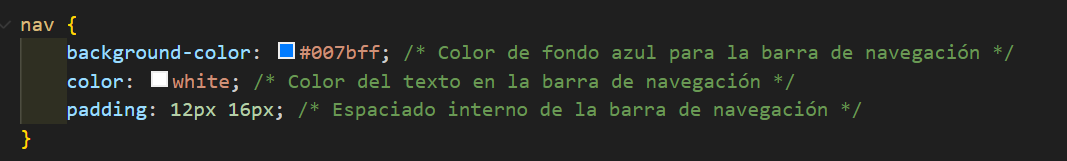


Ilustración 59 Estilo Nav

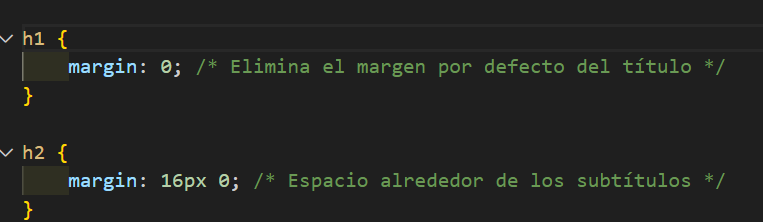


Ilustración 60 Estilos títulos

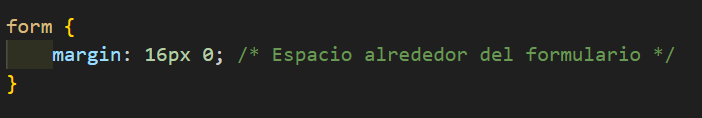


Ilustración 61 Estilo Formulario

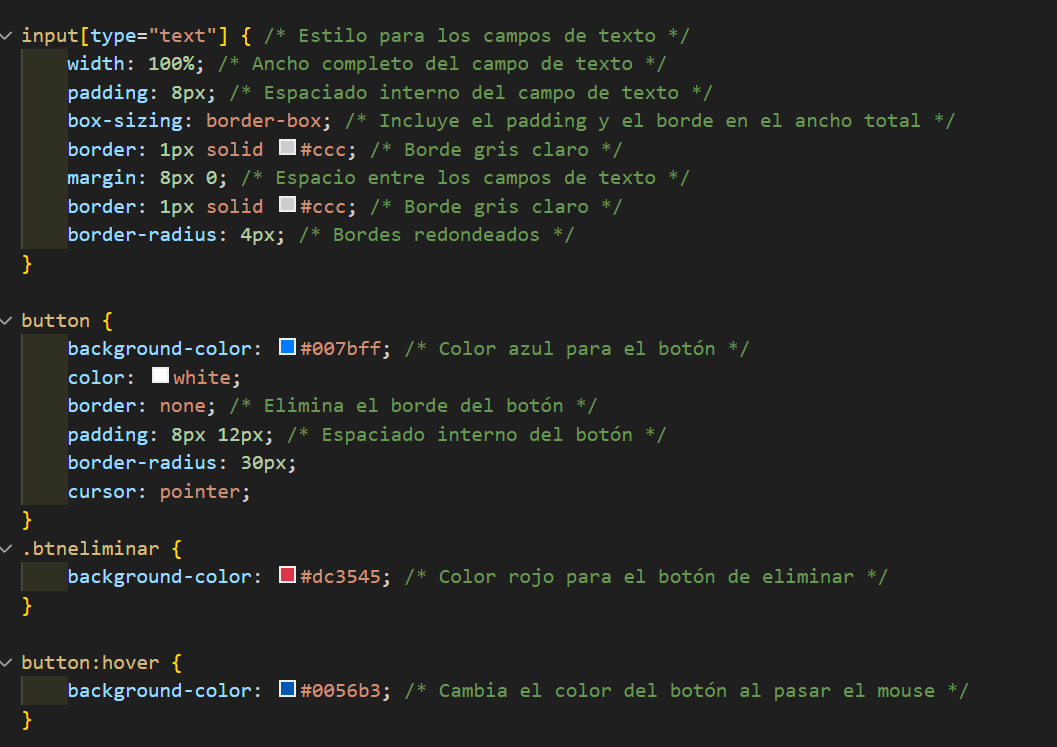


Ilustración 62 Estilos para botones y cuadro de texto

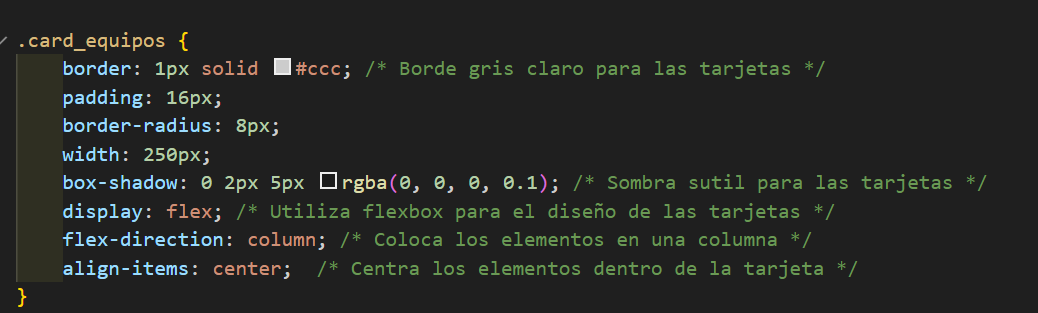


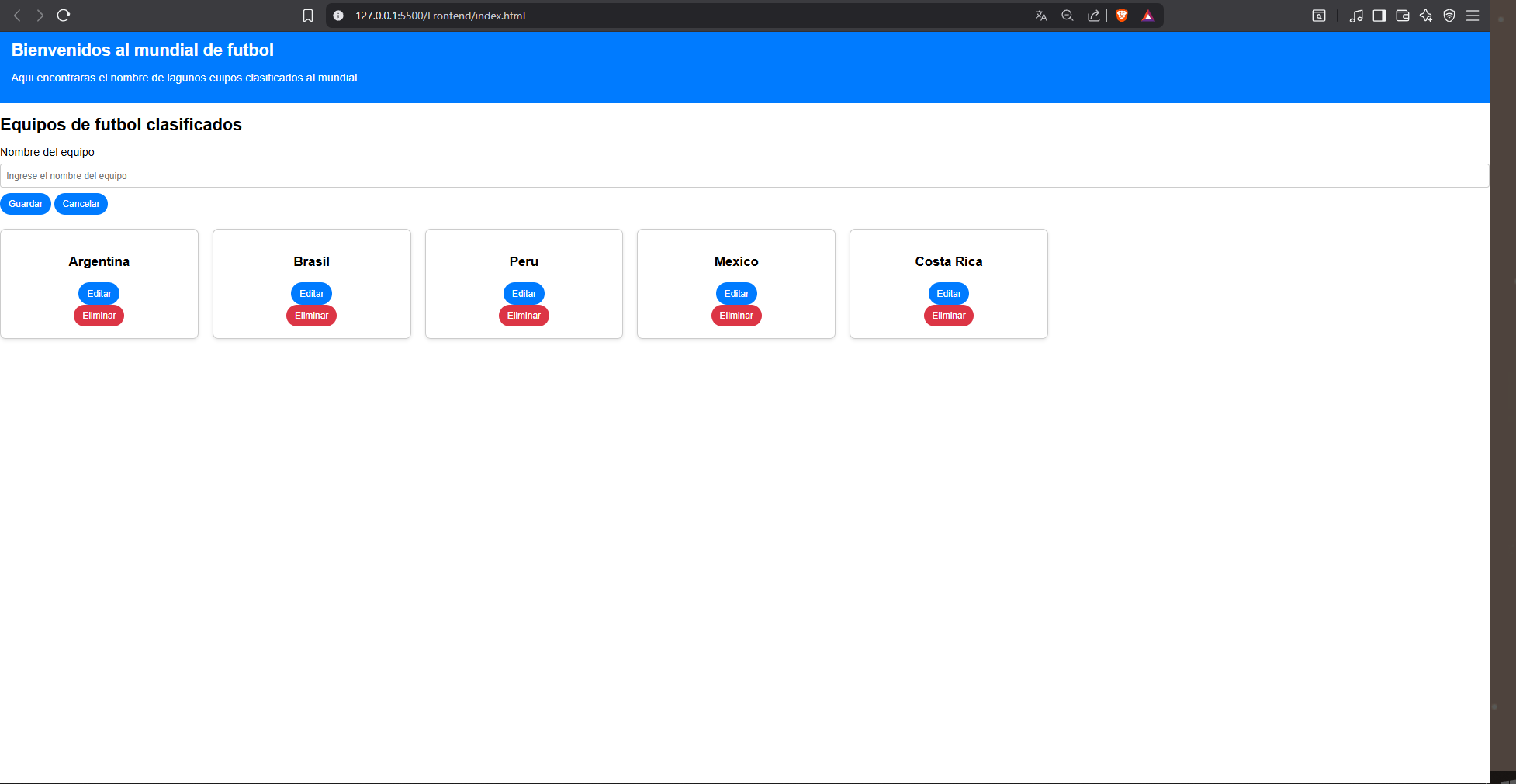
Ilustración 63 Estilo para los Cards

Ilustración 64 Resultado aplicando estilo a la pagina