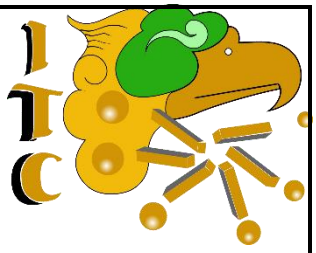


TECNOLOGICO NACIONAL DE MEXICO

INSTITUTO TECNOLOGICO DE CUAUTLA

- BELAUNZARÁN FIORENTINI JOSEL
- ESPÍNDOLA TORRES ÁLVARO ISAIAS
- GUZMÁN GÓMEZ ÁNGEL HIRAMG
- JUÁREZ MENDOZA EFRAÍN
- PÉREZ SÁNCHEZ ANGIE HARUMI
- VIVAR TIEMPOS RIGOBERTO

PROGRAMACIÓN LÓGICA Y FUNCIONAL

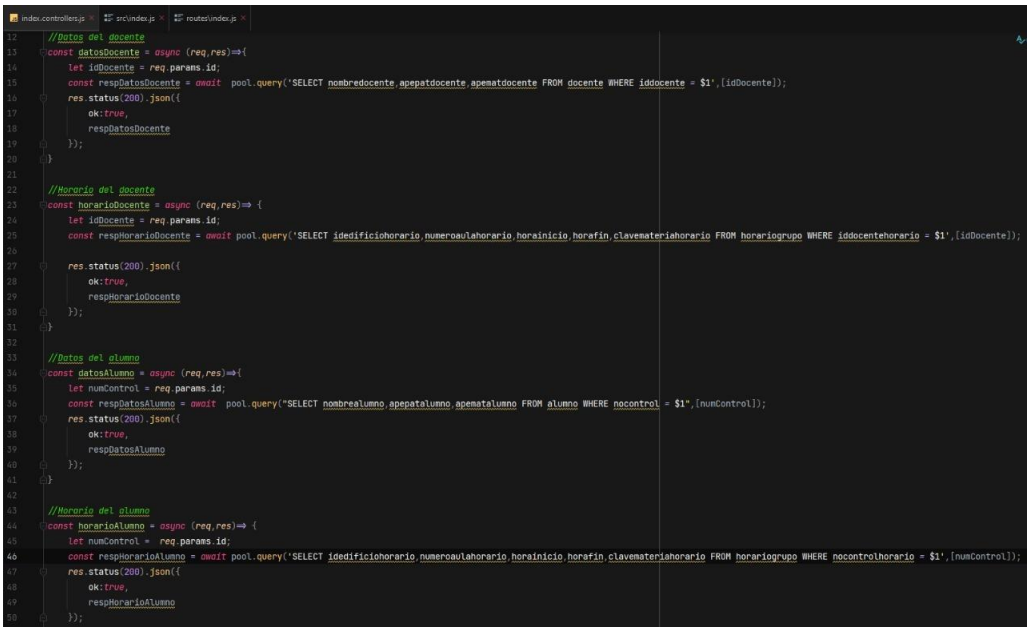
ING. GISELA VEGA TORRES

EJERCICIO HORARIO.

7mo SEMESTRE GRUPO 1

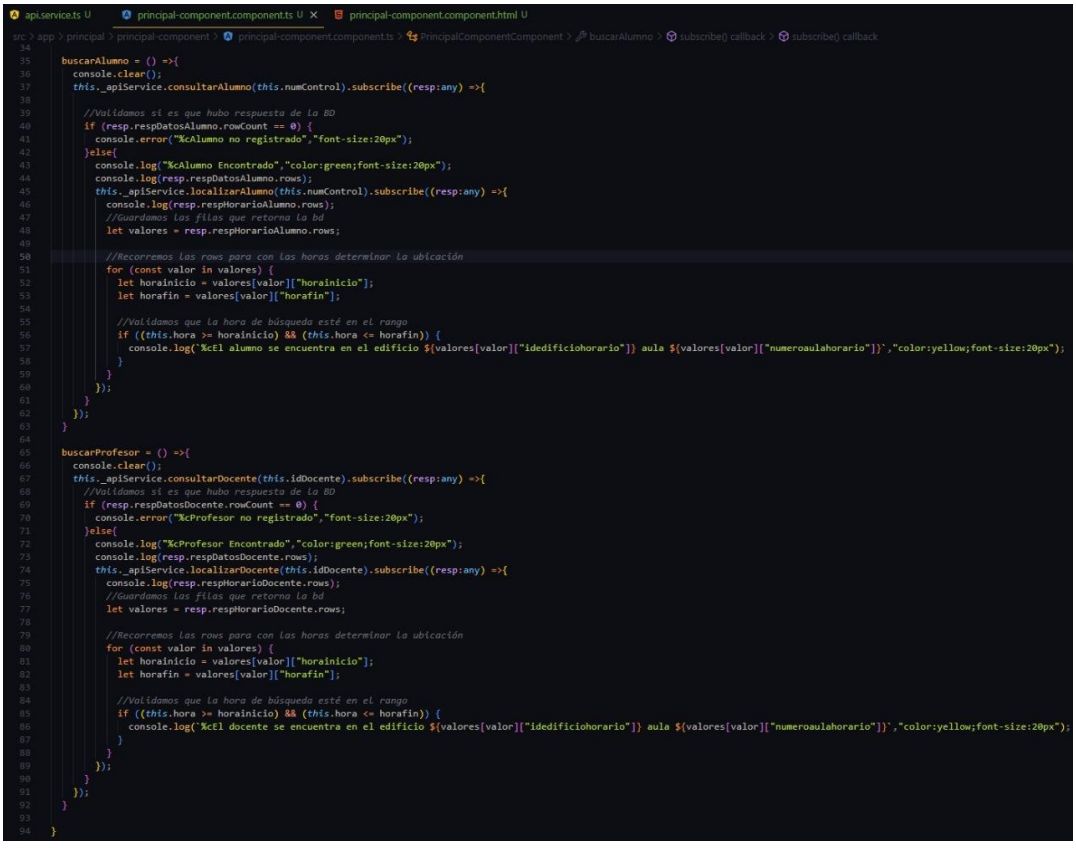
Codificación.

Aquí tenemos el BackEnd, el cual es el encargado de realizar las consultas en la Base de datos y retornar una respuesta.



```
index.controllers.js | src\index.js | routes\index.js

//Datos del docente
const datosDocente = async (req,res)=>{
    let idDocente = req.params.id;
    const respDatosDocente = await  pool.query('SELECT nombredocente,apepatdocente,apematdocente FROM docente WHERE iddocente = $1',[idDocente]);
    res.status(200).json({
        ok:true,
        respDatosDocente
    });
}

//Horario del docente
const horarioDocente = async (req,res)=> {
    let idDocente = req.params.id;
    const respHorarioDocente = await pool.query('SELECT idedificiohorario,numeroaulahorario,horainicio,horafin,clavemateriahorario FROM horariogrupo WHERE iddocentehorario = $1',[idDocente]);

    res.status(200).json({
        ok:true,
        respHorarioDocente
    });
}

//Datos del alumno
const datosAlumno = async (req,res)=>{
    let numControl = req.params.id;
    const respDatosAlumno = await  pool.query("SELECT nombrealumno,apepatalumno,apematalumno FROM alumno WHERE nocontrol = $1",[numControl]);
    res.status(200).json({
        ok:true,
        respDatosAlumno
    });
}

//Horario del alumno
const horarioAlumno = async (req,res)=> {
    let numControl =  req.params.id;
    const respHorarioAlumno = await pool.query('SELECT idedificiohorario,numeroaulahorario,horainicio,horafin,clavemateriahorario FROM horariogrupo WHERE nocontrolhorario = $1',[numControl]);
    res.status(200).json({
        ok:true,
        respHorarioAlumno
    });
```

En cuestión del FrontEnd, contamos con dos funciones principales, las cuales son las encargadas de solicitar las peticiones al BackEnd y posteriormente hacer las respectivas validaciones.



```
api.service.ts U | principal-component.component.ts U | principal-component.component.html U

buscarAlumno = () =>{
  console.clear();
  this._apiService.consultarAlumno(this.numControl).subscribe((resp:any) =>{

    //Validamos si es que hubo respuesta de la BD
    if (resp.respDatosAlumno.rowCount == 0) {
      console.error("%cAlumno no registrado","font-size:20px");
    }else{
      console.log("%cAlumno Encontrado","color:green;font-size:20px");
      console.log(resp.respDatosAlumno.rows);
      this._apiService.localizarAlumno(this.numControl).subscribe((resp:any) =>{
        console.log(resp.respHorarioAlumno.rows);
        //Guardamos las filas que retorna la bd
        let valores = resp.respHorarioAlumno.rows;

        //Recorremos las rows para con las horas determinar la ubicación
        for (const valor in valores) {
          let horainicio = valores[valor]["horainicio"];
          let horafin = valores[valor]["horafin"];

          //Validamos que la hora de búsqueda esté en el rango
          if ((this.hora >= horainicio) && (this.hora <= horafin)) {
            console.log(`%cEl alumno se encuentra en el edificio ${valores[valor]["idedificiohorario"]} aula ${valores[valor]["numeroaulahorario"]}`,"color:yellow;font-size:20px");
          }
        }
      });
    }
  });
}

buscarProfesor = () =>{
  console.clear();
  this._apiService.consultarDocente(this.idDocente).subscribe((resp:any) =>{
    //Validamos si es que hubo respuesta de la BD
    if (resp.respDatosDocente.rowCount == 0) {
      console.error("%cProfesor no registrado","font-size:20px");
    }else{
      console.log("%cProfesor Encontrado","color:green;font-size:20px");
      console.log(resp.respDatosDocente.rows);
      this._apiService.localizarDocente(this.idDocente).subscribe((resp:any) =>{
        console.log(resp.respHorarioDocente.rows);
        //Guardamos las filas que retorna la bd
        let valores = resp.respHorarioDocente.rows;

        //Recorremos las rows para con las horas determinar la ubicación
        for (const valor in valores) {
          let horainicio = valores[valor]["horainicio"];
          let horafin = valores[valor]["horafin"];

          //Validamos que la hora de búsqueda esté en el rango
          if ((this.hora >= horainicio) && (this.hora <= horafin)) {
            console.log(`%cEl docente se encuentra en el edificio ${valores[valor]["idedificiohorario"]} aula ${valores[valor]["numeroaulahorario"]}`,"color:yellow;font-size:20px");
          }
        }
      });
    }
  });
}

}
```

Esta será el API, la cual es la encargada de conectar ambos servicios, tanto el BackEnd como el FronEnd.

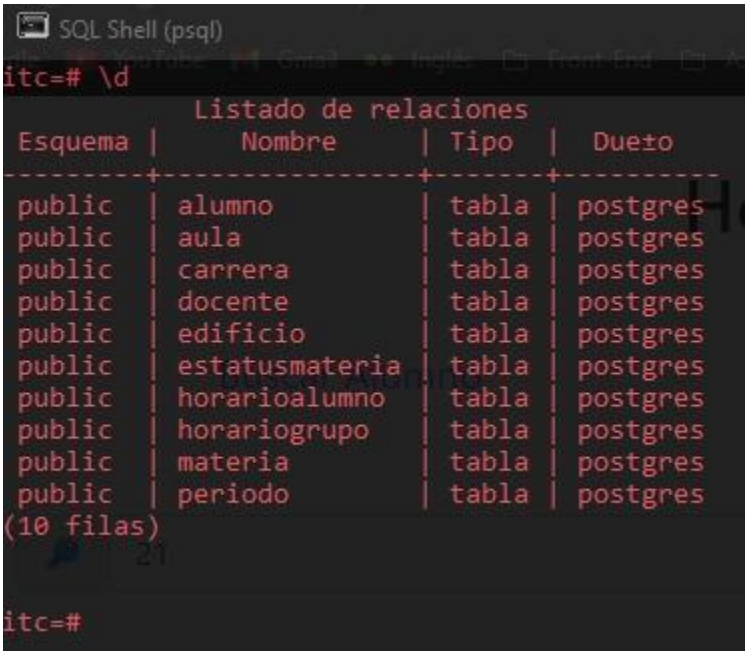
api.service.ts U ● | principal-component.component.ts U | principal-component.component.html U

src > app > api.service.ts > ApiService

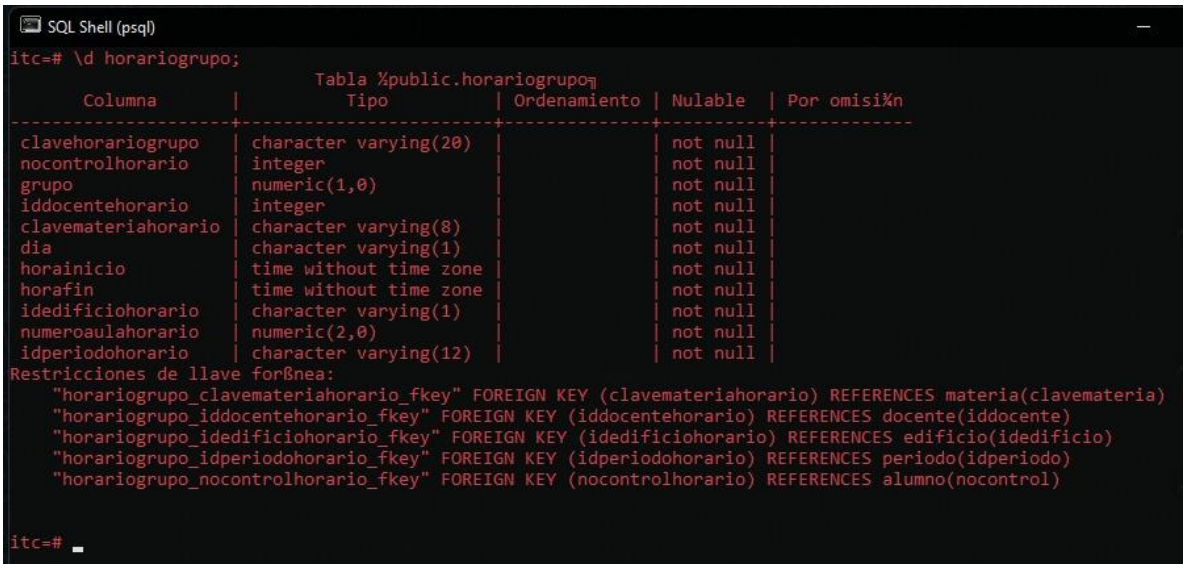
```
1  import { Injectable } from '@angular/core';
2  import {HttpClient} from '@angular/common/http';
3
4  @Injectable({
5    providedIn: 'root'
6  })
7  export class ApiService {
8    public url;
9
10   constructor(private _http:HttpClient) {
11     this.url = "http://localhost:3000";
12   }
13
14   consultarAlumno = (id:any) =>{
15     let url = `${this.url}/datos/alumno/${id}`;
16     return this._http.get(url);
17   }
18
19   consultarDocente = (id:any) =>{
20     let url = `${this.url}/datos/docente/${id}`;
21     return this._http.get(url);
22   }
23
24   localizarAlumno = (id:any) =>{
25     let url = `${this.url}/buscar/alumno/${id}`;
26     return this._http.get(url);
27   }
28
29   localizarDocente = (id:any,) =>{
30     let url = `${this.url}/buscar/docente/${id}`;
31     return this._http.get(url);
32   }
33 }
```

Registros.

Aquí encontramos las tablas con las que cuenta la Base de datos.



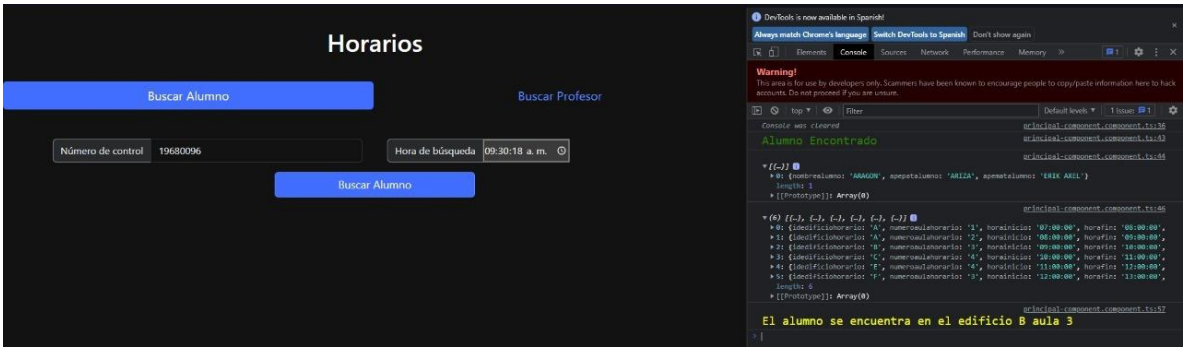
La tabla horariogrupo, contendrá información sobre los horarios, la cual está relacionada mediante llaves foráneas con las tablas pertinentes.



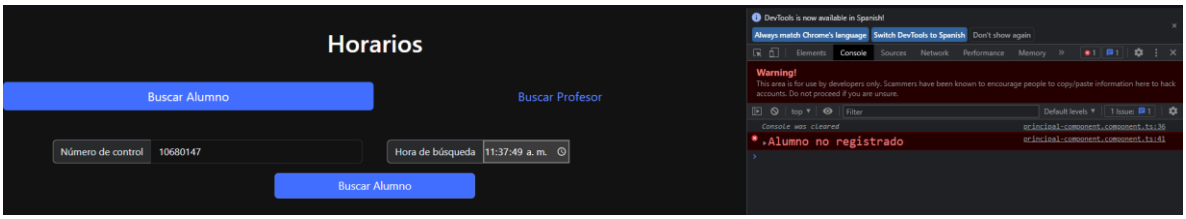
Registros de la tabla horariogrupo.

Función de las salidas.

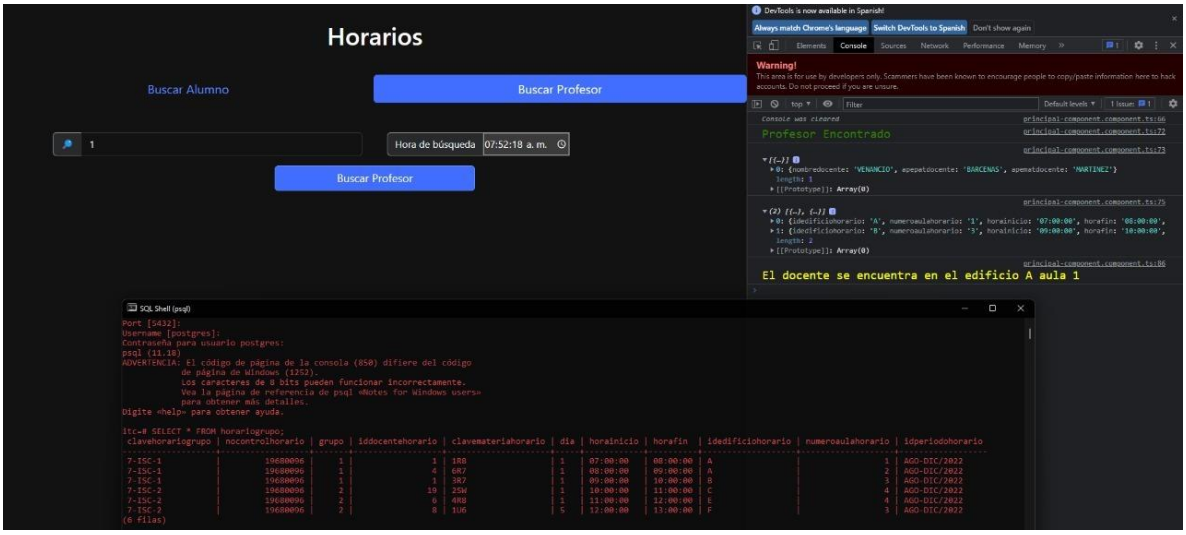
Prueba exitosa de la búsqueda Alumno.



Prueba fallidade la búsqueda Alumno.



Prueba exitosa de la búsqueda Docente.



Prueba fallida de la búsqueda Docente.