



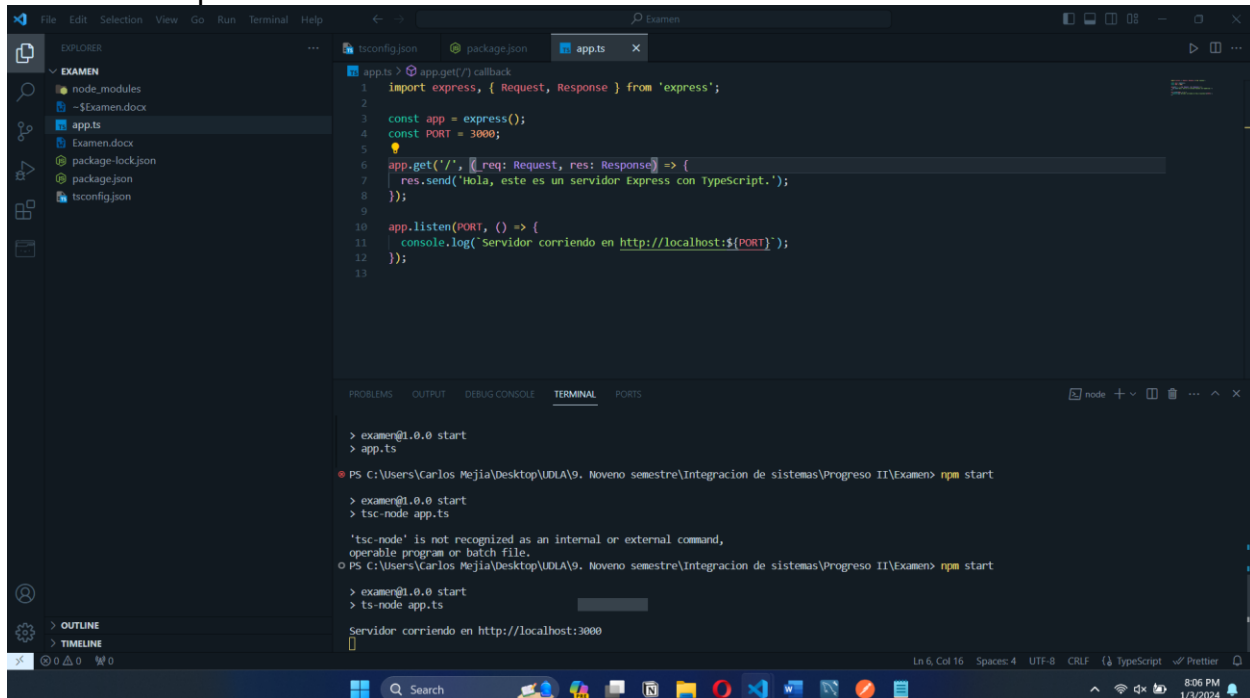
Facultad de Ingeniería y Ciencias Aplicadas
Integración de sistemas
2023-20

Carlos Andrés Mejía Hidalgo

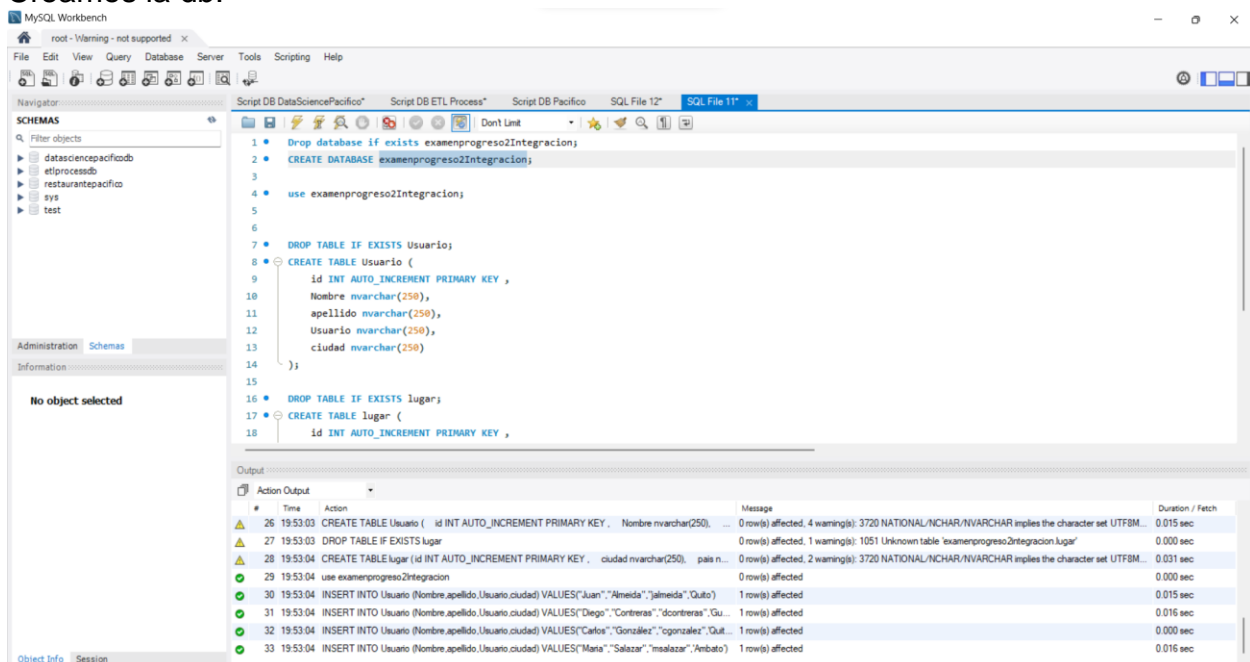
Fecha de entrega:
01/04/2024

Examen

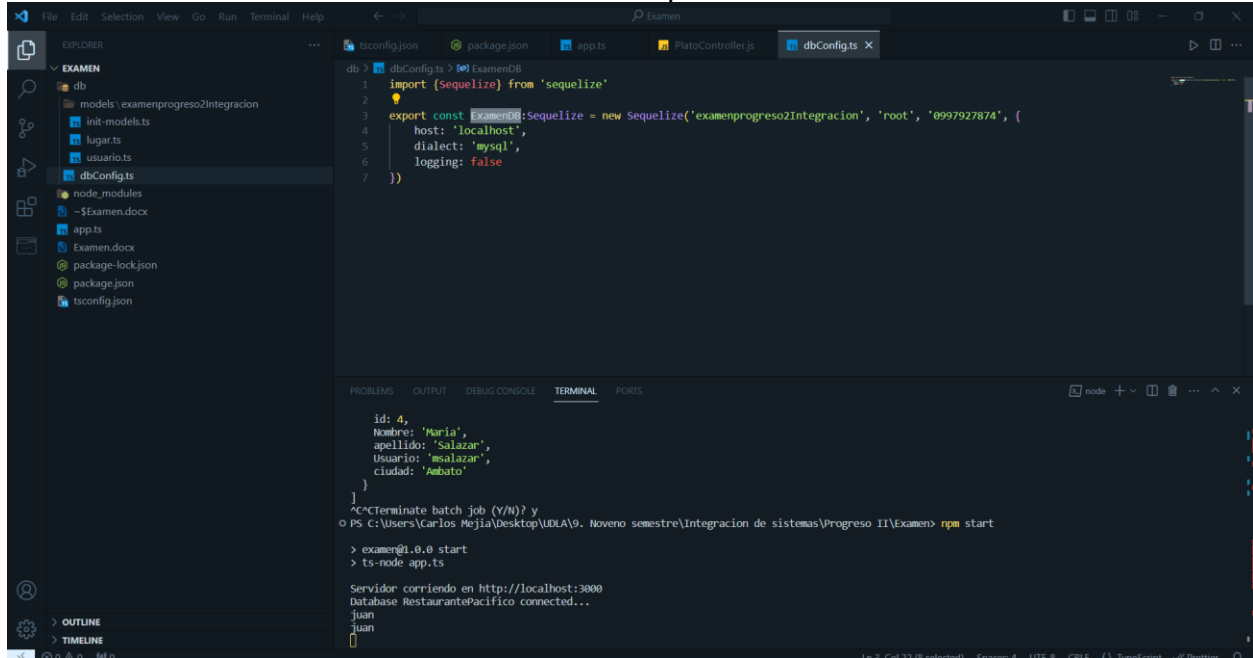
Creamos la api en node:



Creamos la db:



Creamos la conexión a la db con la librería sequelize



The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying the project structure. The main editor window shows the `dbConfig.ts` file with the following code:

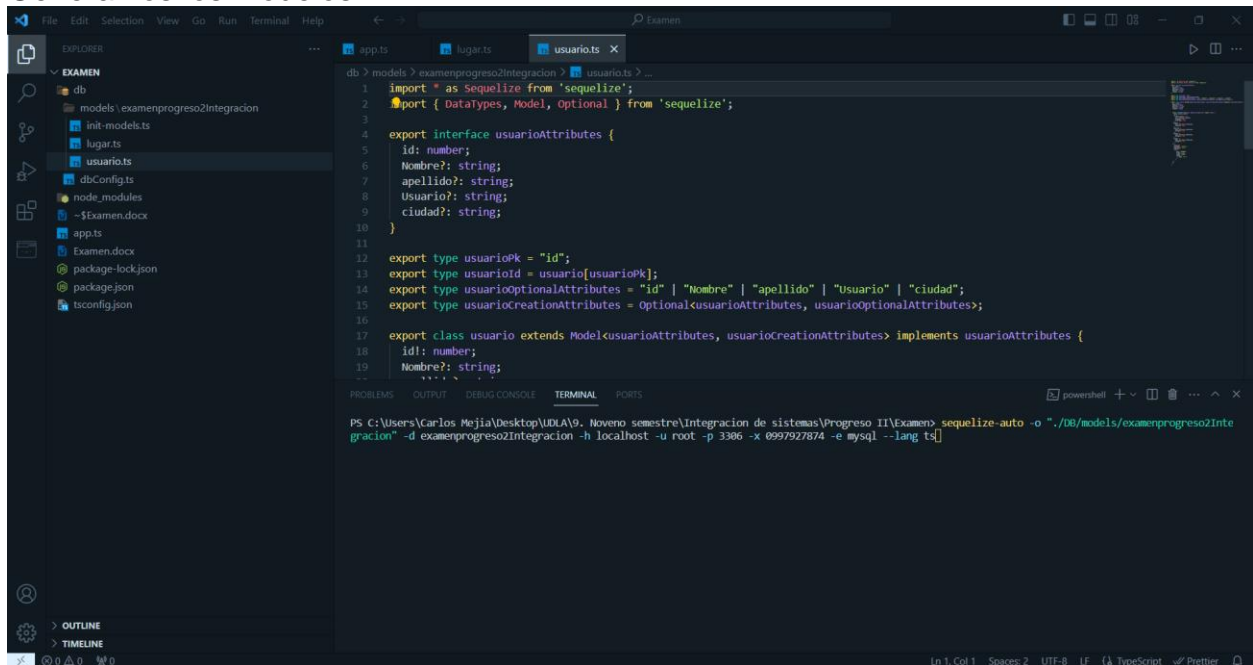
```
1 import { Sequelize } from 'sequelize'
2
3 export const ExamDB: Sequelize = new Sequelize('examenprogreso2Integracion', 'root', '0997927874', {
4   host: 'localhost',
5   dialect: 'mysql',
6   logging: false
7 })
```

The terminal window at the bottom shows the output of running the application:

```
id: 4,
Nombre: 'Maria',
apellido: 'Salazar',
Usuario: 'msalazar',
ciudad: 'Ambato'
}
^C^CTerminate batch job (Y/N)? y
PS C:\Users\Carlos Mejia\Desktop\UDLA\9. Noveno semestre\Integracion de sistemas\Progreso II\Examen> npm start
> examen@1.0.0 start
> ts-node app.ts

Servidor corriendo en http://localhost:3000
Database RestaurantPacífico connected...
juan
juan
}
```

Generamos los modelos:



The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying the project structure. The main editor window shows the `usuario.ts` file with the following code:

```
1 import * as Sequelize from 'sequelize';
2 import { DataTypes, Model, Optional } from 'sequelize';
3
4 export interface usuarioAttributes {
5   id: number;
6   Nombre?: string;
7   apellido?: string;
8   Usuario?: string;
9   ciudad?: string;
10 }
11
12 export type usuarioPk = "id";
13 export type usuarioId = usuario[usuarioPk];
14 export type usuarioOptionalAttributes = "id" | "Nombre" | "apellido" | "Usuario" | "ciudad";
15 export type usuarioCreationAttributes = Optional<usuarioAttributes, usuarioOptionalAttributes>;
16
17 export class usuario extends Model<usuarioAttributes, usuarioCreationAttributes> implements usuarioAttributes {
18   id!: number;
19   Nombre?: string;
20   ...
21 }
```

The terminal window at the bottom shows the output of running the application:

```
PS C:\Users\Carlos Mejia\Desktop\UDLA\9. Noveno semestre\Integracion de sistemas\Progreso II\Examen> sequelize-auto -o ".\DB\models\examenprogreso2Integracion" -d examenprogreso2Integracion -h localhost -u root -p 3306 -x 0997927874 -e mysql --lang ts
```

Creamos un método que recibe el usuario y realiza la petición a la api externa:

```

app.get('/api/:usuarioPar', async (req: Request, res: Response) => {

  try{
    const{usuarioPar}=req.params
    let usuario:any
    await modelExamen.usuario.findOne({raw: true, where: { 'Usuario': usuarioPar } }).then((res:any)=>{
      usuario=res
    })
    if(usuario){
      await llamadaApiGeo(usuario.ciudad)

    }

    res.json(usuario);

  }catch (error:any) {
    console.error('Error al realizar la petición:', error.message);
    res.json('Ocurrio error');
  }

});

```

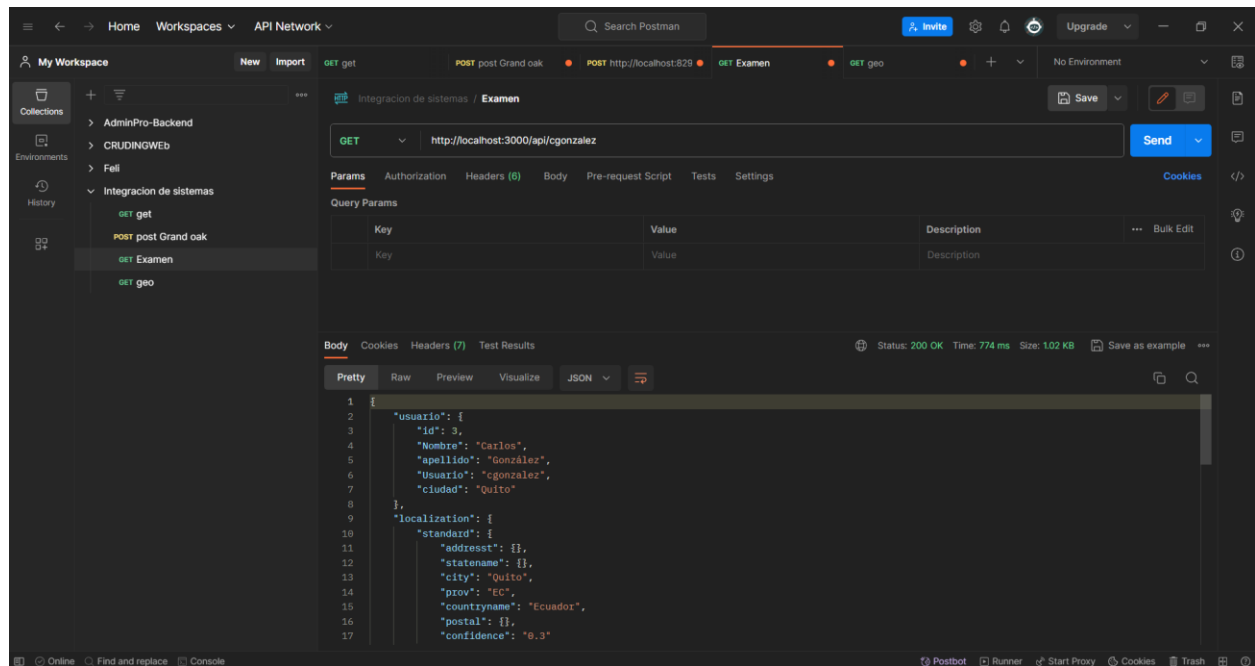
Creamos una función para la llamada a la api externa:

```

33
34 const llamadaApiGeo=async(ciudad:string)=>{
35   try {
36     let lugar:any;
37     const url = `https://geocode.xyz/${ciudad}?json=1&auth=682343438346333111693x45401`; // Reemplaza con la URL de la
      API externa
38
39     const response = await fetch(url);
40
41     if (!response.ok) {
42       throw new Error('Error de red: ${response.status}');
43     }
44
45     const data = await response.json();
46
47     await modelExamen.lugar.findOne({raw: true, where: { 'ciudad': data.standard.city } }).then((resp:any)=>{
48       lugar=resp
49     })
50     console.log(lugar)
51
52     if(lugar === null){
53       await modelExamen.lugar.create({ 'ciudad': ciudad, 'pais': data.standard.countryname }).then()
54     }
55
56     return data;
57   } catch (error:any) {
58     console.error('Error al realizar la petición:', error.message);
59   }
60 }
61

```

Resultado:



Link github: <https://github.com/CralosMejia/ExamenprogresollIntegracion>