Taller Unidad 2 Backend

Leidy Carolina Botina Botina

Fredery Duvan Buesaquillo Guacales

Daniel Guancha

Cristhian Alejandro Escobar Diaz

Universidad de Nariño

Ingeniería de sistemas

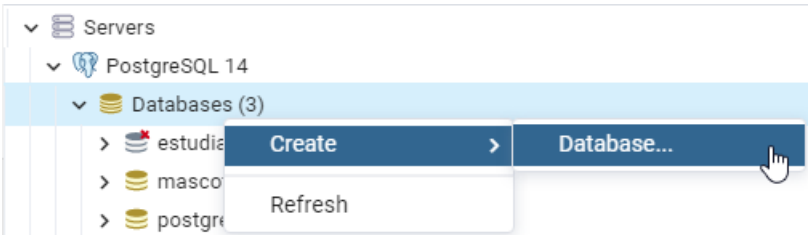
Diplomado de actualización en nuevas tecnologías para el desarrollo de Software.

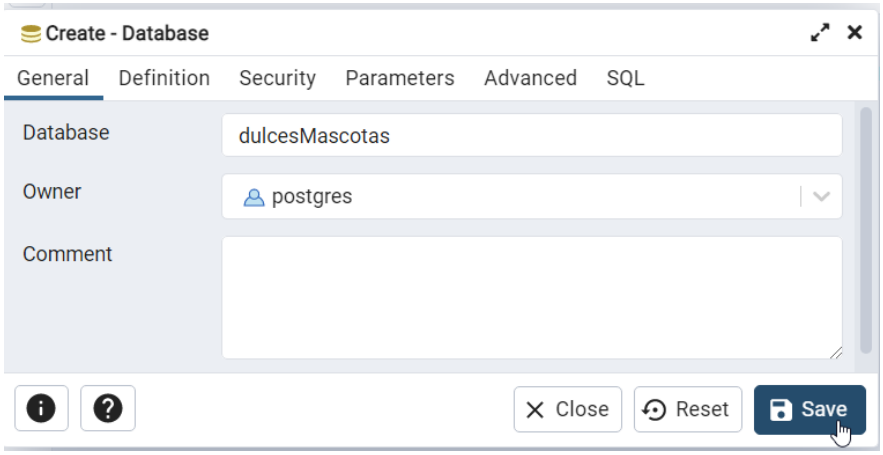
San Juan de Pasto

2022

1. Creación de bases de datos postgres.

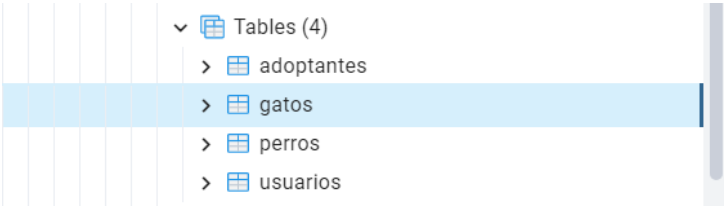
1.1. Nos dirigimos a PgAdmin y creamos ahí nuestra base de datos con nombre: dulcesMascotas.





1.2. Creación de tablas mediante código en el apartado index.js sincronizando con gestor de base de datos.

```
index.js
import express from 'express';
const app = express();
import router from './src/Routes/routes.js';
import { sequelize } from './src/Database/Database.js';
import cors from 'cors';
import './src/Models/Adoptante.js';
import './src/Models/Gatos.js';
import './src/Models/Perros.js';

const testDb = async () => {
  try {
    await sequelize.sync();
    console.log('Connection has been established successfully.');
    app.use(cors());
    app.use(express.json({ extended: false }));
    app.use(router);

    app.set('port', 5000);
    app.listen(app.get('port'));
    console.log(`Servidor Activo en Puerto ${app.get('port')}`);
  } catch (error) {
    console.error('Unable to connect to the database '+error);
  }
}

testDb();
```



2. Desarrollo de la aplicación Backend.

2.1. Instalación de bcrypt para cifrar contraseñas.

```
PS D:\Diplomado\Modulo Uno\dulcesMacotasBk> npm i bcrypt
```

2.2. Creación del package.json mediante el comando **npm init -y.**

2

```
PS D:\Diplomado\Modulo Uno\TallerUnidad2> cd .\macotasBk\
PS D:\Diplomado\Modulo Uno\TallerUnidad2\macotasBk> npm init -y
```

2.3. Instalación de express y postgres mediante el comando **npm install express pg.**

```
PS D:\Diplomado\Modulo Uno\TallerUnidad2\macotasBk> npm install express pg
```

2.4. Instalación de nodemon mediante comando **npm install nodemos -D.**

```
PS D:\Diplomado\Modulo Uno\TallerUnidad2\macotasBk> npm install nodemon -D
```

2.5. Instalación sequalize mediante comando **npm install –save sequelize.**

```
PS D:\Diplomado\Modulo Uno\TallerUnidad2\macotasBk> npm install --save sequelize
```

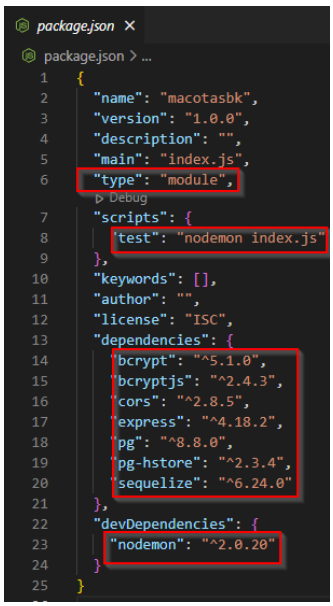
2.6. Instalación pg-hstore mediante comando **npm install –save pg-hstore.**

```
PS D:\Diplomado\Modulo Uno\TallerUnidad2\macotasBk> npm install --save pg-hstore
```

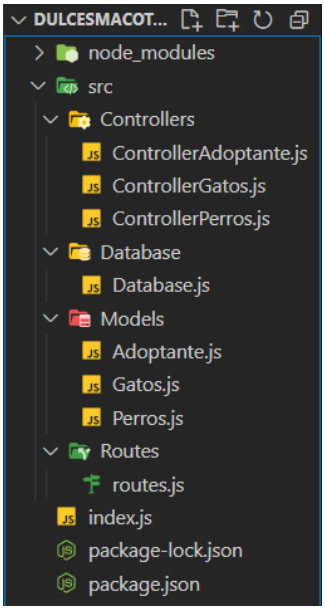
2.7. Instalación cors mediante comando **npm install cors.**

```
PS D:\Diplomado\Modulo Uno\TallerUnidad2\macotasBk> npm install cors
```

2.8. Configuración completa del archivo **packege.json.**



2.9. Creación de la estructura del proyecto con sus respetivos controladores, módulos y rutas.



2.10. Implementación archivo **ControllerUsuario.js** con su respectivo get, post, login, put, y delete Usuario.

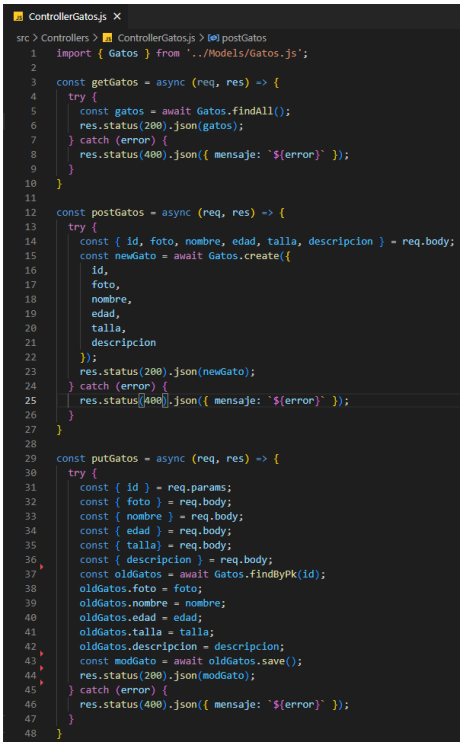
ControllerUsuario.js M

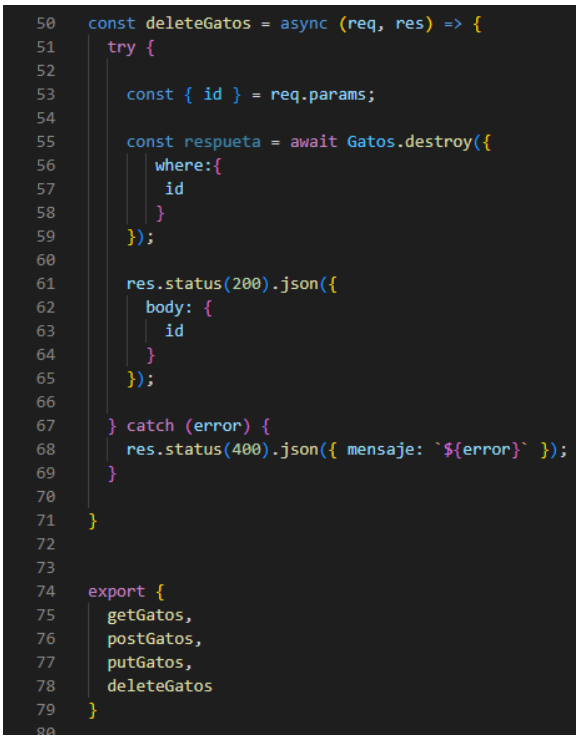
src > Controllers > ControllerUsuario.js > postusuario

```
import { Usuario } from '../Models/Usuarios.js';
import  bcrypt  from 'bcryptjs';
//const { bcrypt } = bcrypt;

const getUsuario = async (req, res) => {
  try {
    const usuarios = await Usuario.findAll();
    res.status(200).json(usuarios);
  } catch (error) {
    res.status(400).json({ mensaje: `${error}` });
  }

}

const postusuario = async (req, res) => {
  try {
    const { id } = req.body;
    const { nombre } = req.body;
    const { correo } = req.body;
    const { contraseña } = req.body;
    const newUsuario = await Usuario.create({
      id,
      nombre,
      correo,
      contraseña: await bcrypt.hash(contraseña, 10)
    });

    res.status(200).json(newUsuario);
  } catch (error) {
    res.status(400).json({ mensaje: `${error}` });
  }
}

const login = async (req, res) => {
  try {

    const { contraseña } = req.body;
    const { correo } = req.body;

    const user = await Usuario.findOne({ where: {correo: correo }});
    const isSame = await bcrypt.compare(contraseña, user.contraseña);
    if(isSame){
      res.status(200).json(user);
    }
  } catch (error) {
    res.status(400).json({ mensaje: `${error}` });
  }
}
```

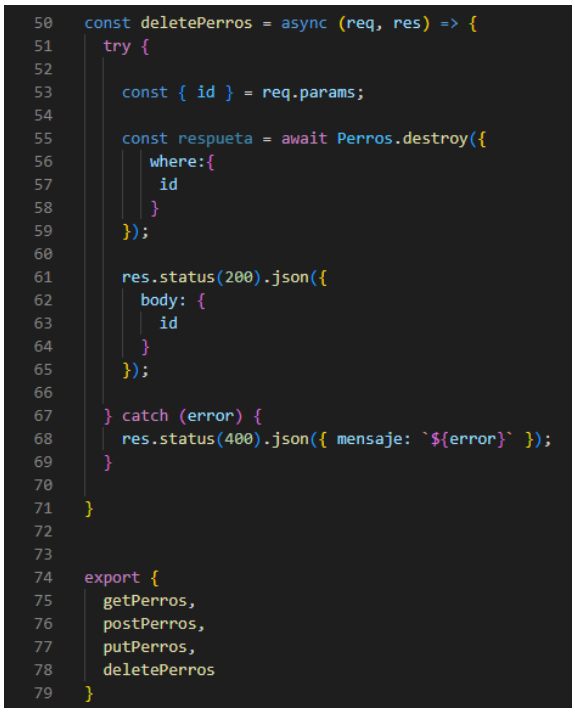
```
const putUsuario = async (req, res) => {
  try {

    const { id } = req.params;
    const { nombre } = req.body;
    const { correo } = req.body;
    const { contraseña } = req.body;

    const oldUsuario = await Usuario.findByPk(id);

    oldUsuario.nombre=nombre;
    oldUsuario.correo=correo;
    oldUsuario.contraseña=contraseña;
    const modUsuario = await oldUsuario.save();
    res.status(200).json(modUsuario);

  } catch (error) {
    res.status(400).json({ mensaje: `${error}` });
  }
}

const deleteUsuario = async (req, res) => {
  try {
    const { id } = req.params;
    const respueta = await Usuario.destroy({
      where:{
       id
      }
    });
    res.status(200).json({
      body: {
        id
      }
    });
  } catch (error) {
    res.status(400).json({ mensaje: `${error}` });
  }
}

export {
  getUsuario,
  postusuario,
  putUsuario,
  deleteUsuario,
  login
}
```

2.11. Implementación archivo **ControllerGatos.js** con su respectivo get, post, login, put, y delete Gatos**.**





2.12. Implementación archivo **ControllerPerros.js** con su respectivo get, post, login, put, y delete Perros.



2.13. Implementación archivo **Database.js.**

Database.js

src > Database > Database.js > ...

```
import  Sequelize  from "sequelize";

const sequelize = new Sequelize('dulcesMascotas', 'postgres', '12345678', {
  host: 'localhost',
  dialect: 'postgres'
});

export {
  sequelize
}
```

2.14. Implementación archivo **Usuarios.js.**

Usuarios.js

src > Models > Usuarios.js > ...

```
import { DataTypes }  from 'sequelize';
import { sequelize } from '../Database/Database.js';

const Usuario = sequelize.define('usuario', {
  id: {
    type: DataTypes.INTEGER,
    primaryKey: true
  },
  nombre: {
    type: DataTypes.STRING
  },
  correo: {
    type: DataTypes.STRING
  },
  contraseña: {
    type: DataTypes.STRING,
    unique: true
  }
});

export {
  Usuario
}
```

2.15. Implementación archivo **Adoptante.js.**

Adoptante.js

src > Models > Adoptante.js > ...

```
import { DataTypes }  from 'sequelize';
import { sequelize } from '../Database/Database.js';

const Adoptante = sequelize.define('adoptante', {
  id: {
    type: DataTypes.INTEGER,
    primaryKey: true
  },
  nombre: {
    type: DataTypes.STRING
  },
  correo: {
    type: DataTypes.STRING
  },
  telefono: {
    type: DataTypes.STRING
  }
});

export {
  Adoptante
}
```

2.16. Implementación archivo **Perros.js.**

Perros.js

src > Models > Perros.js > ...

```
import { DataTypes }  from 'sequelize';
import { sequelize } from '../Database/Database.js';

const Perros = sequelize.define('perros', {
  id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    autoIncrement: true
  },
  foto:{
    type: DataTypes.STRING
  },
  nombre: {
    type: DataTypes.STRING
  },
  edad: {
    type: DataTypes.STRING,
  },
  talla: {
    type: DataTypes.STRING
  },
  descripcion: {
    type: DataTypes.STRING
  }
});

export {
  Perros
}
```

2.17. Implementación archivo **Gatos.js.**

Gatos.js

src > Models > Gatos.js > ...

```
import { DataTypes }  from 'sequelize';
import { sequelize } from '../Database/Database.js';

const Gatos = sequelize.define('gatos', {
  id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    autoIncrement: true
  },
  foto:{
    type: DataTypes.STRING
  },
  nombre: {
    type: DataTypes.STRING
  },
  edad: {
    type: DataTypes.STRING,
  },
  talla: {
    type: DataTypes.STRING
  },
  descripcion: {
    type: DataTypes.STRING
  }
});

export {
  Gatos
}
```

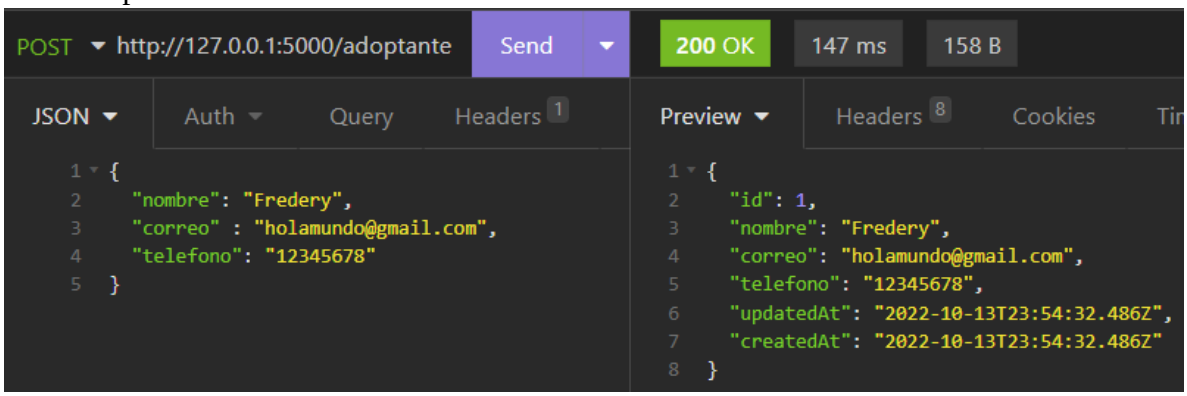
2.18. Implementación archivo **routes.js.**

routes.js

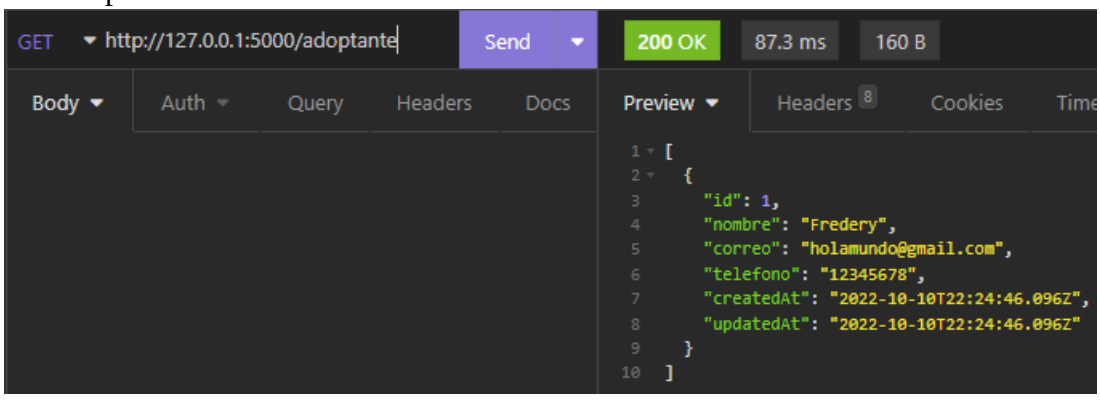
```
import  {Router} from 'express';
import { deleteAdoptante, getAdoptante, postAdoptante, putAdoptante } from '../Controllers/ControllerAdoptante.js';
import { getGatos, postGatos, putGatos, deleteGatos } from '../Controllers/ControllerGatos.js';
import { getPerros, postPerros, putPerros, deletePerros } from '../Controllers/ControllerPerros.js';
import { postusuario, login } from '../Controllers/ControllerUsuario.js';

const router=Router();

router.get('/',(req,res)=>{
  res.send("Hola estamos en Index");
});

router.post('/usuario',postusuario);
router.post('/login',login);

router.get('/gatos',getGatos);

router.post('/gatos',postGatos);

router.put('/gatos/:id',putGatos);

router.delete('/gatos/:id',deleteGatos);


router.get('/perros',getPerros);

router.post('/perros',postPerros);

router.put('/perros/:id',putPerros);

router.delete('/perros/:id',deletePerros);


router.get('/adoptante',getAdoptante);

router.post('/adoptante',postAdoptante);

router.put('/adoptante/:id',putAdoptante);

router.delete('/adoptante/:id',deleteAdoptante);

export default router;
```

3. VERIFICACION DE OPERACIONES A TRAVEZ DE INSOMNIA.

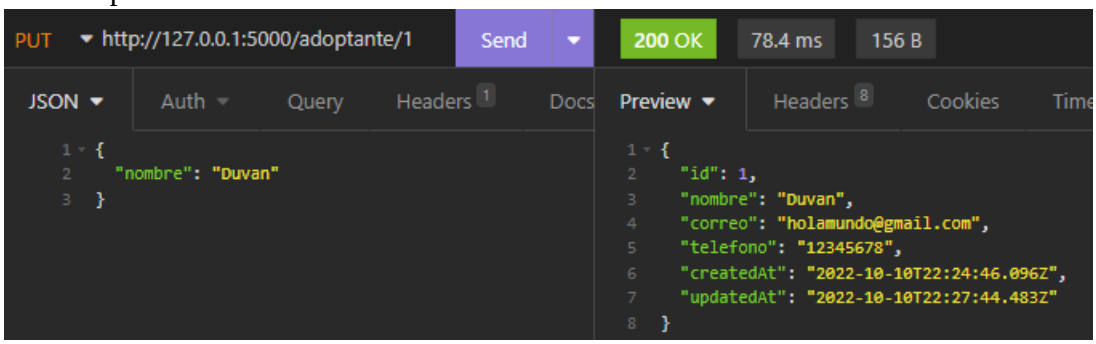
3.1. Post adoptante.



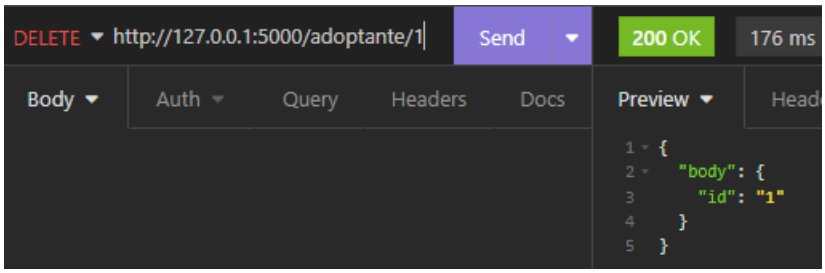
3.2. Get adoptante.



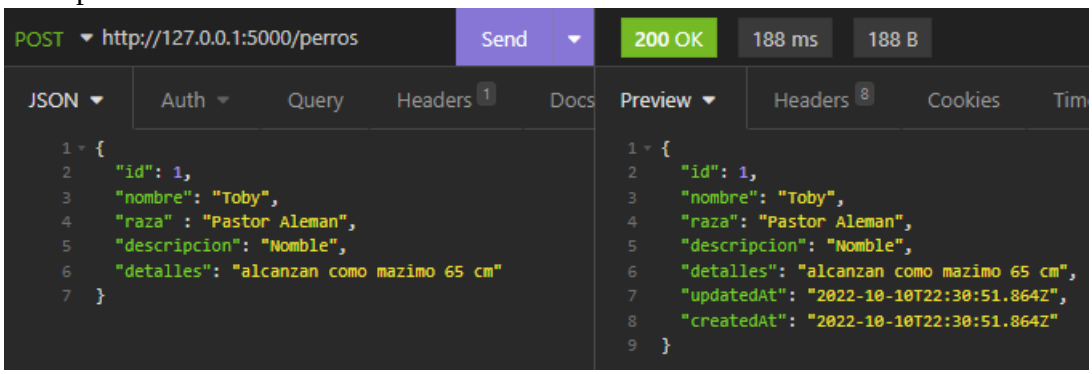
3.3. Put adoptante.



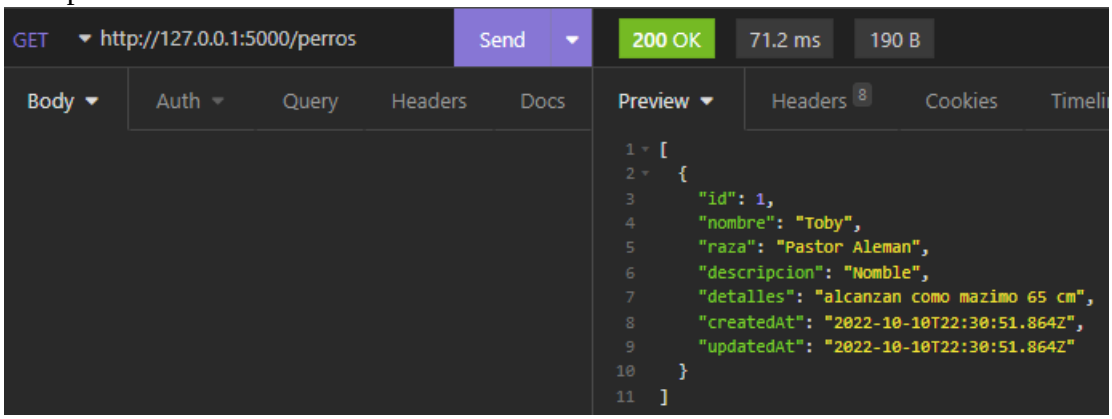
3.4. Delete adoptante.



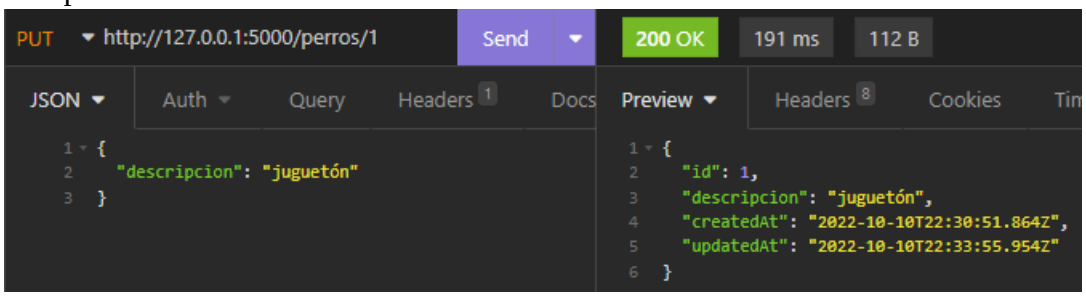
3.5. Post perros.



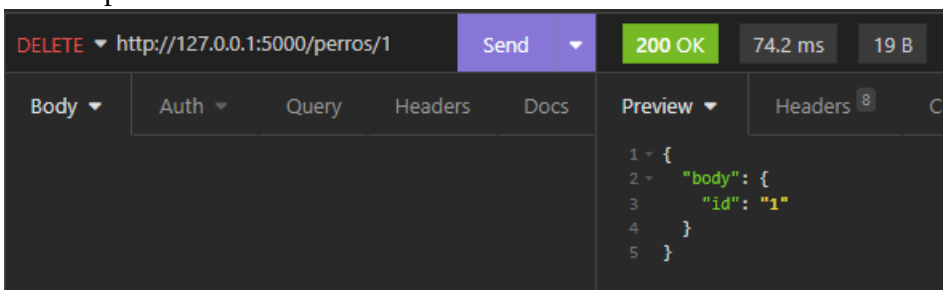
3.6. Get perros.



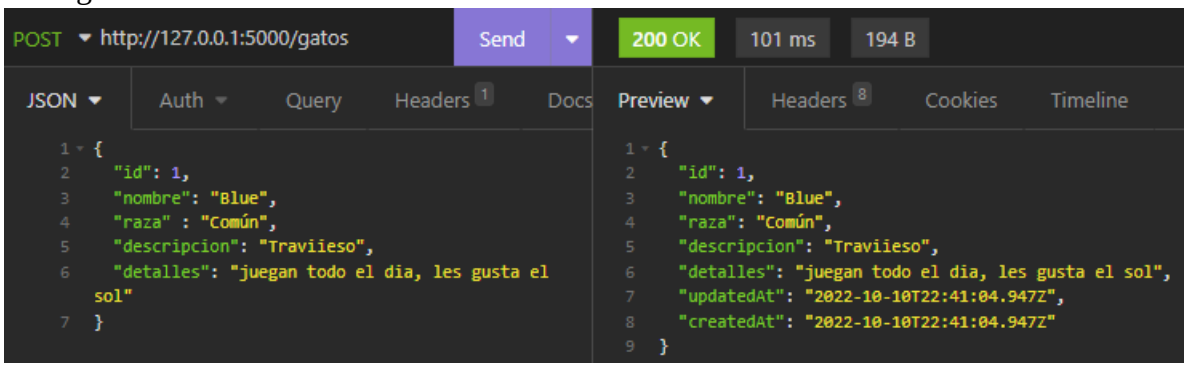
3.7. Put perros.



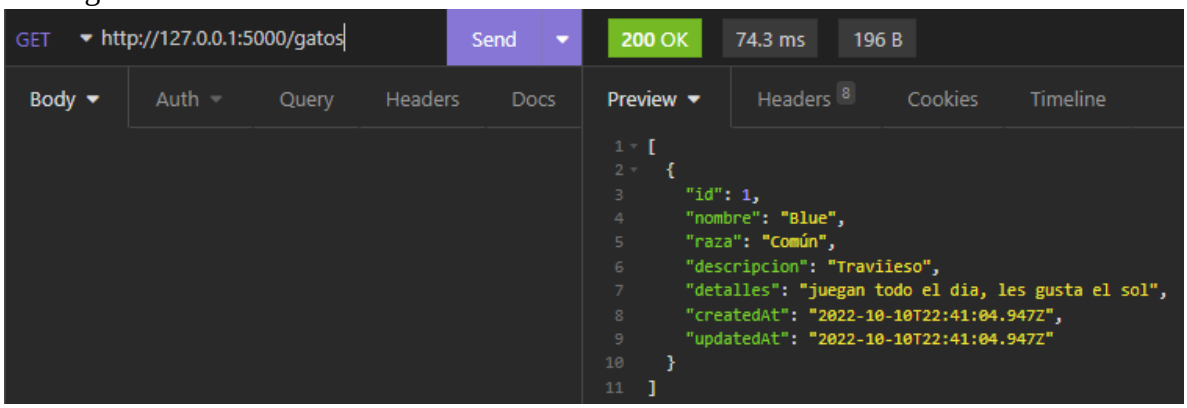
3.8. Delete perros.



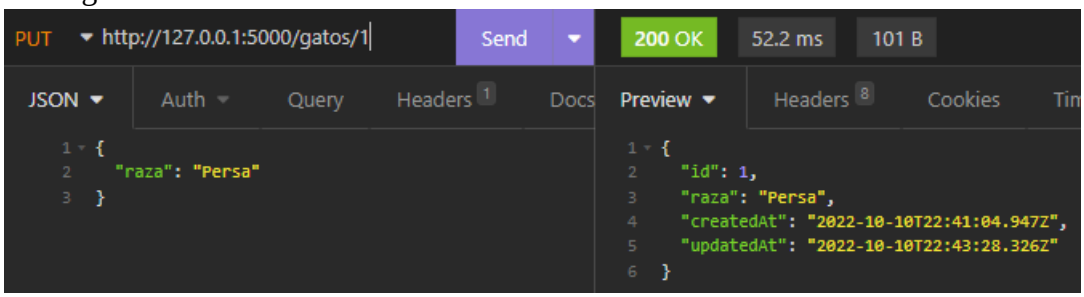
3.9. Post gatos.



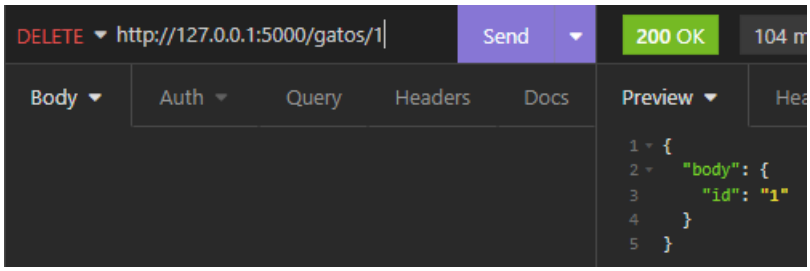
3.10. Get gatos.



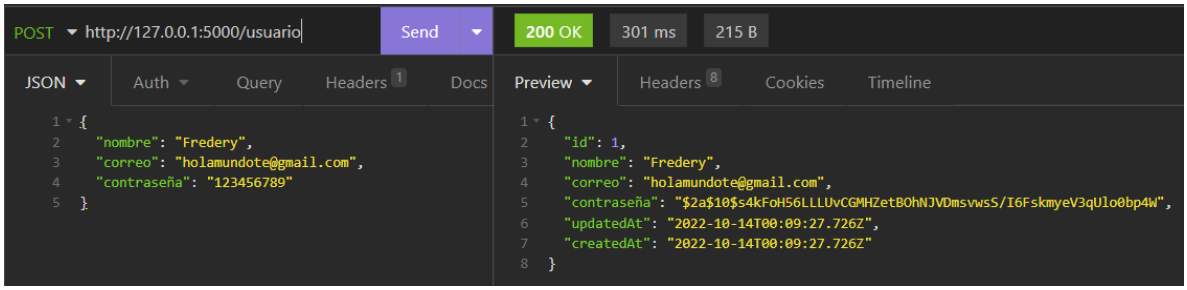
3.11. Put gatos.



3.12. Delete gatos.



3.13. Post usuario.



3.14. Login usuario.