Informe Backend Fruver la 17

Presentado por:

Luis Fernando Oviedo Dominguez

Presentado a:

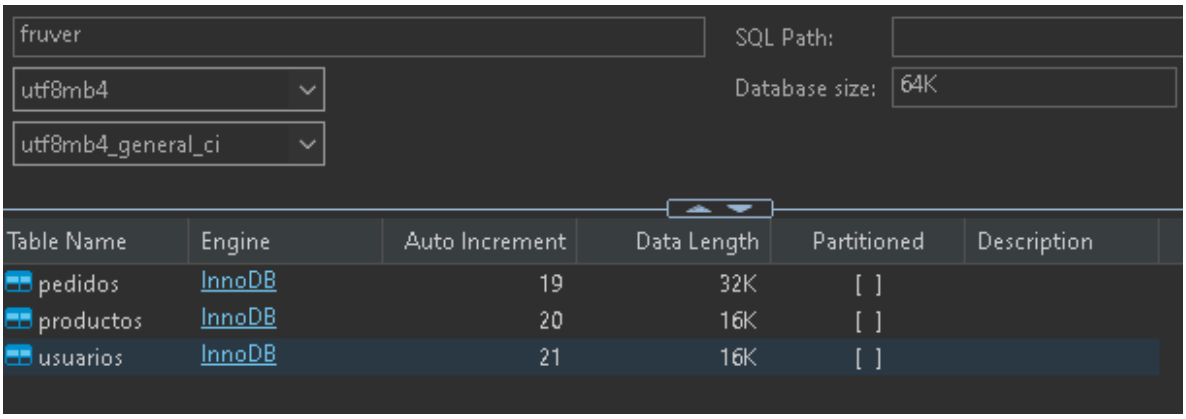
Mg. Euler Vicente Aux Revelo

Universidad de Nariño

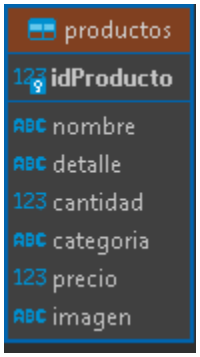
2023

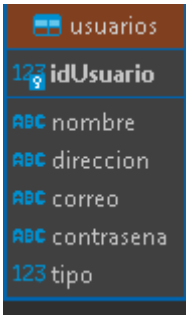
Pasos para la construcción del back_end encaminado hacia la frutería la 17

Creación de la base de datos desde MariaDB con las tablas pedidos, productos y usuarios



La tabla pedidos, productos y usuarios tienen la siguiente estructura





Posteriormente se procede a crear un proyecto donde se instalan las bibliotecas para trabajar con el gestor de paquetes de NodeJS usando el comando **npm install**, una vez realizada la instalación se instala ExpressJS con el comando **npm install express.**

Paso a seguir, se crea la conexión a la base de datos, en mi caso particular el servicio de MySQL se ejecuta desde el puerto **3307**

```
import { Sequelize } from "sequelize";
const sequelize = new Sequelize('fruver', 'root', '', {
    host: 'localhost',
    dialect: "mysql",
    port: 3307
});
export{sequelize}
```

Después es necesario crear un archivo server.js con el fin de establecer instrucciones para asignar un puerto de conexión y establecer comunicación con la base de datos configurada previamente

```
const testDb = async () => {
  try {
    await sequelize.sync();
    console.log(`Conexion realizada con éxito`);
    //Correr Servicio por puerto 3000
    app.listen(app.get("port"), () => {
      console.log(`Servidor Escuchando por puerto ${app.get("port")}`);
    });
  } catch (error) {
    console.log(`Error al realizar conexión ${error}`);
  }
};

testDb();
```

También es necesario configurar un manejador de rutas con el fin de establecer cada ruta necesaria para la realización de las operaciones de cada método creado

Rutas para el manejo de productos

```
router.get("/productos", getProductos);
router.get("/productos/:idProducto", getProducto);
router.post("/productos",postProductos);
router.put("/productos/:idProducto",putProductos);
router.delete("/productos/:idProducto", deleteProductos);
```

Rutas para el manejo de los pedidos

```
router.get("/pedidos", getPedidos);
router.get("/pedidos/:idPedido", getPedido);
router.post("/pedidos",postPedidos);
router.put("/pedidos/:idPedido",putPedido);
router.delete("/pedidos/:idPedido", deletePedido);
```

Rutas para el manejo de usuarios

```
router.get("/registro/usuarios/:idUsuario", getUsuario);
router.post("/registro/usuarios",postUsuarios);
router.post("/acceder",postUsuarioLogin);
```

Además, se crean modelos de datos con el fin de definir la estructura de cada tabla de la base de datos, es asi como se crea el modelo pedidos, productos y usuarios

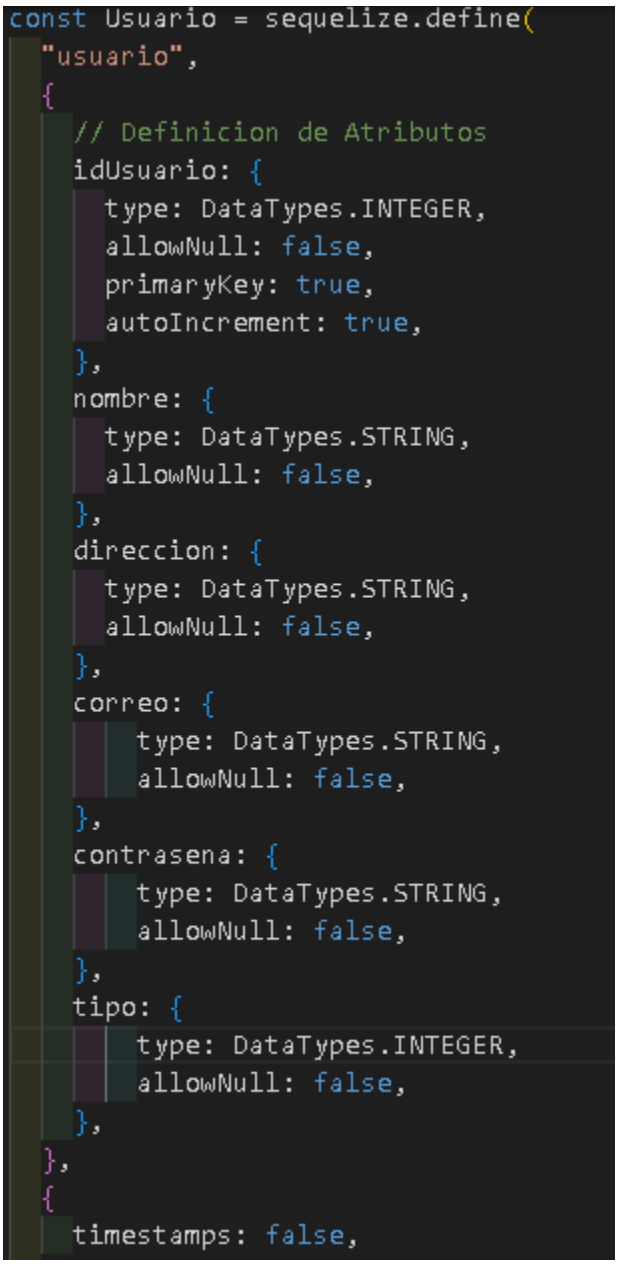
Modelos pedidos

```
const Pedido = sequelize.define(
  "pedido",
  {
    // Definicion de Atributos
    idPedido: {
      type: DataTypes.INTEGER,
      allowNull: false,
      primaryKey: true,
      autoIncrement: true,
    },
    detalle: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    cantidad: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    fecha: {
      type: DataTypes.DATE,
      allowNull: false,
    }
  },
  {
    timestamps: false,
  }
);
```

Modelo productos

```
const Producto = sequelize.define(
  "producto",
  {
    // Definicion de Atributos
    idProducto: {
      type: DataTypes.INTEGER,
      allowNull: false,
      primaryKey: true,
      autoIncrement: true,
    },
    nombre: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    detalle: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    cantidad: {
      type: DataTypes.INTEGER,
      allowNull: false,
    },
    categoria: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    precio: {
      type: DataTypes.DECIMAL,
      allowNull: false,
    },
    imagen: {
      type: DataTypes.STRING,
    },
```

Modelo usuarios



```
const Usuario = sequelize.define(
  "usuario",
  {
    // Definicion de Atributos
    idUsuario: {
      type: DataTypes.INTEGER,
      allowNull: false,
      primaryKey: true,
      autoIncrement: true,
    },
    nombre: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    direccion: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    correo: {
        type: DataTypes.STRING,
        allowNull: false,
    },
    contrasena: {
        type: DataTypes.STRING,
        allowNull: false,
    },
    tipo: {
        type: DataTypes.INTEGER,
        allowNull: false,
    },
  },
  {
    timestamps: false,
```

Posteriormente se crea un archivo llamado controller.js el cual contiene cada una de las peticiones http, estas serán descritas a continuación:

Peticiones para el producto

```
const getProductos = async (req, res) => {
  try {
    const productos = await Producto.findAll();
    res.status(200).json(productos);
  } catch (error) {
    res.status(400).json({ mensaje: error });
  }
};
```

getProductos obtiene todos los productos que encuentre en la tabla producto de la base de datos

Verificación de resultados en Insomnia

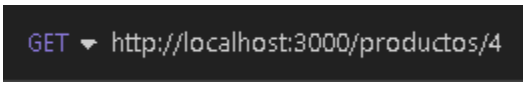
GET ▾ http://localhost:3000/productos


```
[
  {
    "idProducto": 1,
    "nombre": "uva",
    "detalle": "Uva cubana",
    "cantidad": 39,
    "categoria": "temporada",
    "precio": "5000",
    "imagen": "assets/images/uva.jpg"
  },
  {
    "idProducto": 2,
    "nombre": "mango",
    "detalle": "Mango \"Golden Sunrise\"",
    "cantidad": 20,
    "categoria": "pepita",
    "precio": "5000",
    "imagen": "assets/images/mango.jpg"
  },
```

```
const getProducto = async (req, res) => {
  const { idProducto } = req.params;
  try {
    const producto = await Producto.findByPk(idProducto);
    res.status(200).json([producto]);
  } catch (error) {
    res.status(400).json({ mensaje: error });
  }

};
```

getProducto obtiene solo un producto especificando el **idProducto** como parámetro para realizar la operación

Verificación de resultados en Insomnia

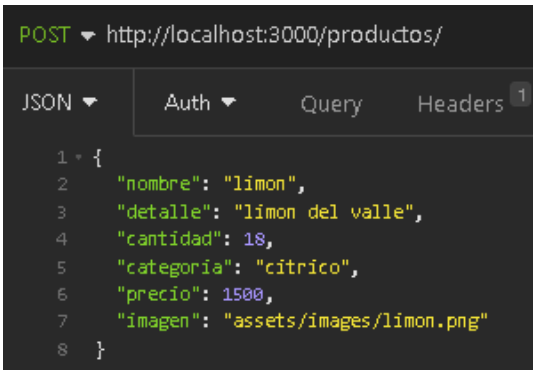


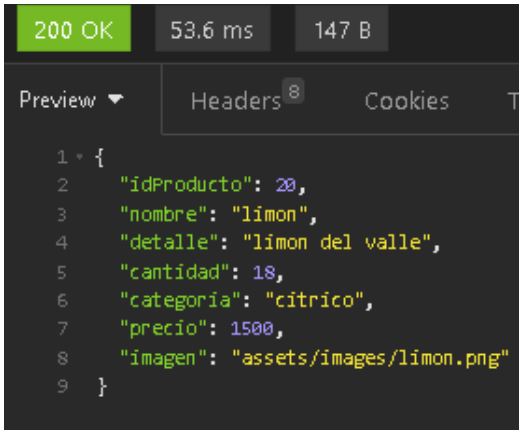

```
[
  {
    "idProducto": 4,
    "nombre": "piña",
    "detalle": " piña tropical",
    "cantidad": 13,
    "categoria": "tropical",
    "precio": "4600",
    "imagen": "assets/images/piña.jpeg"
  }
]
```

```
const postProductos = async (req, res) => {
  const { nombre, detalle, cantidad, categoria, precio, imagen } = req.body;

  try {
    const newProducto = await Producto.create({
      nombre,
      detalle,
      cantidad,
      categoria,
      precio,
      imagen,
    });
    res.status(200).json(newProducto);
  } catch (error) {
    res.status(400).json({ mensaje: error });
  }
};
```

postProductos permite registrar un nuevo producto en la base de datos y se requiere que se especifique el nombre, detalle, cantidad, categoría, precio y la imagen de esa fruta

Verificación de resultados en Insomnia

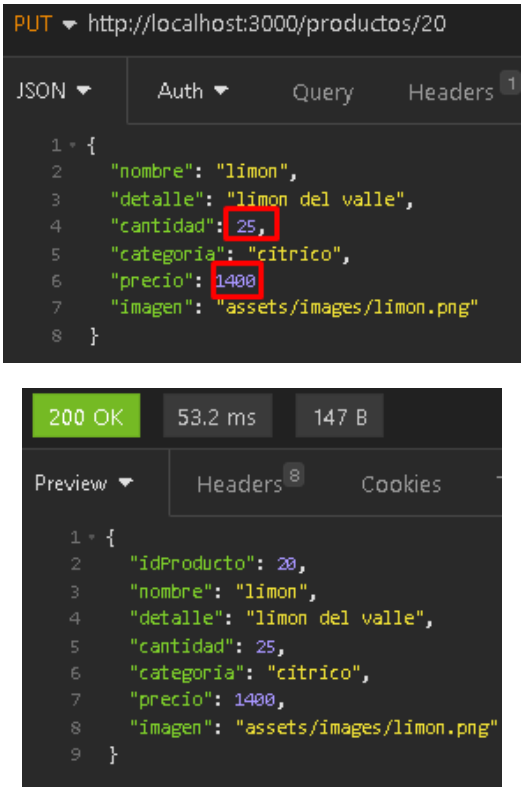




```
const putProductos = async (req, res) => {
  const { idProducto } = req.params;
  const { nombre, detalle, cantidad, categoria, precio, imagen } = req.body;
  try {
    const oldProducto = await Producto.findByPk(idProducto);
    oldProducto.nombre = nombre;
    oldProducto.detalle = detalle;
    oldProducto.cantidad = cantidad;
    oldProducto.categoria = categoria;
    oldProducto.precio = precio;
    oldProducto.imagen = imagen;
    const modProducto = await oldProducto.save();
    res.status(200).json(modProducto);
  } catch (error) {
    res.status(400).json({ mensaje: error });
  }
};
```

putProductos es un método que es capaz de modificar los datos de un producto ya existente o registrado en la base de datos

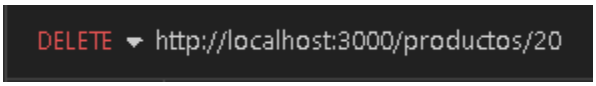
Verificación de resultados en Insomnia

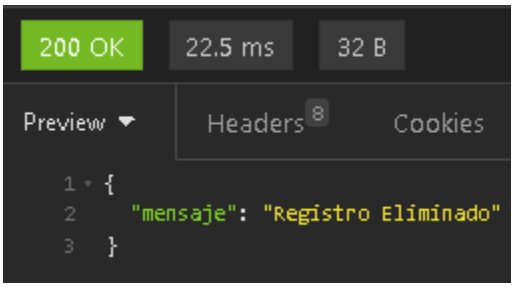


```
const deleteProductos = async (req, res) => {
  const { idProducto } = req.params;
  try {
    const respuesta = await Producto.destroy({
      where: {
        idProducto,
      },
    });
    res.status(200).json({ mensaje: "Registro Eliminado" });
  } catch (error) {
    res.status(400).json({ mensaje: "Registro No Eliminado" + error });
  }
};
```

deleteProductos es capaz de eliminar un producto existente en la base de datos con el idProducto

Verificación de resultados en Insomnia



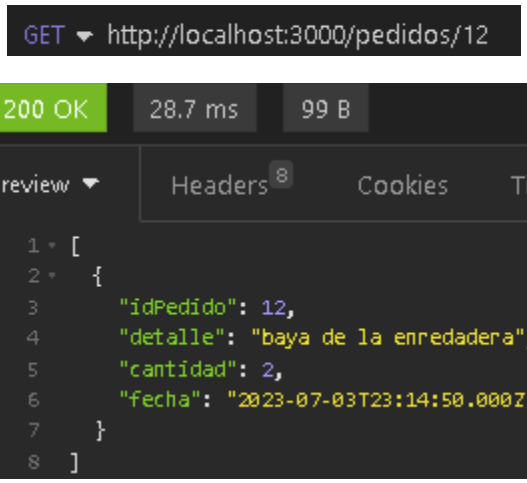


Peticiones para pedidos

```
const getPedido = async (req, res) => {
  const { idPedido } = req.params;
  try {
    const pedido = await Pedido.findByPk(idPedido);
    res.status(200).json([pedido]);
  } catch (error) {
    res.status(400).json({ mensaje: error });
  }
};
```

getPedido obtiene un pedido especificado por el idPedido

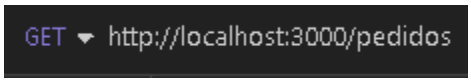
Verificación de resultados en Insomnia

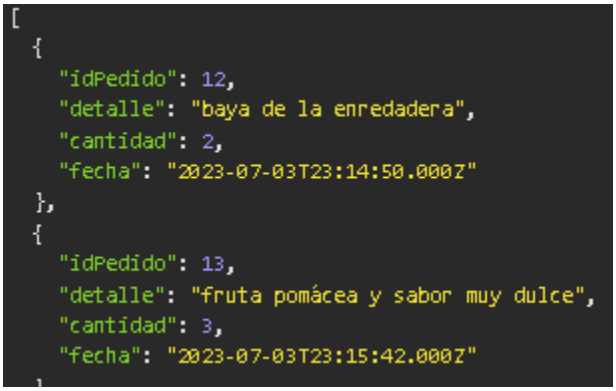



```
const getPedidos = async (req, res) => {
  try {
    const pedidos = await Pedido.findAll();
    res.status(200).json(pedidos);
  } catch (error) {
    res.status(400).json({ mensaje: error });
  }
};
```

getPedidos obtiene todos los pedidos registrados en la base de datos

Verificación de resultados en Insomnia

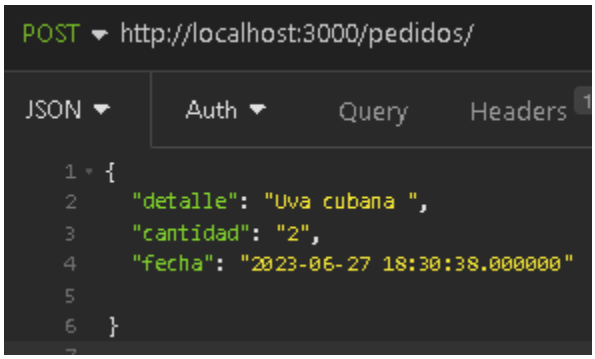


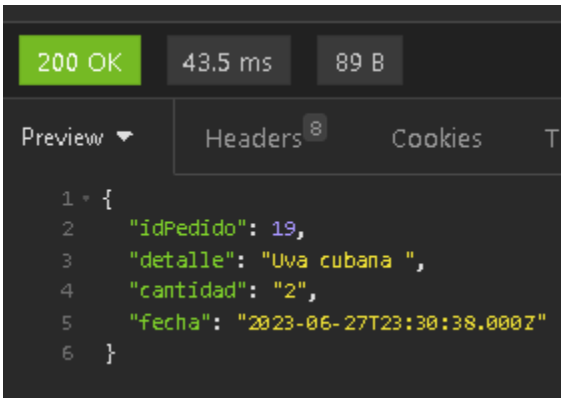


```
const postPedidos = async (req, res) => {
  const { detalle, cantidad, fecha } = req.body;

  try {
    const newPedido = await Pedido.create({
      detalle,
      cantidad,
      fecha,
    });
    res.status(200).json(newPedido);
  } catch (error) {
    res.status(400).json({ mensaje: error });
  }
};
```

postPedidos es capaz de guardar nuevos registros de pedidos donde se requiere el detalle, la cantidad y la fecha que se realizó el pedido

Verificación de resultados en Insomnia

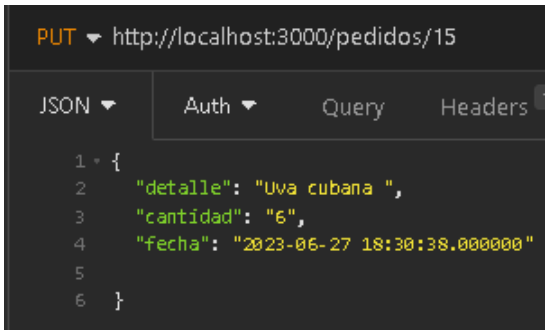


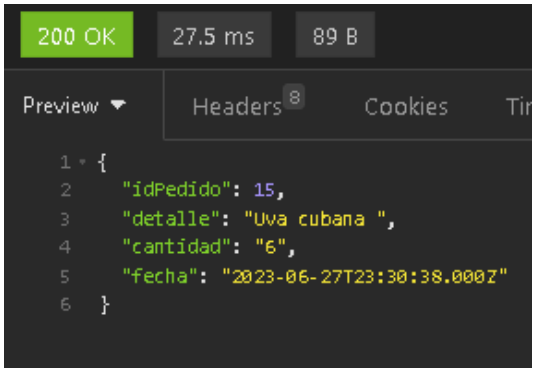


```
const putPedido = async (req, res) => {
  const { idPedido } = req.params;
  const { detalle, cantidad, fecha } = req.body;
  try {
    const oldPedido = await Pedido.findByPk(idPedido);
    oldPedido.detalle = detalle;
    oldPedido.cantidad = cantidad;
    oldPedido.fecha = fecha;
    const modPedido = await oldPedido.save();
    res.status(200).json(modPedido);
  } catch (error) {
    res.status(400).json({ mensaje: error });
  }
};
```

putPedido permite actualizar los datos de los pedidos existentes en la base de datos

Verificación de resultados en Insomnia

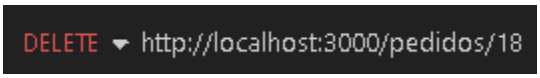




```
const deletePedido = async (req, res) => {
  const { idPedido } = req.params;
  try {
    const respuesta = await Pedido.destroy({
      where: {
        idPedido,
      },
    });
    res.status(200).json({ mensaje: "Pedido Eliminado" });
  } catch (error) {
    res.status(400).json({ mensaje: "Pedido No Eliminado" + error });
  }
};
```

deletePedido tiene la función de borrar un pedido en la base de datos donde se especifica el idPedido para realizar esta operación

Verificación de resultados en Insomnia

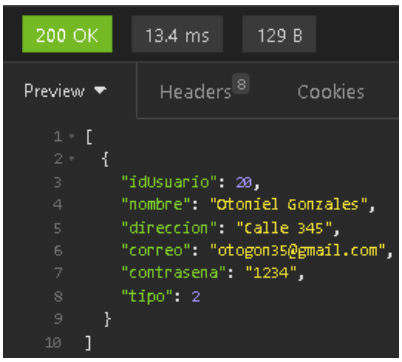


Peticiones para usuarios

```
const getUsuario = async (req, res) => {
  const { idUsuario } = req.params;
  try {
    const usuario = await Usuario.findByPk(idUsuario);
    res.status(200).json([usuario]);
  } catch (error) {
    res.status(400).json({ mensaje: error });
  }
};
```

getUsuario obtiene un registro de usuario especificando el idUsuario para realizar esta operación

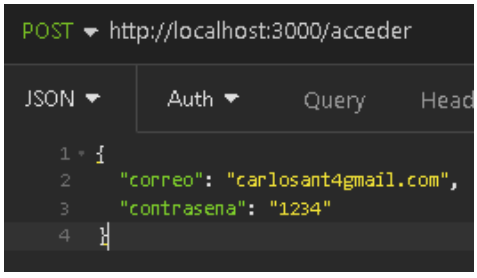
Verificación de resultados en Insomnia

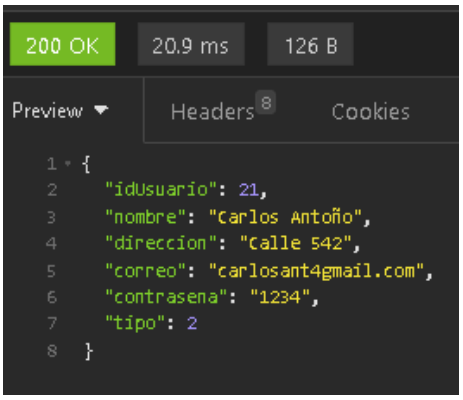



```
const postUsuarioLogin = async (req, res) => {
  const { correo, contrasena } = req.body;

  try {
    const usuario = await Usuario.findOne({
      where: { correo: correo, contrasena: contrasena },
    });
    if (!usuario) {
      return res.status(401).json({ mensaje: "Credenciales inválidas correo" });
    }

    if (usuario.contrasena !== contrasena) {
      return res.status(401).json({ mensaje: "Credenciales inválidas" });
    }

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

postUsuarioLogin permite realizar la autenticación del usuario con el fin de iniciar sesión en la aplicación, para esto se especifica el correo y contraseña

Verificación de resultados en Insomnia






```
const postUsuarios = async (req, res) => {
  const { nombre, direccion, correo, contrasena, tipo } = req.body;
  try {
    const newUsuario = await Usuario.create({
      nombre,
      direccion,
      correo,
      contrasena,
      tipo,
    });
    res.status(200).json(newUsuario);
  } catch (error) {
    res.status(400).json({ mensaje: error });
  }
};
```

postUsuarios es capaz de registrar nuevos usuarios al sistema a partir del nombre, dirección, correo, contraseña y tipo de usuario

Verificación de resultados en Insomnia