

# LAB 12: Implementar API Rest para la clase Medico

**Nombre:** Zurita Paco Elvis Jherson

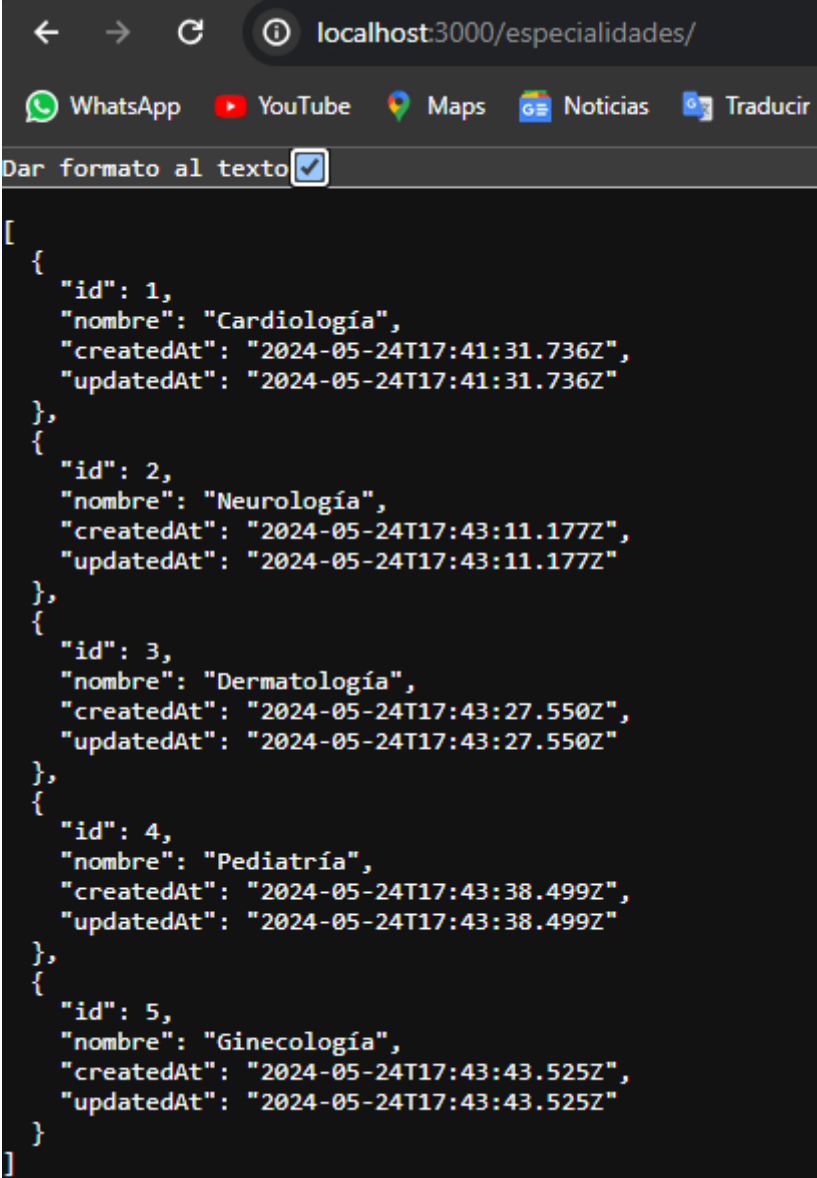
**Carrera:** Ing. de Sistemas

**CU:** 35-5372

## 1. VISUALISACION DE LAS CLASES:

Se implementó las clases médico y especialidad, donde cada médico tendrá una especialidad que será otra tabla. Los datos se insertaron mediante Postman.

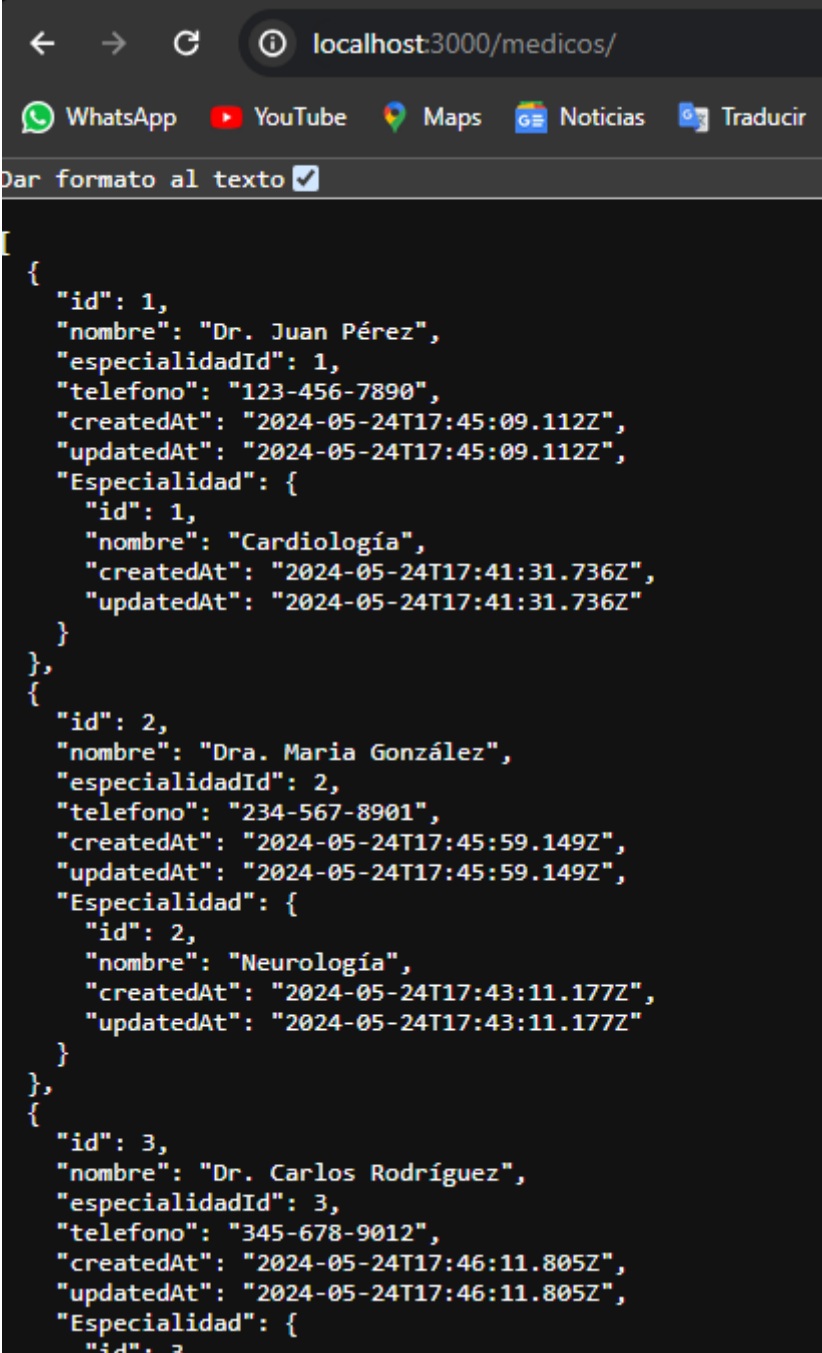
### a) Especialidad:



```
[
  {
    "id": 1,
    "nombre": "Cardiología",
    "createdAt": "2024-05-24T17:41:31.736Z",
    "updatedAt": "2024-05-24T17:41:31.736Z"
  },
  {
    "id": 2,
    "nombre": "Neurología",
    "createdAt": "2024-05-24T17:43:11.177Z",
    "updatedAt": "2024-05-24T17:43:11.177Z"
  },
  {
    "id": 3,
    "nombre": "Dermatología",
    "createdAt": "2024-05-24T17:43:27.550Z",
    "updatedAt": "2024-05-24T17:43:27.550Z"
  },
  {
    "id": 4,
    "nombre": "Pediatría",
    "createdAt": "2024-05-24T17:43:38.499Z",
    "updatedAt": "2024-05-24T17:43:38.499Z"
  },
  {
    "id": 5,
    "nombre": "Ginecología",
    "createdAt": "2024-05-24T17:43:43.525Z",
    "updatedAt": "2024-05-24T17:43:43.525Z"
  }
]
```

## LAB 12: Implementar API Rest para la clase Medico

### b) Médicos:



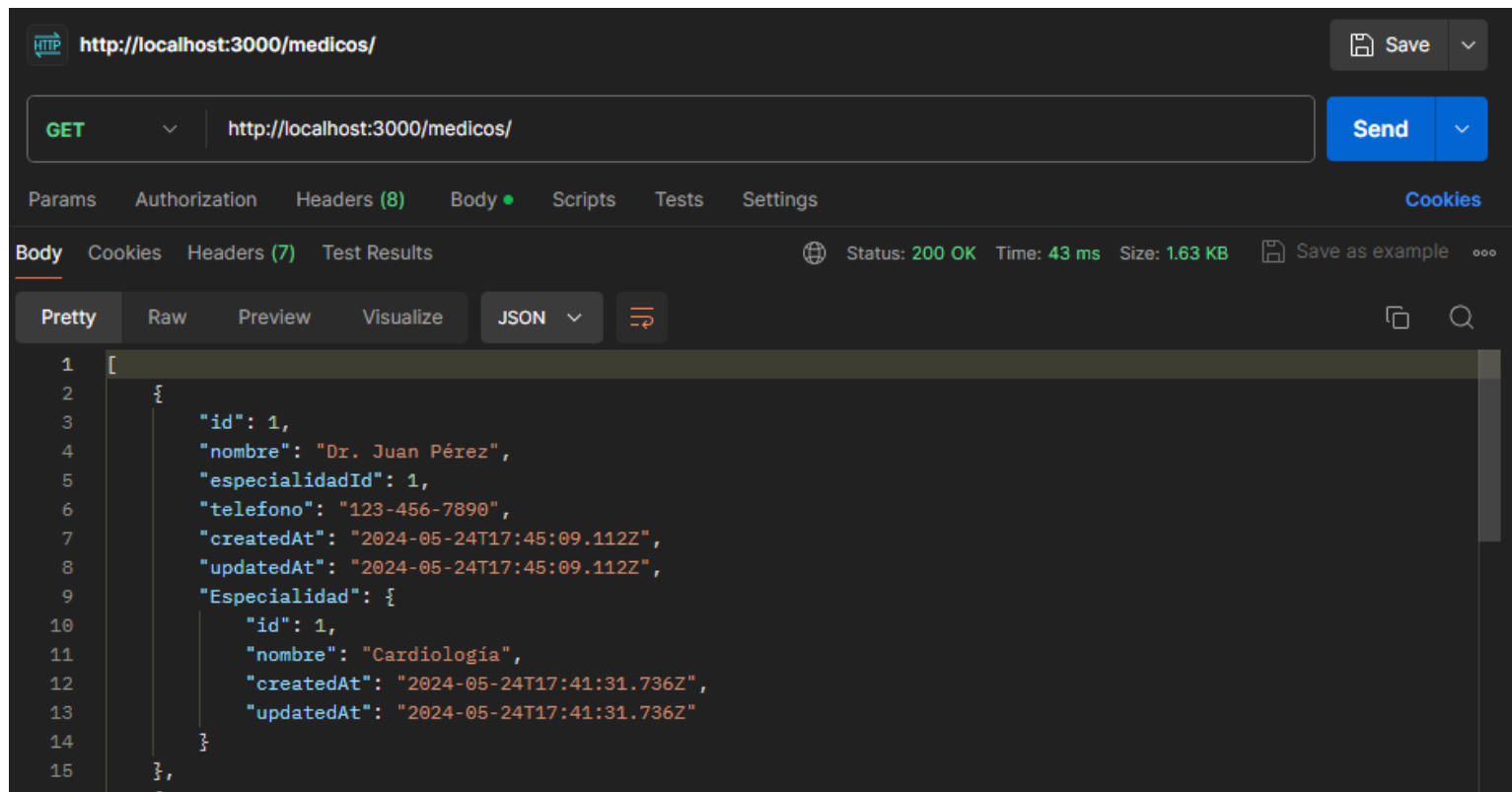
```
{
  "id": 1,
  "nombre": "Dr. Juan Pérez",
  "especialidadId": 1,
  "telefono": "123-456-7890",
  "createdAt": "2024-05-24T17:45:09.112Z",
  "updatedAt": "2024-05-24T17:45:09.112Z",
  "Especialidad": {
    "id": 1,
    "nombre": "Cardiología",
    "createdAt": "2024-05-24T17:41:31.736Z",
    "updatedAt": "2024-05-24T17:41:31.736Z"
  }
},
{
  "id": 2,
  "nombre": "Dra. Maria González",
  "especialidadId": 2,
  "telefono": "234-567-8901",
  "createdAt": "2024-05-24T17:45:59.149Z",
  "updatedAt": "2024-05-24T17:45:59.149Z",
  "Especialidad": {
    "id": 2,
    "nombre": "Neurología",
    "createdAt": "2024-05-24T17:43:11.177Z",
    "updatedAt": "2024-05-24T17:43:11.177Z"
  }
},
{
  "id": 3,
  "nombre": "Dr. Carlos Rodríguez",
  "especialidadId": 3,
  "telefono": "345-678-9012",
  "createdAt": "2024-05-24T17:46:11.805Z",
  "updatedAt": "2024-05-24T17:46:11.805Z",
  "Especialidad": {
    "id": 3,
```

### c) Comandos SQL en la terminal :

```
Executing (default): SELECT `Medico`.`id`, `Medico`.`nombre`, `Medico`.`especialidadId`, `Medico`.`telefono`, `Medico`.`createdAt`, `Medico`.`updatedAt`, `Especialidad`.`id` AS `Especialidad.id`, `Especialidad`.`nombre` AS `Especialidad.nombre`, `Especialidad`.`createdAt` AS `Especialidad.createdAt`, `Especialidad`.`updatedAt` AS `Especialidad.updatedAt` FROM `medicos` AS `Medico` LEFT OUTER JOIN `especialidades` AS `Especialidad` ON `Medico`.`especialidadId` = `Especialidad`.`id` WHERE `Medico`.`id` = '5';
GET /medicos/5 200 10.081 ms - 285
```

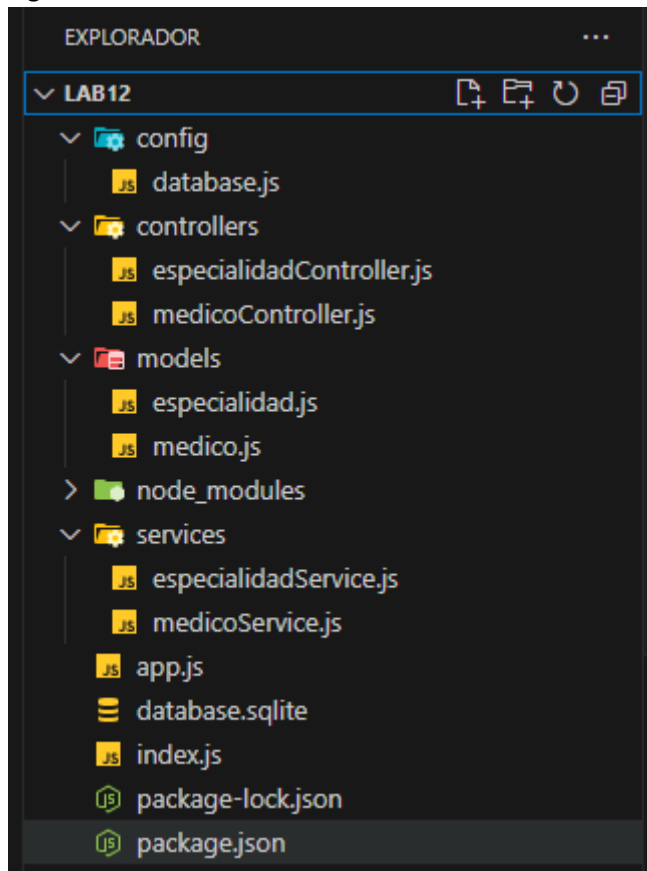
# LAB 12: Implementar API Rest para la clase Medico

## d) PostMan:



# LAB 12: Implementar API Rest para la clase Medico

2. **IMPLEMENTACION DE LAS CLASES:** Para la implementación de las clases se siguió la siguiente estructura:



Aquí se detallará unas clases de importancia:

**database.js:**

```
const { Sequelize } = require('sequelize');

const sequelize = new Sequelize({
  dialect: 'sqlite',
  storage: './database.sqlite'
});

module.exports = sequelize;
```

## LAB 12: Implementar API Rest para la clase Medico

Medico.js:

```
const { DataTypes } = require('sequelize');
const sequelize = require('../config/database');
const Especialidad = require('../especialidad');

const Medico = sequelize.define('Medico', {
  id: {
    type: DataTypes.INTEGER,
    autoIncrement: true,
    primaryKey: true
  },
  nombre: {
    type: DataTypes.STRING,
    allowNull: false
  },
  especialidadId: {
    type: DataTypes.INTEGER,
    allowNull: false,
    references: {
      model: Especialidad,
      key: 'id'
    }
  },
  telefono: {
    type: DataTypes.STRING,
    allowNull: false
  }
}, {
  tableName: 'medicos'
});

Medico.belongsTo(Especialidad, { foreignKey: 'especialidadId' });
Especialidad.hasMany(Medico, { foreignKey: 'especialidadId' });

module.exports = Medico;
```

# LAB 12: Implementar API Rest para la clase Medico

## Especialidad.js:

```
const { DataTypes } = require('sequelize');
const sequelize = require('../config/database');

const Especialidad = sequelize.define('Especialidad', {
  id: {
    type: DataTypes.INTEGER,
    autoIncrement: true,
    primaryKey: true
  },
  nombre: {
    type: DataTypes.STRING,
    allowNull: false
  }
}, {
  tableName: 'especialidades'
});

module.exports = Especialidad;
```

## Index.js:

```
const app = require('./app');

const PORT = process.env.PORT || 3000;

app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

## App.js:

```
const express = require('express');
const bodyParser = require('body-parser');
const morgan = require('morgan');
const medicoController = require('./controllers/medicoController');
const especialidadController =
require('./controllers/especialidadController');
const sequelize = require('../config/database');

const app = express();

app.use(morgan('dev'));
app.use(bodyParser.json());

app.post('/medicos', medicoController.addMedico);
app.get('/medicos', medicoController.getMedicos);
```

## LAB 12: Implementar API Rest para la clase Medico

```
app.get('/medicos/:id', medicoController.getMedicoById);
app.put('/medicos/:id', medicoController.updateMedico);
app.delete('/medicos/:id', medicoController.deleteMedico);

app.get('/especialidades', especialidadController.getAllEspecialidades);
app.get('/especialidades/:id',
especialidadController.getEspecialidadById);
app.post('/especialidades', especialidadController.createEspecialidad);
app.put('/especialidades/:id',
especialidadController.updateEspecialidad);
app.delete('/especialidades/:id',
especialidadController.deleteEspecialidad);

sequelize.sync({ force: true })
  .then(() => {
    console.log('Database synchronized');
  })
  .catch(err => {
    console.error('Unable to synchronize the database:', err);
  });

module.exports = app;
```

### Resumen

**SQLite:** Un sistema de gestión de bases de datos relacional ligero, que utiliza un único archivo para almacenar toda la base de datos.

**Sequelize:** Un ORM para Node.js que facilita la interacción con bases de datos relacionales utilizando modelos y consultas en JavaScript.

**Postman:** Una herramienta de desarrollo de API que facilita el envío de solicitudes HTTP, incluyendo métodos POST para agregar datos a la base de datos.