

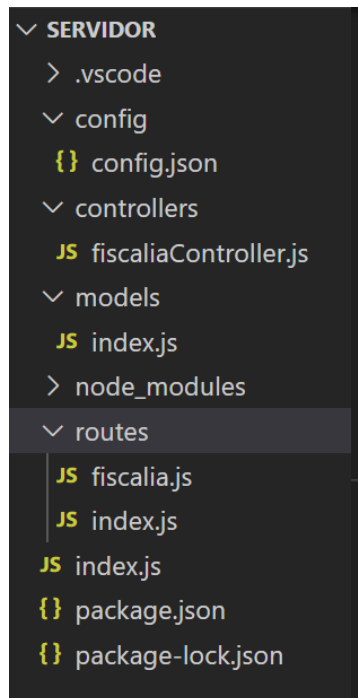
## Manual técnico

Para este proyecto se utilizó:

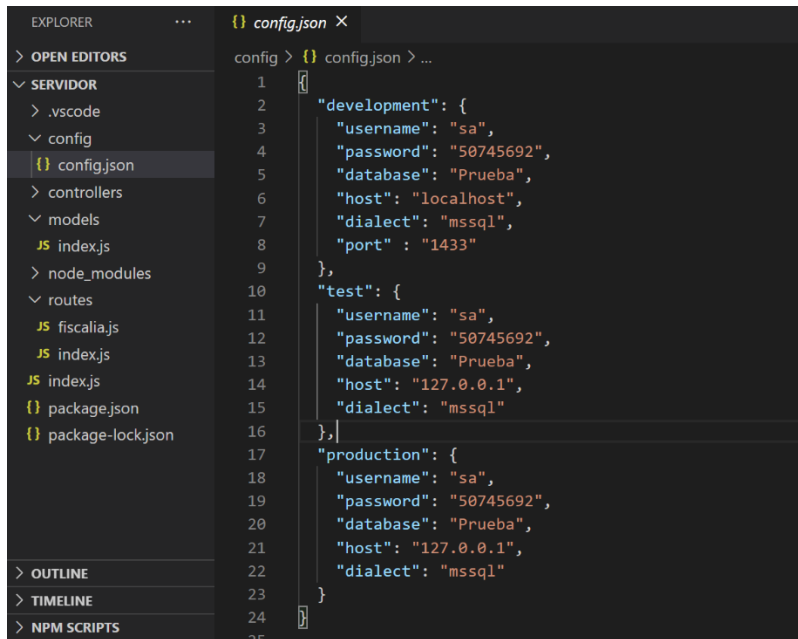
- node js v 14.8
- body-parser v 1.19
- cors v 2.8.5
- express v 4.17
- sequelize v 6.3
- tedious v 9.1

Objetivo: El objetivo de este proyecto es tener al alcance la información de las distintas fiscalías, además de poder actualizarlas o eliminarlas cuando sea necesario.

Estructura: El proyecto está estructurado de la siguiente manera



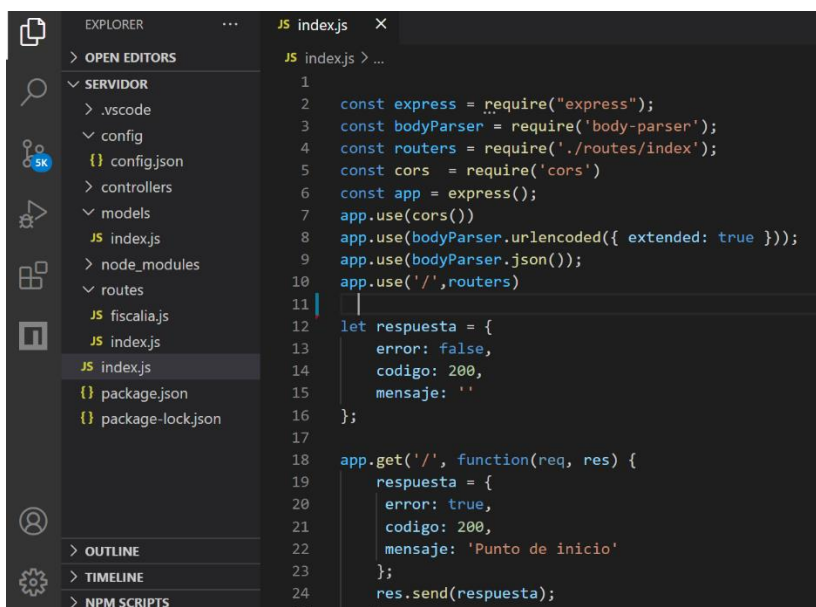
**Apartado config:** en este apartado contamos con un archivo config.json, el cual es utilizado para la configuración de nuestra cadena de conexión a la base de datos, en este caso se utilizó sql, al hacer uso del comando npm sequelize-cli init, este nos crea un archivo de configuración por defecto el cual debe editarse con los datos de nuestra conexión, para este proyecto se colocó la misma configuración en los 3 tipos de cadenas ya que se esta utilizando una misma base de datos tanto como para desarrollo, pruebas y producción.



```
1 {
2   "development": {
3     "username": "sa",
4     "password": "50745692",
5     "database": "Prueba",
6     "host": "localhost",
7     "dialect": "mssql",
8     "port": "1433"
9   },
10  "test": {
11    "username": "sa",
12    "password": "50745692",
13    "database": "Prueba",
14    "host": "127.0.0.1",
15    "dialect": "mssql"
16  },
17  "production": {
18    "username": "sa",
19    "password": "50745692",
20    "database": "Prueba",
21    "host": "127.0.0.1",
22    "dialect": "mssql"
23  }
24 }
```

**Archivo index.js:** en este archivo se realizan las configuraciones de nuestro servidor, como lo es la importación de express, bodyParser para el manejo de los archivos json, cors para las solicitudes del navegador, routers que se utiliza para cargar

las rutas de nuestro api, también en este apartado se define en que puerto deberá ejecutarse nuestra aplicación.



```
1
2 const express = require("express");
3 const bodyParser = require('body-parser');
4 const routers = require('./routes/index');
5 const cors = require('cors')
6 const app = express();
7 app.use(cors())
8 app.use(bodyParser.urlencoded({ extended: true }));
9 app.use(bodyParser.json());
10 app.use('/', routers)
11
12 let respuesta = {
13   error: false,
14   codigo: 200,
15   mensaje: ''
16 };
17
18 app.get('/', function(req, res) {
19   respuesta = {
20     error: true,
21     codigo: 200,
22     mensaje: 'Punto de inicio'
23   };
24   res.send(respuesta);
25 }
```

**Apartado routes:** En este apartado se cuenta con dos archivos, el archivo `fiscalía.js` y el archivo `index.js`.

- **fiscalía.js:** en este archivo se configuran las rutas de nuestro api, en nuestro caso se definió una única ruta a la cual se le configuraron sus 4 llamadas: get, post, put y delete

```
JS index.js JS fiscalia.js X
routes > JS fiscalia.js > ...
1  const express = require("express");
2  var router = express.Router();
3  const controller = require('../controllers/fiscaliaController')
4
5  router.route('/')
6    .get(controller.GetFiscalias)
7
8    .post(controller.SetFiscalia)
9
10   .put(controller.UpdateFiscalia)
11
12   .delete(controller.DeleteFiscalia)
13 ;
14 module.exports = router;
15
```

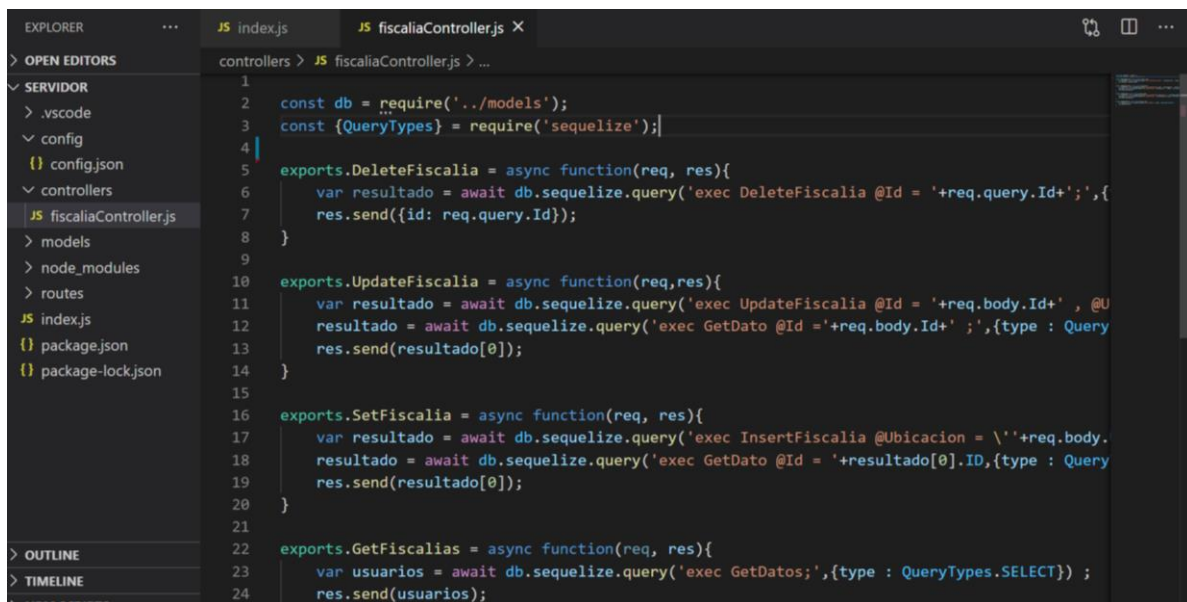
- **index.js:** este archivo es utilizado para unir las distintas rutas, en nuestro caso solo tenemos la ruta `fiscalía`, por lo que se hace el importe de nuestro archivo `fiscalía.js` y se registra como ruta de acceso la llamada `"/fiscalia"`

```
JS index.js \ JS index.js routes X
routes > JS index.js > ...
1  const fiscalia = require( './fiscalia')
2  const express = require("express");
3  var router = express.Router();
4
5  router.use('/fiscalia', fiscalia);
6
7  module.exports = router;
```

**Apartado Controllers:** en este apartado registramos los controladores de nuestro api, aquí va la acción que realizara cada llamada a nuestra api, en este caso tenemos 4 funciones las cuales son:

- DeleteFiscalia que se utiliza para realizar un delete a nuestra base de datos.
- UpdateFiscalia la cual se utiliza para generar un update a nuestra base de datos
- SetFiscalia que se utiliza para realizar un insert a nuestra base de datos
- GetFiscalias la cual se utiliza para obtener todas las fiscalías almacenadas en nuestra base de datos.

Cada una de nuestras acciones a la base de datos se realiza mediante una llamada a un procedimiento almacenado.



```
1
2 const db = require('../models');
3 const {QueryTypes} = require('sequelize');
4
5 exports.DeleteFiscalia = async function(req, res){
6   var resultado = await db.sequelize.query('exec DeleteFiscalia @Id = '+req.query.Id+',{
7     res.send({id: req.query.Id});
8   }
9
10  exports.UpdateFiscalia = async function(req,res){
11    var resultado = await db.sequelize.query('exec UpdateFiscalia @Id = '+req.body.Id+ ' , @U
12    resultado = await db.sequelize.query('exec GetDato @Id = '+req.body.Id+ '',{type : Query
13    res.send(resultado[0]);
14  }
15
16  exports.SetFiscalia = async function(req, res){
17    var resultado = await db.sequelize.query('exec InsertFiscalia @Ubicacion = \''+req.body.
18    resultado = await db.sequelize.query('exec GetDato @Id = '+resultado[0].ID,{type : Query
19    res.send(resultado[0]);
20  }
21
22  exports.GetFiscalias = async function(req, res){
23    var usuarios = await db.sequelize.query('exec GetDatos',{type : QueryTypes.SELECT}) ;
24    res.send(usuarios);
```

## Esquema de base de datos

Fiscalia	
	id
	Ubicacion
	Telefono