

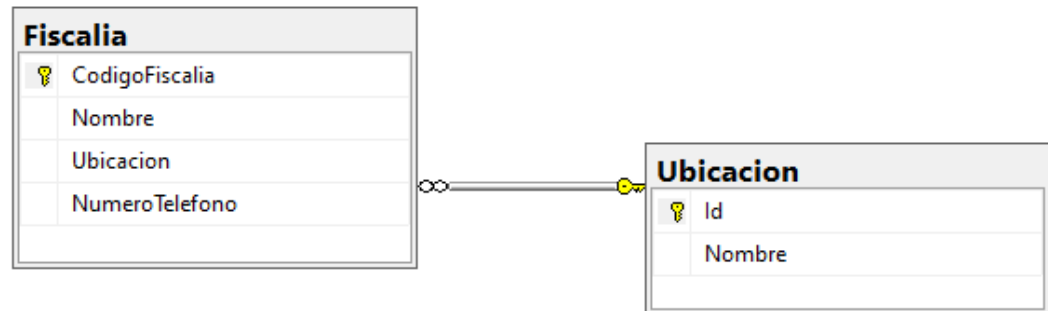
Manual Tecnico

Prueba Tecnica MP

Herber Manolo Cerón Elías

Base de Datos

- Principalmente se utilizó SQL Server en su versión 2022
- Se creó la base de datos MPDB, contiene 2 tablas, Fiscalías y Ubicación



- De igual forma se llenaron los datos necesarios para el proyecto
- Las Credenciales de la base de Datos son MpAdmin y Mp301295

Results				
	CodigoFiscalia	Nombre	Ubicacion	NumeroTelefono
1	202	Pueblosanto	1	32654879
2	203	Chiquier	2	65458798
3	204	Tapachula	10010	64587987
4	205	Santosano	10014	32654879

	Id	Nombre
1	1	Chiquimula
2	2	Zacapa
3	10002	Alta Verapaz
4	10003	Baja Verapaz
5	10004	Chimaltenango
6	10005	Petén
7	10006	El Progreso
8	10007	Quiché
9	10008	Escuintla
10	10009	Guatemala
11	10010	Huehuetena...
12	10011	Izabal
13	10012	Jalapa
14	10013	Jutiapa
15	10014	Quetzaltena...
16	10015	Retalhuleu
17	10016	Sacatepéquez
18	10017	San Marcos
19	10018	Santa Rosa
20	10019	Sololá
21	10020	Suchitepéquez
22	10021	Totonicapán

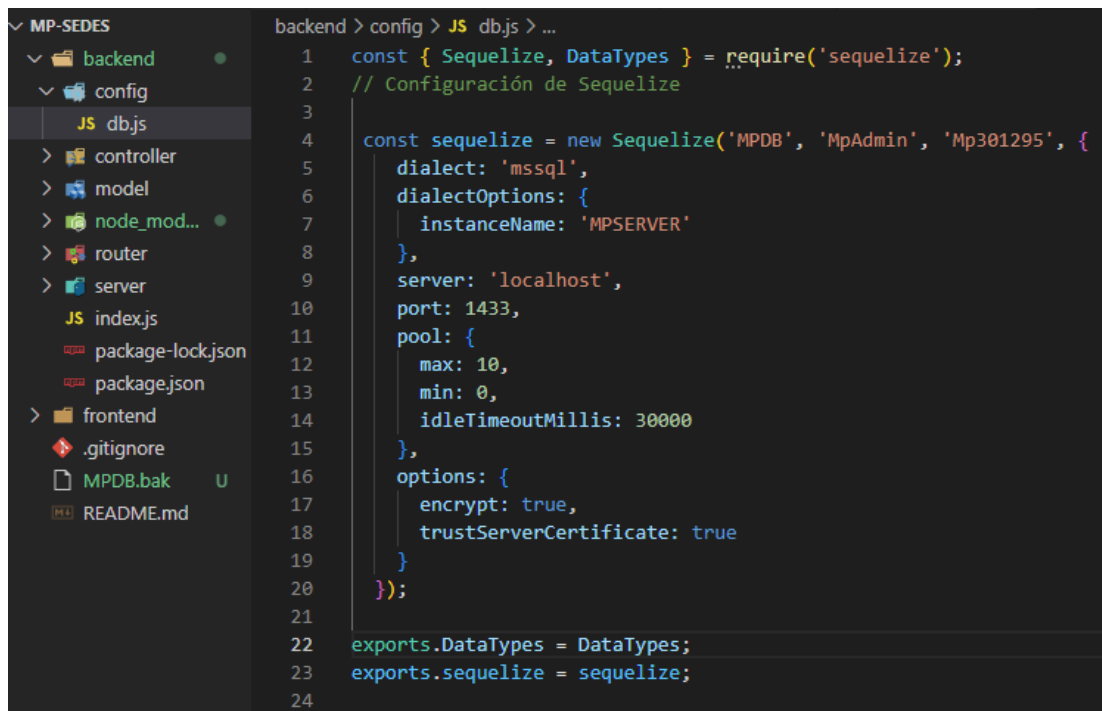
Backend

- Para comenzar se utilizó Node y Express para desarrollar la API Restful, así como Sequelize para facilitar el medio de conexión con la base de datos que en este caso es SQL Server.

```
{
  "name": "backend",
  "version": "1.0.0",
  "description": "backend de servicio mp",
  "main": "index.js",
  > Debug
  "scripts": {
    "nodemon": "nodemon index.js",
    "start": "node index.js"
  },
  "keywords": [
    "SERN",
    "Node"
  ],
  "author": "Herber Manolo Cerón Elías",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.20.2",
    "cors": "^2.8.5",
    "express": "^4.18.2",
    "sequelize": "^6.29.3",
    "tedious": "^15.1.3"
  },
  "devDependencies": {
    "nodemon": "^2.0.21"
  }
}
```

- Como se puede apreciar en la imagen anterior se utilizaron body-parser, cors, express, sequelize y tedious, este ultimo es mas que todo como driver para la conexión.

- Para levantar la conexión se realizó de la siguiente manera



```
backend > config > JS dbjs > ...
1  const { Sequelize, DataTypes } = require('sequelize');
2  // Configuración de Sequelize
3
4  const sequelize = new Sequelize('MPDB', 'MpAdmin', 'Mp301295', {
5    dialect: 'mysql',
6    dialectOptions: {
7      instanceName: 'MPSERVER'
8    },
9    server: 'localhost',
10   port: 1433,
11   pool: {
12     max: 10,
13     min: 0,
14     idleTimeoutMillis: 30000
15   },
16   options: {
17     encrypt: true,
18     trustServerCertificate: true
19   }
20 });
21
22 exports.DataTypes = DataTypes;
23 exports.sequelize = sequelize;
24
```

- Se utilizó una conexión básica con Sequelize, por motivo de tiempo no se utilizaron variables de entorno, en su lugar se dejaron las credenciales establecidas físicamente.
- Luego se establecieron y crearon los modelos para la estructura y manejo de datos del lado de la API, específicamente Fiscalía y Ubicación los cuales se muestran en las siguientes páginas.

MP-SEDES

backend

config

JS db.js

controller

JS fiscalia.js

model\MPDB

JS fiscalia.js

JS init-models.js

JS ubicacion.js

node_mod...

router

server

JS index.js

package-lock.json

package.json

frontend

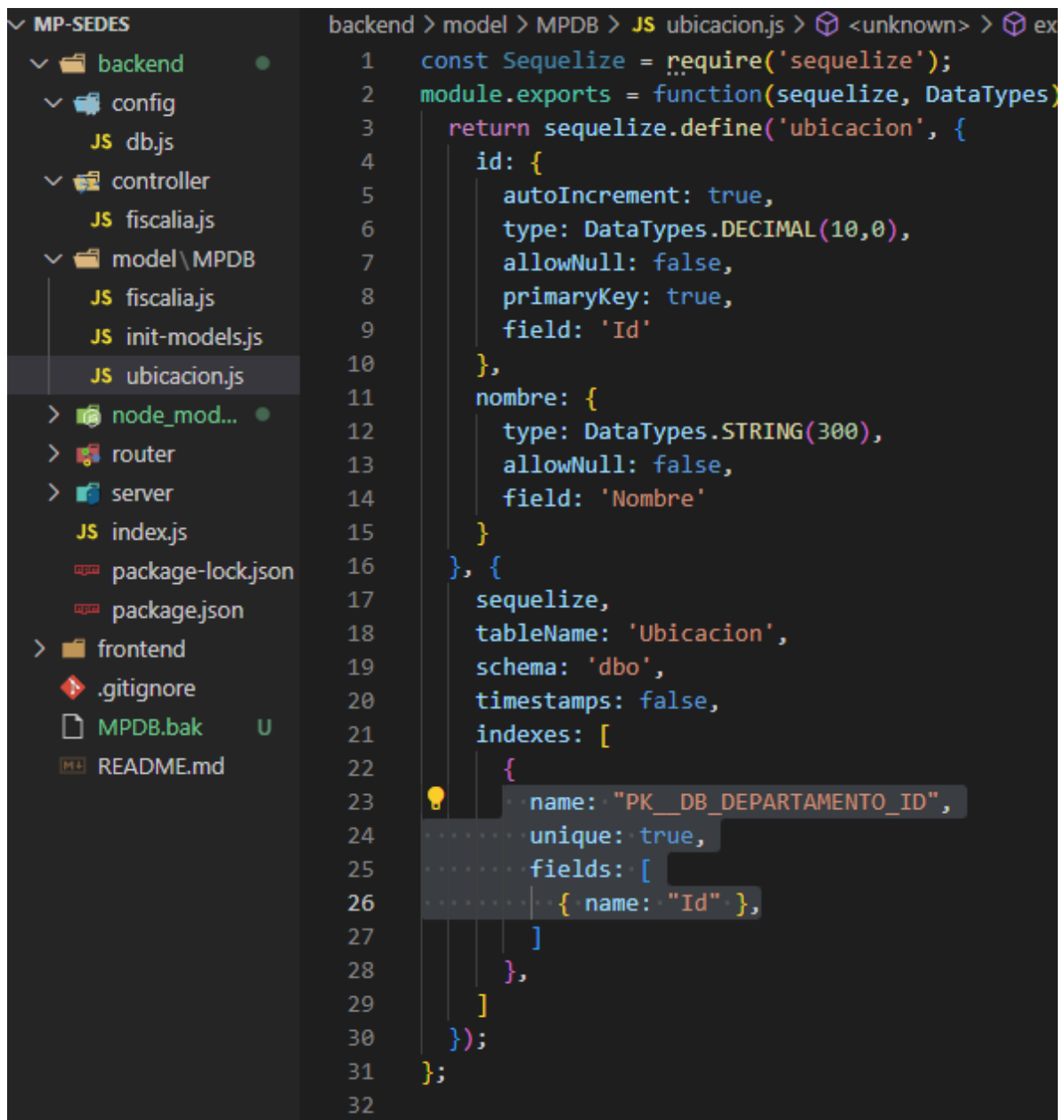
.gitignore

MPDB.bak

README.md

backend > model > MPDB > JS fiscalia.js > <unknown> > exports

```
1  const Sequelize = require('sequelize');
2  module.exports = function(sequelize, DataTypes) {
3    return sequelize.define('fiscalia', {
4      codigoFiscalia: {
5        type: DataTypes.INTEGER,
6        allowNull: false,
7        primaryKey: true,
8        field: 'CodigoFiscalia'
9      },
10     nombre: {
11       type: DataTypes.STRING(50),
12       allowNull: false,
13       field: 'Nombre'
14     },
15     ubicacion: {
16       type: DataTypes.DECIMAL(10,0),
17       allowNull: true,
18       references: {
19         model: 'Ubicacion',
20         key: 'Id'
21       },
22       field: 'Ubicacion'
23     },
24     numeroTelefono: {
25       type: DataTypes.STRING(20),
26       allowNull: false,
27       field: 'NumeroTelefono'
28     }
29   }, {
30     sequelize,
31     tableName: 'Fiscalia',
32     schema: 'dbo',
33     timestamps: false,
34     indexes: [
35       {
36         name: "PK__Fiscalia__5BA6D7C2967D73FE",
37         unique: true,
38         fields: [
39           { name: "CodigoFiscalia" },
40         ]
41       },
42     ]
43   });
44 };
45
```



```
backend > model > MPDB > JS ubicacion.js > <unknown> > ex
1  const Sequelize = require('sequelize');
2  module.exports = function(sequelize, DataTypes) {
3    return sequelize.define('ubicacion', {
4      id: {
5        autoIncrement: true,
6        type: DataTypes.DECIMAL(10,0),
7        allowNull: false,
8        primaryKey: true,
9        field: 'Id'
10     },
11     nombre: {
12       type: DataTypes.STRING(300),
13       allowNull: false,
14       field: 'Nombre'
15     }
16   }, {
17     sequelize,
18     tableName: 'Ubicacion',
19     schema: 'dbo',
20     timestamps: false,
21     indexes: [
22       {
23         name: "PK_DB_DEPARTAMENTO_ID",
24         unique: true,
25         fields: [
26           { name: "Id" },
27         ]
28       },
29     ]
30   });
31 };
32
```

- De igual forma se inicializaron los modelos para realizar un llamado de manera más dinámica dentro del controlador. el archivo es el siguiente:



```
backend > model > MPDB > JS init-models.js > ...
1  var DataTypes = require("sequelize").DataTypes;
2  var _fiscalia = require("../fiscalia");
3  var _ubicacion = require("../ubicacion");
4
5  function initModels(sequelize) {
6    var fiscalia = _fiscalia(sequelize, DataTypes);
7    var ubicacion = _ubicacion(sequelize, DataTypes);
8
9    fiscalia.belongsTo(ubicacion, { as: "lugar", foreignKey: "ubicacion"});
10   ubicacion.hasMany(fiscalia, { as: "fiscalia", foreignKey: "ubicacion"});
11
12   return {
13     fiscalia,
14     ubicacion,
15   };
16 }
17 module.exports = initModels;
18 module.exports.initModels = initModels;
19 module.exports.default = initModels;
20
```

- Con este ultimo archivo concluye el contenido de los models, como pueden observar se debe tomar muy en cuenta los atributos de cada item de las tablas, ya que si esta mal especificado en los modelos, lo más seguro es que las llamadas de los endpoint a la base de datos no funcionen correctamente. En este archivo se configuro también la estructura de unión de las tablas como se ve en la imagen anterior.
- Como siguiente punto se estableció el controlador que maneja todas funciones para interactuar con la base de datos desde un FindAll de sequelize que equivale a un Select, update, destroy que equivale a un delete, el create que equivale al insert y el findByPk que es un select para un solo dato. A continuación, se presenta la estructura del controlador Fiscalia

```
✓ MP-SEDES
  ✓ backend
  ✓ config
    JS dbjs
  ✓ controller
    JS fiscalia.js
  ✓ model \MPDB
    JS fiscalia.js
    JS init-models.js
    JS ubicacion.js
  > node_mod...
  > router
  > server
    JS index.js
    package-lock.json
    package.json
  > frontend
  .gitignore
  MPDB.bak
  README.md

backend > controller > JS fiscalia.js > getAllFiscalias > fiscalias > include
1  const { where } = require('sequelize')
2  const { sequelize, DataTypes } = require('../config/db')
3  const initModels = require('../model/MPDB/init-models')
4  const models = initModels(sequelize)
5
6  const FiscaliaModel = models.fiscalia
7  const UbicacionModel = models.ubicacion
8
9  async function getAllFiscalias(req, res) { //obtener todos los registros
10   try {
11     const fiscalias = await FiscaliaModel.findAll({
12       include:[
13         {
14           model: UbicacionModel,
15           as: 'lugar'
16         }
17       ]
18     });
19     res.json(fiscalias);
20   } catch (error) {
21     console.error(error);
22     res.status(500).json({ message: 'Hubo un error al obtener las fiscalías.' });
23   }
24 }
25
26 async function updateFiscalias(req, res) {
27   const { id } = req.params
28   const { codigoFiscalia, nombre, ubicacion, numeroTelefono } = req.body;
29   try {
30     const updateFiscalia = await FiscaliaModel.update({
31       codigoFiscalia,
32       nombre,
33       ubicacion,
34       numeroTelefono
35     }, {
36       where: {
37         codigoFiscalia: id
38       }
39     });
40     res.status(200).send(updateFiscalia);
41   } catch (error) {
42     console.error(error);
43     res.status(500).json({ message: 'Hubo un error al obtener las fiscalías.' });
44   }
45 }
```



```

async function deleteFiscalias(req, res) {
  const { id } = req.params;
  try {
    const fiscalias = await FiscaliaModel.destroy({
      where: {
        codigoFiscalia: id
      }
    });
    res.status(200).send(true);
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Hubo un error al obtener las fiscalías.' });
  }
}

async function insertFiscalias(req, res) {
  const { codigoFiscalia, nombre, ubicacion, numeroTelefono } = req.body;
  console.log(codigoFiscalia, nombre, ubicacion, numeroTelefono)
  try {
    const nuevaFiscalia = await FiscaliaModel.create({
      codigoFiscalia,
      nombre,
      ubicacion,
      numeroTelefono
    });
    res.status(200).send(nuevaFiscalia);
    console.log("obtuvimos datos pero debes revisar el registro con el get")
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Hubo un error al insertar las fiscalías.' });
  }
}

async function getAllLugares(req, res) { //obtener todos los registros
  try {
    const lugares = await UbicacionModel.findAll();
    res.json(lugares);
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Hubo un error al obtener los lugares.' });
  }
}

```

```

async function getFiscaliaById(req, res) {
  const { id } = req.params;

  try {
    const fiscalia = await FiscaliaModel.findByPk(id, {
      include: [
        {
          model: UbicacionModel,
          as: 'lugar',
        },
      ],
    });

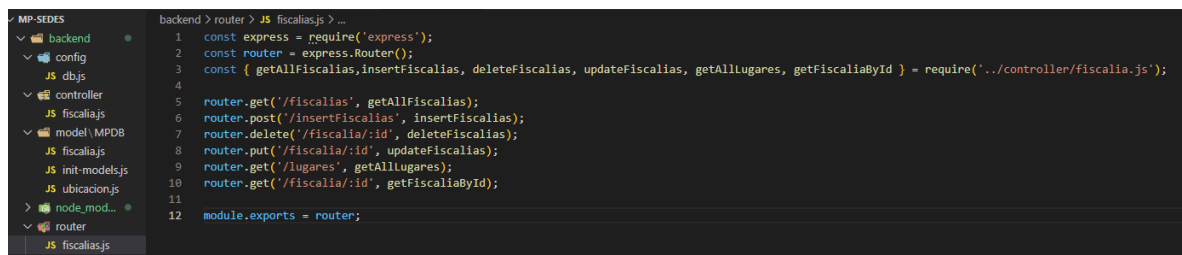
    if (!fiscalia) {
      return res.status(404).json({ message: 'Fiscalía no encontrada.' });
    }

    res.json(fiscalia);
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Hubo un error al obtener la fiscalía.' });
  }
}

module.exports = {
  getAllFiscalias,
  updateFiscalias,
  deleteFiscalias,
  insertFiscalias,
  getAllLugares,
  getFiscaliaById
};

```

- Al final del controlador se importan las funciones para cada requerimiento.
- Luego de esto se trabaja el archivo de Router para establecer los endpoint que se deben acceder para consumirlos.



```

backend > router > JS fiscalias.js > ...
1  const express = require('express');
2  const router = express.Router();
3  const { getAllFiscalias, insertFiscalias, deleteFiscalias, updateFiscalias, getAllLugares, getFiscaliaById } = require('../controller/fiscalia.js');
4
5  router.get('/fiscalias', getAllFiscalias);
6  router.post('/insertFiscalias', insertFiscalias);
7  router.delete('/fiscalia/:id', deleteFiscalias);
8  router.put('/fiscalia/:id', updateFiscalias);
9  router.get('/lugares', getAllLugares);
10 router.get('/fiscalia/:id', getFiscaliaById);
11
12 module.exports = router;

```

- Cada router esta definida su funcion, siendo estas get, post, delete, y put

- Como ultimo el archivo index que es el principal debe importarse todas las dependencias necesarias para que se ejecute sin problemas en este caso body-parser para que no tenga problemas en la lectura de los datos de sql a json, cors para que la conexión pueda ser establecida y no sea bloqueada ya que no es una conexión segura por ejecutarse sobre http.

```
backend > JS index.js > test
1  const express = require('express');
2  const connection = require('./config/db'); // Importa la conexión
3  const fiscaliasRouter = require('./router/fiscalias');
4  const bodyParser = require('body-parser')
5  const cors = require("cors")
6
7  const app = express();
8
9  app.use(cors())
10 app.use(bodyParser.json())
11 app.use(fiscaliasRouter)
12
13
14 async function test(){
15   try {
16     await connection.sequelize.authenticate();
17     console.log("exitoso")
18   }catch(error){
19     console.error("fallo")
20   }
21
22 }
23
24 test()
25
26
27 app.listen(3000, () => {
28   console.log('Servidor iniciado en http://localhost:3000');
29 });
30
```

FrontEnd

Para este se utilizó React, React-bootstrap y axios (para la conexión con el backend)

```
frontend > npm package.json > ...
1  {
2    "name": "frontend",
3    "private": true,
4    "version": "0.0.0",
5    "type": "module",
6    "scripts": {
7      "dev": "vite",
8      "build": "vite build",
9      "preview": "vite preview"
10   },
11   "dependencies": {
12     "axios": "^1.3.4",
13     "bootstrap": "^5.2.3",
14     "react": "^18.2.0",
15     "react-bootstrap": "^2.7.2",
16     "react-dom": "^18.2.0"
17   },
18   "devDependencies": {
19     "@types/react": "^18.0.28",
20     "@types/react-dom": "^18.0.11",
21     "@vitejs/plugin-react": "^3.1.0",
22     "vite": "^4.2.0"
23   }
24 }
25
```

- Se estableció la conexión mediante axios con el siguiente archivo

```
JS axios.js  X  JS Fiscalia.js  JS Lugares.js
frontend > src > config > JS axios.js > ...
1  import axios from "axios"
2
3  const clienteAxios = axios.create({
4    baseURL: 'http://localhost:3000',
5  })
6
7  export default clienteAxios
8
```

- Luego se establecieron cada uno de los servicios para consumir los endpoints que se crearon del lado del backend, tanto de fiscalías como de lugares (ubicaciones)

frontend > src > services > JS Fiscalia.js > getFiscalias

```
1  import clienteAxios from "../config/axios"
2
3  const getFiscalias = async () => {
4    const dataTable = []
5
6    const resDatosTable = await clienteAxios
7      .get("/fiscalias")
8      .then((res) => {
9        return res.data
10      })
11      .catch((err) => {
12        console.error(err)
13      })
14
15    for (const dataObj of resDatosTable) {
16      dataTable.push(dataObj)
17    }
18
19    return dataTable
20  }
21
22  const saveFiscalia = async (datos) => {
23    try {
24      const respuesta = await clienteAxios.post("/insertFiscalias", datos)
25      return {
26        success: true,
27        data: respuesta.data,
28      }
29    } catch (error) {
30      return {
31        success: false,
32        mensaje: error.response.data.msg,
33      }
34    }
35  }
36
```

```

const updateFiscalia = async (id, datos) => {
  try {
    const datoEditado = await clienteAxios.put(`/fiscalia/${id}`, datos)
    return {
      success: true,
      data: datoEditado.data,
    }
  } catch (error) {
    return {
      success: false,
      mensaje: error.response.data.msg,
    }
  }
}

const deleteFiscalias= async (id) => {
  const datos = {
    isDeleted: true,
  }
  try {
    const respuesta = await clienteAxios.delete(`/fiscalia/${id}`, datos).catch((error) => {
      console.error(error)
    })

    return respuesta.data
  } catch (error) {
    console.error(error)
  }
}

const obtenerFiscaliaPorId = async (id) => {
  try {
    const response = await clienteAxios.get(`/fiscalia/${id}`);
    return {
      success: true,
      data: response.data,
    }
  } catch (error) {
    return {
      success: false,
      mensaje: error.response.data.msg,
    }
  }
}

export { getFiscalias, saveFiscalia, updateFiscalia, deleteFiscalias, obtenerFiscaliaPorId }

```

```

import clienteAxios from "../config/axios"

const getLugares = async () => {
  const dataTable = []

  const resDatosTable = await clienteAxios
    .get("/lugares")
    .then((res) => {
      return res.data
    })
    .catch((err) => {
      console.error(err)
    })

  for (const dataObj of resDatosTable) {
    dataTable.push(dataObj)
  }

  return dataTable
}

export default getLugares

```

- Se debe exportar debidamente cada una de las funciones las cuales consumirán los endpoints para obtener y procesar los datos.
- Luego se crearon cada uno de los componentes en este caso son 2, un modal para ingreso y edición de datos, asimismo una tabla para completar la estructura del crud , en esta tabla se agregaron las funciones de editar y eliminar

```

1 import { useState, useEffect } from "react";
2 import { Table, Button } from "react-bootstrap";
3 import {
4   getFiscalias,
5   obtenerFiscaliaPorId,
6   deleteFiscalias,
7 } from "../services/Fiscalia";
8 import ModalAgregarFiscalia from "../common/AgregarFiscaliaModal.jsx";
9
10 const TablaFiscalias = () => {
11   const [fiscalias, setFiscalias] = useState([]);
12   const [mostrarModal, setMostrarModal] = useState(false);
13   const [dataFiscalia, setDataFiscalia] = useState(null);
14
15   useEffect(() => {
16     obtenerFiscalias();
17   }, []);
18
19   const obtenerFiscalias = async () => {
20     const datosFiscalias = await getFiscalias();
21     setFiscalias(datosFiscalias);
22   };
23
24   const abrirModal = () => {
25     setDataFiscalia(null);
26     setMostrarModal(true);
27   };
28
29   const cerrarModal = () => {
30     setMostrarModal(false);
31   };
32
33   const editarFiscalia = async (id) => {
34     // Obtener los datos actuales de la fila seleccionada
35     const response = await obtenerFiscaliaPorId(id);
36
37     if (response.success) {
38       // Establecer los datos de la fila seleccionada
39       setDataFiscalia(response.data);
40       setMostrarModal(true);
41     } else {
42       // Mostrar un mensaje de error si no se pudieron obtener los datos
43       alert(response.mensaje);
44     }
45   };
46 };

```


- Para la Tabla se aprovecharon los hooks de react useState y useEffect, para poder setear los valores mas fácilmente, y llamar el ciclo para que actualice los datos de la tabla con unas acciones especificas como lo son, guardar registro, editar registro, eliminar registro.

```
const eliminarFiscalia = async (id) => {
  // Obtener los datos actuales de la fila seleccionada
  const response = await deleteFiscalias(id);
  obtenerFiscalias();
};

return (
  <div>
    <Button onClick={abrirModal}>Nueva Fiscalía</Button>
    <Table striped bordered hover>
      <thead>
        <tr>
          <th>ID</th>
          <th>Nombre</th>
          <th>Ubicación</th>
          <th>Teléfono</th>
          <th colspan={2}>Acciones</th>
        </tr>
      </thead>
      <tbody>
        {fiscalias.map((fiscalia) => (
          <tr key={fiscalia.codigoFiscalia}>
            <td>{fiscalia.codigoFiscalia}</td>
            <td>{fiscalia.nombre}</td>
            <td>{fiscalia.lugar.nombre}</td>
            <td>{fiscalia.numeroTelefono}</td>
            <td>
              <button onClick={() => editarFiscalia(fiscalia.codigoFiscalia)}>
                Editar
              </button>
            </td>
            <td>
              <button
                onClick={() => eliminarFiscalia(fiscalia.codigoFiscalia)}
              >
                Eliminar
              </button>
            </td>
          </tr>
        ))}
      </tbody>
    </Table>
    <ModalAgregarFiscalia
      show={mostrarModal}
      onHide={cerrarModal}
      recargarDatos={obtenerFiscalias}
      dataFiscal={dataFiscalia}
    ></ModalAgregarFiscalia>
  </div>
)
```

- En esta misma tabla se manda a llamar el componente de modal el cual se muestra a continuación

```

frontend > src > common > AgregarFiscaliaModal.jsx > ModalAgregarFiscalia > useEffect() callback

1  import React, { useState, useEffect } from "react";
2  import { Modal, Button, Form } from "react-bootstrap";
3  import getLugares from "../services/Lugares";
4  import { saveFiscalia, updateFiscalia } from "../services/Fiscalia";
5
6  const ModalAgregarFiscalia = ({ show, onHide, recargarDatos, dataFiscal }) => {
7    const [codigoFiscalia, setCodigoFiscalia] = useState("");
8    const [nombre, setNombre] = useState("");
9    const [ubicacion, setUbicacion] = useState("");
10   const [numeroTelefono, setNumeroTelefono] = useState("");
11   const [lugares, setLugares] = useState([]);
12
13   useEffect(() => {
14     const getApi = async () => {
15       const lugares = await getLugares();
16       setLugares(lugares);
17     };
18     getApi();
19   }, []);
20
21   useEffect(() => {
22     if (dataFiscal !== null) {
23       setCodigoFiscalia(dataFiscal.codigoFiscalia);
24       setNombre(dataFiscal.nombre);
25       setUbicacion(dataFiscal.ubicacion);
26       setNumeroTelefono(dataFiscal.numeroTelefono);
27     } else {
28       setCodigoFiscalia("");
29       setNombre("");
30       setUbicacion("");
31       setNumeroTelefono("");
32     }
33   }, [dataFiscal]);
34
35   const handleSubmit = async (event) => {
36     event.preventDefault();
37
38     try {
39       let respuesta = {};
40       if (dataFiscal === null) {
41         respuesta = await saveFiscalia({
42           codigoFiscalia,
43           nombre,
44           ubicacion,
45           numeroTelefono,
46         });

```

- Al igual que en la tabla en este componente se aprovechan los hooks de react useState y useEffect para obtener los datos de los lugares al momento de utilizar la modal, ya que utiliza un dropdown para seleccionar la ubicación de la fiscalía.

- Una vez configurado todo se manda a llamar la tabla en App.jsx

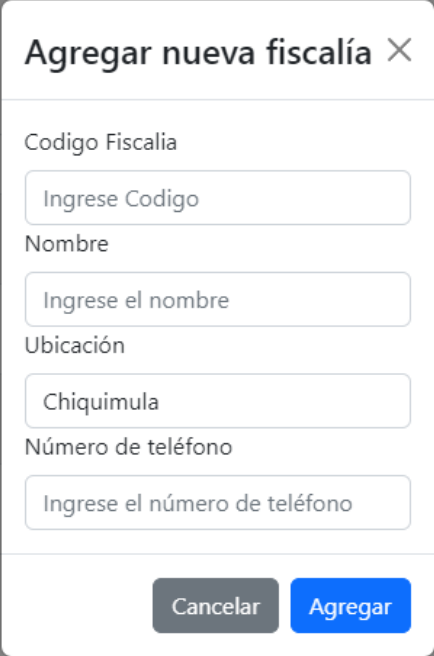
```
1  import './App.css'
2  import TablaFiscalias from './common/Tabla.jsx'
3
4
5  function App() {
6
7
8      return (
9          <div>
10             <TablaFiscalias></TablaFiscalias>
11          </div>
12      )
13  }
14
15  export default App
16
```

A continuación, el producto final

Nueva Fiscalía

ID	Nombre	Ubicación	Teléfono	Acciones	
202	Pueblosanto	Chiquimula	32654879	Editar	Eliminar
203	Chiquifer	Zacapa	65458798	Editar	Eliminar
204	Tapachula	Huehuetenango	64587987	Editar	Eliminar
205	Santosano	Quetzaltenango	32654879	Editar	Eliminar

- Como se puede observar, tenemos una tabla que obtiene los datos de las fiscalías, aquí mismo tenemos las opciones para agregar una nueva fiscalía, editar una existente o eliminarlas.



The image shows a modal window titled "Agregar nueva fiscalía" with a close button (X). The modal contains four input fields: "Codigo Fiscalia" (placeholder: "Ingrese Codigo"), "Nombre" (placeholder: "Ingrese el nombre"), "Ubicación" (placeholder: "Chiquimula"), and "Número de teléfono" (placeholder: "Ingrese el número de teléfono"). At the bottom of the modal are two buttons: "Cancelar" and "Agregar". In the background, a table is visible with columns "ID", "Nombre", "Ubicación", "Teléfono", and "Acciones". The "Acciones" column contains "Eliminar" buttons for each row.

- Aquí tenemos la Modal para ingresar nuevas fiscalías.

ID	Nombre	Acciones
202	Pueblosanto	Eliminar
203	Chiquifer	Eliminar
204	Tapachula	Eliminar
205	Santosano	Eliminar

Agregar nueva fiscalía ✕

Codigo Fiscalia

Nombre

Ubicación

Número de teléfono

Cancelar
Agregar

- Se agregarán los datos

Nueva Fiscalía					
ID	Nombre	Ubicación	Teléfono	Acciones	
202	Pueblosanto	Chiquimula	32654879	Editar	Eliminar
203	Chiquifer	Zacapa	65458798	Editar	Eliminar
204	Tapachula	Huehuetenango	64587987	Editar	Eliminar
205	Santosano	Quetzaltenango	32654878	Editar	Eliminar
206	La Ponderosa	Chiquimula	324687	Editar	Eliminar

- El dato se creo correctamente

- Al presionar en Editar cambiará el mensaje y la descripción del botón de aceptación y cualquier cambio será registrado

The image shows a modal window titled "Editar Fiscalia" with a close button (X). It contains four input fields: "Codigo Fiscalia" (value: 206), "Nombre" (value: La Ponderosa), "Ubicación" (value: San Marcos), and "Número de teléfono" (value: 46547898). At the bottom are "Cancelar" and "Editar" buttons. In the background, a table with columns "ID", "Nombre", and "Acciones" is visible, showing rows for IDs 202 through 206 with "Eliminar" buttons.

- Para este ejemplo se cambió el número de teléfono y la ubicación.

Nueva Fiscalía					
ID	Nombre	Ubicación	Teléfono	Acciones	
202	Pueblosanto	Chiquimula	32654879	Editar	Eliminar
203	Chiquifer	Zacapa	65458798	Editar	Eliminar
204	Tapachula	Huehuetenango	64587987	Editar	Eliminar
205	Santosano	Quetzaltenango	32654878	Editar	Eliminar
206	La Ponderosa	San Marcos	46547898	Editar	Eliminar

- Como pueden ver se actualizo correctamente la ubicación y el número de teléfono.

- Por ultimo al seleccionar Eliminar, borrará el registro seleccionado, para ejemplo eliminaremos el que se creo recién, el código 206

Nueva Fiscalía					
ID	Nombre	Ubicación	Teléfono	Acciones	
202	Pueblosanto	Chiquimula	32654879	Editar	Eliminar
203	Chiquifer	Zacapa	65458798	Editar	Eliminar
204	Tapachula	Huehuetenango	64587987	Editar	Eliminar
205	Santosano	Quetzaltenango	32654878	Editar	Eliminar

- La tabla se actualizo al momento de presionar eliminar también, ejecuto la función de delete y obtuvo los datos actualizados.