TECNOLOGICO DE ESTUDIOS SUPERIORES DE ECATEPEC

División de ingeniería en sistemas computacionales

Integrantes:

Albor Ramos Jovani

Rojas Jiménez Alejandro

Sánchez Martínez Diego Alan

BD para dispositivos móviles

Griselda Cortes Barrera

Proyecto final(Manual Backend)

Manual backend pasos a seguir

Antes de comenzar debemos instalar estos comandos para que todo funcione sin problemas:

npm install express sequelize mysql2 cors body-parser

npm install --save-dev nodemon

y el siguiente para inicializar un servidor:

npm init -y

con esto dicho ahora si continuamos.

Creamos un archivo js llamado **server.js** donde importaremos la configuración de express, así como asignar el puerto a utilizar para la conexión a la base de datos mostrando un mensaje en consola que nos permita saber si el servidor esta activo.

```
const app = require('./app/app'); // Importa la configuración de Express
const port = process.env.PORT || 3000;

// Iniciar el servidor
Tabnine | Edit | Test | Explain | Document | Ask
app.listen(port, () => {
    console.log(`Servidor corriendo en http://localhost:${port}`);
});
```

Lo siguiente es crear nuevamente un archivo js llamado **database.js** donde mediante el nombre de la base de datos, el usuario y la contraseña de la instancia en nuestro caso de MySQL se pueda hacer la conexión con nuestra base de datos.

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

// Configuración de Sequelize
const sequelize = new Sequelize('eventos', 'root', 'root', {
    host: 'localhost',
    dialect: 'mysql',
});

module.exports = sequelize;
```

Ahora pasaremos a crear los modelos de cada una de las tablas de nuestra base de datos tomando los campos agregados en esta y colocando el tipo de dato a utilizar esto para que tome los valores de las tablas sin ningún problema todo esto colocado en un archivo js llamado como cada tabla en este caso la primera es la tabla de **Conferencias.js** repitiendo este proceso dos veces más ya que tenemos la tabla de Talleres y Concursos que mantendrán la misma estructura.

Tabla Conferencias

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

// Definir el modelo "Usuario"
const Conferencias = sequelize.define('Conferencias', {
    idconf:{type: DataTypes.STRING,primaryKey: true,},
    titulo: { type: DataTypes.STRING, },
    fecha: { type: DataTypes.STRING, },
    lugar: { type: DataTypes.STRING, },
    duracion: { type: DataTypes.STRING, },
    descripcion: { type: DataTypes.STRING,  },
    imgcon: { type: DataTypes.STRING, },

}, {
    tableName: 'Conferencias', // Nombre de la tabla en MySQL
    timestamps: false,     // Desactiva campos createdAt y updatedAt
    freezeTableName: true,
});

module.exports = Conferencias;
```

Tabla Talleres

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

// Definir el modelo "Usuario"
const Talleres = sequelize.define('Talleres', {
    idtar:{type: DataTypes.STRING,primaryKey: true,},
    titulo: { type: DataTypes.STRING, },
    fecha: { type: DataTypes.STRING, },
    lugar: { type: DataTypes.STRING, },
    duracion: { type: DataTypes.STRING, },
    descripcion: { type: DataTypes.STRING,  },
    imgtar: { type: DataTypes.STRING, },

}, {
    tableName: 'Talleres', // Nombre de la tabla en MySQL
    timestamps: false,     // Desactiva campos createdAt y updatedAt
    freezeTableName: true,
});

module.exports = Talleres;
```

Tabla Concursos

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

// Definir el modelo "Usuario"
const Concursos = sequelize.define('Concursos', {
    idconc:{type: DataTypes.STRING,primaryKey: true,},
    titulo: { type: DataTypes.STRING, },
    fecha: { type: DataTypes.STRING, },
    lugar: { type: DataTypes.STRING, },
    duracion: { type: DataTypes.STRING, },
    descripcion: { type: DataTypes.STRING,  },
    imgconc: { type: DataTypes.STRING, },

}, {
    tableName: 'Concursos', // Nombre de la tabla en MySQL
    timestamps: false,     // Desactiva campos createdAt y updatedAt
    freezeTableName: true,
});

module.exports = Concursos;
```

Lo siguiente será la creación de las rutas para cada una de las tablas que de igual forma cada tabla ira en su respectivo archivo js en este caso el primero es **Conferenciasrout.js** aquí podremos hacer la obtención de los datos de la base de datos en

formato JSON que nos permitirá rescatar esa información al Frontend sin ningún problema esta estructura se repite dos veces más una para la tabla Talleres y la otra para Concursos cambiando sus respectivas variables.

Rutas Conferencias

```
const express = require('express');
const Conferencias = require('../models/Conferencias');

const router = express.Router();

// Endpoint para obtener todos los usuarios
Tabnine | Edit | Test | Explain | Document | Ask
router.get('/', async (req, res) => {
    try {
        const Conferenciasrout = await Conferencias.findAll(); // Consulta todos los registros
        res.json(Conferenciasrout); // Devuelve los datos en formato JSON
    } catch (error) {
        res.status(500).json({ error: 'Error al obtener los usuarios', detalle: error.message });
    }
});

module.exports = router;
```

Rutas Talleres

```
const express = require('express');
const Talleres = require('../models/Talleres');

const router = express.Router();

// Endpoint para obtener todos los usuarios
Tabnine | Edit | Test | Explain | Document | Ask
router.get('/', async (req, res) => {
    try {
        const Talleresrout = await Talleres.findAll(); // Consulta todos los registros
        res.json(Talleresrout); // Devuelve los datos en formato JSON
    } catch (error) {
        res.status(500).json({ error: 'Error al obtener los usuarios', detalle: error.message });
    }
});

module.exports = router;
```

Rutas Concursos

```
const express = require('express');
const Concursos = require('../models/Concursos');

const router = express.Router();

// Endpoint para obtener todos los usuarios
Tabnine | Edit | Test | Explain | Document | Ask
router.get('/', async (req, res) => {
    try {
        const Concursosrout = await Concursos.findAll(); // Consulta todos los registros
        res.json(Concursosrout); // Devuelve los datos en formato JSON
    } catch (error) {
        res.status(500).json({ error: 'Error al obtener los usuarios', detalle: error.message });
    }
});

module.exports = router;
```

Ya para finalizar nos queda el archivo **app.js** donde primeramente haremos uso de cors para poder hacer solicitudes entre el back y el front ya que de no ser usado tendremos problemas para el rescate de información lo siguiente es el formato en que se mostrara la información que como habíamos dicho usaremos el formato JSON, a su vez registraremos las rutas creadas anteriormente y para finalizar haremos la autenticación de la base de datos para saber si tuvimos una conexión exitosa y de no ser así mostrar un mensaje en consola.

```
const express = require('express');
const cors = require('cors');
const ConferenciasRoutes = require('./routes/Conferenciasrout');
const  uire('./routes/Talleresrout');
const  quire('./routes/Concursosrout');
const app = express();

app.use(cors({
    origin: 'http://localhost:8100'  // Permite solo solicitudes desde este origen
}));
// Middleware
app.use(express.json());

// Registrar rutas
app.use('/Conferencias', ConferenciasRoutes);
app.use('/Talleres', TalleresRoutes);
app.use('/Concursos', ConcursosRoutes);
```

const app: Express

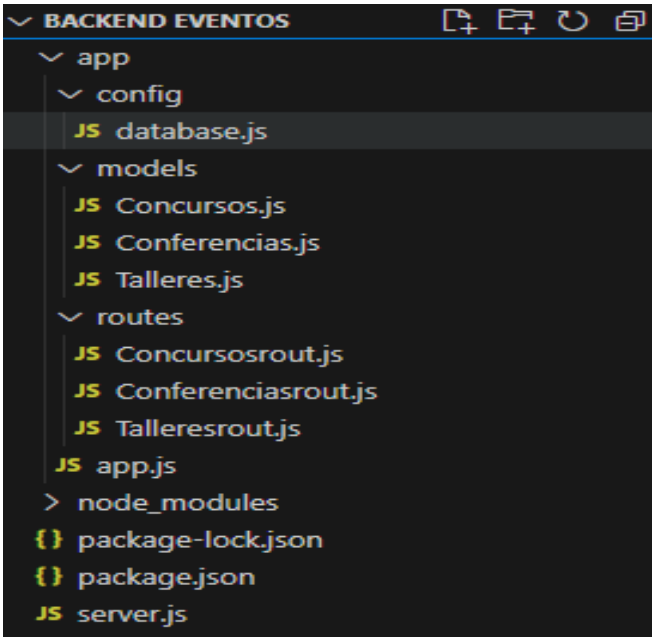
```
const sequelize = require('./config/database');

sequelize.authenticate()
  Tabnine | Edit | Test | Explain | Document | Ask
  .then(() => {
    console.log('Conexión a la base de datos exitosa');
  })
  Tabnine | Edit | Test | Explain | Document | Ask
  .catch((error) => {
    console.error('No se pudo conectar a la base de datos:', error);
  });

module.exports = app;
```

NOTA: cada módulo necesita ser exportado para que otro modulo pueda usarlo sin problema esto mediante **module.exports =**

Y estos serían todos los archivos que debemos tener.



Y bueno ahora procederemos a ver como iniciar el servidor y ver que todo funcione mediante el puerto que elegimos y que este muestre la información respectiva.

Para poder iniciar el servidor haremos uso del comando **node server.js** así como se muestra

```
PS C:\Users\joval\Downloads\proyectos ionic\Backend Eventos> node server.js
Servidor corriendo en http://localhost:3000
Executing (default): SELECT 1+1 AS result
Conexión a la base de datos exitosa
```

Ahora mediante el puerto y las rutas mostraremos la información de la base de datos en formato JSON.

localhost:3000/concursos


```
[
    {
        "idconc": 1001,
        "titulo": "Concurso Angular",
        "fecha": "15/02/2022",
        "lugar": "Salon 1",
        "duracion": "2 horas y media",
        "descripcion": "Concurso sobre Angular",
        "imgconc": "https://imgur.com/2DQ7sqR.png"
    },
    {
        "idconc": 1002,
        "titulo": "Concurso sobre Ionic",
        "fecha": "15/02/2022",
        "lugar": "Salon 2",
        "duracion": "1 hora y media",
        "descripcion": "Concurso sobre Ionic",
        "imgconc": "https://imgur.com/RO26z8B.jpg"
    },
    {
        "idconc": 1003,
        "titulo": "Concurso de React native ",
        "fecha": "15/02/2022",
        "lugar": "Salon 3",
        "duracion": "2 horas",
        "descripcion": "Concurso sobre React native",
        "imgconc": "https://imgur.com/Pontwpx.jpg"
    }
]
```


localhost:3000/talleres


```
[
    {
        "idtar": 100,
        "titulo": "Taller de Github",
        "fecha": "14/02/2022",
        "lugar": "Salon 4",
        "duracion": "2 horas y media",
        "descripcion": "Taller sobre Github",
        "imgtar": "https://imgur.com/EkyQD8n.png"
    },
    {
        "idtar": 101,
        "titulo": "Taller de angular",
        "fecha": "10/02/2022",
        "lugar": "Salon 5",
        "duracion": "1 hora y media",
        "descripcion": "Taller sobre angular",
        "imgtar": "https://imgur.com/wzk75G4.png"
    },
    {
        "idtar": 102,
        "titulo": "Taller de ionic",
        "fecha": "11/02/2022",
        "lugar": "Salon 6",
        "duracion": "1 hora",
        "descripcion": "Uso basico de ionic",
        "imgtar": "https://imgur.com/4MjrQCK.png"
    }
]
```

localhost:3000/conferencias


```
[
    {
        "idconf": 1,
        "titulo": "Angular 10",
        "fecha": "10/02/2022",
        "lugar": "Salon 1",
        "duracion": "3 horas",
        "descripcion": "Conferencia sobre angular 10",
        "imgcon": "https://imgur.com/9NcA0wa.jpg"
    },
    {
        "idconf": 2,
        "titulo": "Ionic 5",
        "fecha": "11/02/2022",
        "lugar": "Salon 2",
        "duracion": "4 horas",
        "descripcion": "Conferencia sobre ionic 5",
        "imgcon": "https://imgur.com/IIWNj0V.jpg"
    },
    {
        "idconf": 3,
        "titulo": "React native",
        "fecha": "12/02/2022",
        "lugar": "Salon 3",
        "duracion": "2 horas",
        "descripcion": "Conferencia sobre React native",
        "imgcon": "https://imgur.com/n5zDgJx.jpg"
    }
]
```


Esto nos quiere decir que tanto la base de datos como el back ya están listos para usarse sin ningún problema.