Actividad En Linea Modelos

conexion.js

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

// Configuración de la conexión a la base de datos
// Aquí usamos SQLite como base de datos por su simplicidad
// El archivo 'database.sqlite' se creará automáticamente
const sequelize = new Sequelize({
    dialect: 'sqlite',
    storage: './database.sqlite',
    logging: false // Desactivamos los logs de SQL para mantener la consola limpia
});

// Exportamos la conexión para que los modelos puedan usarla
module.exports = sequelize;
```

- Importamos Sequelize que es una biblioteca ORM (Object-Relational Mapping) que nos permite trabajar con bases de datos de manera más sencilla
- Creamos una nueva conexión usando SQLite como base de datos
- storage: './database.sqlite' indica dónde se guardará el archivo de la base de datos
- logging: false desactiva los mensajes de SQL en la consola
- Exportamos la conexión para que otros archivos puedan usarla

proveedores.js

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

const proveedores = sequelize.define('proveedores', {
    id: { type: DataTypes.INTEGER, primaryKey: true },
    nombre: { type: DataTypes.STRING },
    direccion: { type: DataTypes.STRING }
}, {
    timestamps: false
});

module.exports = proveedores;
```

- Importamos DataTypes para definir los tipos de datos de las columnas
- Importamos la conexión que creamos anteriormente
- Definimos el modelo proveedores con tres campos:
- id: número entero que sirve como clave primaria
- nombre: texto para el nombre del proveedor
- direccion: texto para la dirección
- timestamps: false indica que no queremos campos automáticos de fecha
- Exportamos el modelo para usarlo en otros archivos

index.js

```
// Importamos los modelos y la conexión
const sequelize = require('./conexion');
const Proveedores = require('./proveedores');
const Articulos = require('./articulos');
const Clientes = require('./clientes');
const Empleados = require('./empleados');

// Función para agregar algunos datos de ejemplo
async function agregarDatosEjemplo() {
    try {
        // Esperamos a que la conexión esté lista y las tablas sincronizadas
        await sequelize.authenticate();
        console.log('Conexión establecida correctamente.');

        // Sincronizamos los modelos con la base de datos
        await sequelize.sync({ force: true });
        console.log('Modelos sincronizados con la base de datos.');

        // Crear un proveedor
        const proveedor = await Proveedores.create({
            id: 1,
            nombre: 'Proveedor Ejemplo',
            direccion: 'Calle Principal 123'
        });
        console.log('Proveedor creado:', proveedor.toJSON());

        // [Código similar para crear artículos, clientes y empleados...]

    } catch (error) {
        console.error('Error al agregar datos de ejemplo:', error);
    }
}

// Ejecutamos la función
agregarDatosEjemplo();
```

- Importamos todos los modelos y la conexión
- Creamos una función asíncrona agregarDatosEjemplo()
- Dentro de la función:

1. Verificamos la conexión con authenticate()
2. Sincronizamos los modelos (creamos las tablas) con sync()
3. Creamos registros de ejemplo usando el método create()
4. Manejamos cualquier error que pueda ocurrir

- La función se ejecuta automáticamente al final del archivo

package.json

```
{
  "name": "sistema-gestion",
  "version": "1.0.0",
  "description": "Sistema de gestión con modelos de datos",
  "main": "index.js",
  "scripts": {
    "start": "node index.js"
  },
  "dependencies": {
    "sequelize": "^6.35.2",
    "sqlite3": "^5.1.7"
  }
}
```

- Define el nombre y versión del proyecto
- Especifica el archivo principal (index.js)
- Define los comandos disponibles (en este caso npm start ejecutará index.js)
- Lista las dependencias necesarias:
- sequelize: para el manejo de la base de datos
- sqlite3: el driver para SQLite