app.js – Archivo principal de la aplicación

Este archivo configura y lanza el servidor principal usando Express. Es el punto de entrada de toda la aplicación.

Funciones clave del archivo:

1. **Importación de módulos:**

Se importa Express para crear el servidor web, y se cargan los archivos de rutas correspondientes a cada recurso:

```
const express = require("express");
const clienteRoutes = require("./routes/clientes");
const proveedorRoutes = require("./routes/proveedores");
const articuloRoutes = require("./routes/articulos");
const empleadoRoutes = require("./routes/empleados");
```

2. **Inicialización de la app:**

Se crea una instancia de la app y se define el puerto:

```
const app = express();
const PORT = 3000;
```

3. **Middleware para procesar JSON:**

Este middleware permite que la app entienda cuerpos de solicitud en formato JSON:

```
app.use(express.json());
```

4. **Registro de rutas:**

Se asocian las rutas importadas a sus respectivos prefijos de URL. Por ejemplo, todas las rutas para clientes comenzarán con /clientes:

```
app.use("/clientes", clienteRoutes);
app.use("/proveedores", proveedorRoutes);
app.use("/articulos", articuloRoutes);
app.use("/empleados", empleadoRoutes);
```

5. **Inicio del servidor:**

Se lanza el servidor en el puerto 3000, mostrando un mensaje en consola:

```
app.listen(PORT, () => {
  console.log(`Servidor corriendo en http://localhost:${PORT}`);
});
```

app.js configura el servidor Express, habilita el uso de JSON en solicitudes, enlaza las rutas específicas para cada tipo de entidad (clientes, proveedores, artículos, empleados) y finalmente arranca el servidor en el puerto 3000.

config/database.js – Configuración de la base de datos

Este archivo define la conexión a la base de datos usando Sequelize, un ORM para Node.js.

Funciones clave del archivo:

1. **Importación de Sequelize:**

Se importa la clase Sequelize, que se usará para crear la conexión:

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

2. **Inicialización de la conexión:**

Se crea una instancia de Sequelize usando SQLite como base de datos. En este caso, se define que los datos se almacenarán en un archivo local:

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

3. **Exportación de la instancia:**

La instancia de conexión se exporta para poder ser usada en otros archivos, como en los modelos:

```
module.exports = sequelize;
```

config/database.js configura y exporta la conexión a una base de datos SQLite utilizando Sequelize. Esta conexión será usada por todos los modelos de la aplicación para interactuar con la base de datos.

models/index.js – Registro e inicialización de modelos

Este archivo se encarga de centralizar la carga de todos los modelos definidos en la aplicación y sincronizarlos con la base de datos.

Funciones clave del archivo:

1. **Importación de dependencias y conexión:**

Se importa Sequelize y la conexión previamente configurada:

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

2. **Carga de modelos:**

Se cargan los modelos (Cliente, Proveedor, Articulo, Empleado) pasando la instancia de Sequelize y los tipos de datos que se usarán:

```
const Cliente = require("./Cliente")(sequelize,
Sequelize.DataTypes);
const Proveedor = require("./Proveedor")(sequelize,
Sequelize.DataTypes);
const Articulo = require("./Articulo")(sequelize,
Sequelize.DataTypes);
const Empleado = require("./Empleado")(sequelize,
Sequelize.DataTypes);
```

Cada uno de estos modelos es una función que recibe sequelize y DataTypes, y retorna el modelo definido.

3. **Sincronización con la base de datos:**

Se asegura que las tablas estén creadas en la base de datos según los modelos definidos:

```
sequelize.sync();
```

4. **Exportación:**

Se exportan la conexión y los modelos para que puedan ser usados desde otros archivos:

```
module.exports = {
  sequelize,
  Cliente,
  Proveedor,
  Articulo,
  Empleado
};
```

models/index.js centraliza la importación y configuración de todos los modelos, los sincroniza con la base de datos y los exporta junto con la conexión sequelize, facilitando su uso en otras partes del proyecto.

models/Cliente.js – Modelo del cliente

Este archivo define el modelo Cliente utilizando Sequelize, que representa la estructura de la tabla Clientes en la base de datos.

Funciones clave del archivo:

1. **Definición del modelo:**

Se exporta una función que recibe la instancia de Sequelize y los tipos de datos:

```
module.exports = (sequelize, DataTypes) => {
```

2. **Estructura del modelo:**

Se define un modelo llamado Cliente con los siguientes campos:

```
const Cliente = sequelize.define("Cliente", {
  id: { type: DataTypes.INTEGER, autoIncrement: true, primaryKey: true },
  nombre: { type: DataTypes.STRING, allowNull: false },
  correo: { type: DataTypes.STRING, allowNull: false },
  telefono: { type: DataTypes.STRING, allowNull: false },
  direccion: { type: DataTypes.STRING, allowNull: false }
});
```

models/Cliente.js define un modelo Sequelize para representar clientes, con campos básicos como nombre, correo, teléfono y dirección. Este modelo se traduce a una tabla en la base de datos y se utiliza para realizar operaciones CRUD.

models/Proveedor.js – Modelo del proveedor

Este archivo define el modelo Proveedor, que representa a los proveedores registrados en el sistema.

Estructura del modelo:

```
const Proveedor = sequelize.define("Proveedor", {
  id: { type: DataTypes.INTEGER, autoIncrement: true, primaryKey: true },
  nombre: { type: DataTypes.STRING, allowNull: false },
  direccion: { type: DataTypes.STRING, allowNull: false }
});
```

models/Proveedor.js define la tabla de proveedores con dos campos de información clave: nombre y dirección. El campo id es la clave primaria.

models/Articulo.js – Modelo del artículo

Este modelo representa productos o artículos disponibles en el sistema.

Estructura del modelo:

```
const Articulo = sequelize.define("Articulo", {
  id: { type: DataTypes.INTEGER, autoIncrement: true, primaryKey: true },
  descripcion: { type: DataTypes.STRING, allowNull: false },
  precio: { type: DataTypes.FLOAT, allowNull: false },
  existencia: { type: DataTypes.INTEGER, allowNull: false }
});
```

models/Articulo.js define los campos de la tabla de artículos, que incluyen descripción, precio y existencia en inventario.

models/Empleado.js – Modelo del empleado

Este modelo representa a los empleados de la empresa.

Estructura del modelo:

```
const Empleado = sequelize.define("Empleado", {
  id: { type: DataTypes.INTEGER, autoIncrement: true, primaryKey: true },
  nombre: { type: DataTypes.STRING, allowNull: false },
  telefono: { type: DataTypes.STRING, allowNull: false },
  fecha_de_nacimiento: { type: DataTypes.DATEONLY, allowNull: false },
  sueldo: { type: DataTypes.FLOAT, allowNull: false }
});
```

models/Empleado.js representa empleados con datos personales y laborales como nombre, teléfono, fecha de nacimiento y sueldo.