Escuela:

Facultad de Informática

Carrera:

Licenciatura en informática

Hecho por:

Gastelum Landeros Antonio

Grupo: 2-1

Código Explicado:

Primera parte Articulos:

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

- Esto exporta una función que recibe:
  - sequelize: la instancia de la conexión con la base de datos.
  - DataTypes: un objeto que contiene todos los tipos de datos que Sequelize puede usar (INTEGER, STRING, FLOAT, BOOLEAN, etc.).

sequelize.define("Articulo", {...})

- Crea un modelo llamado "Articulo".
- Sequelize asume que esto representa una tabla en la base de datos llamada Articulos (agrega "s" por defecto, aunque eso se puede configurar).
- El segundo parámetro es un objeto con los campos (o columnas) del modelo.

Campos definidos:

```
id: {

  type: DataTypes.INTEGER,

  autoIncrement: true,

  primaryKey: true

}
```

- Campo numérico (INTEGER).
- autoIncrement: se incrementa automáticamente al insertar nuevos artículos.
- primaryKey: es la clave primaria, es decir, identificador único de cada fila.

```
descripcion: {

  type: DataTypes.STRING,

  allowNull: false

}
```

- Cadena de texto (STRING).
- allowNull: false: este campo no puede quedar vacío (es obligatorio).

precio: {

type: DataTypes.FLOAT,

allowNull: false

}

- Número decimal (FLOAT), útil para representar precios con centavos.
- También es obligatorio.

existencia: {

type: DataTypes.INTEGER,

allowNull: false

}

- Entero que representa cuántas unidades hay en existencia (stock).
- También obligatorio**.**

Return Articulo

- Devuelve el modelo creado para que pueda ser usado en otras partes del proyecto.

Segunda parte Cliente:

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

- Exporta una función que define un modelo Sequelize.
- Recibe:
  - sequelize: instancia de conexión con la base de datos.
  - DataTypes: objeto que permite definir el tipo de cada campo (STRING, INTEGER, etc.).

sequelize.define("Cliente", { ... })

- Crea un modelo Sequelize llamado "Cliente".
- Sequelize generará una tabla llamada Clientes (en plural, por convención).
- Dentro del objeto se definen los campos de la tabla.

Campos definidos:

id: {

type: DataTypes.INTEGER,

autoIncrement: true,

primaryKey: true

}

- Clave primaria de tipo entero.
- autoIncrement: aumenta automáticamente con cada nuevo cliente.
- primaryKey: lo convierte en el identificador único.

nombre: {

type: DataTypes.STRING,

allowNull: false

}

- Texto.
- No puede ser nulo. Es obligatorio indicar el nombre del cliente.

correo: {

type: DataTypes.STRING,

allowNull: false

}

- Texto (cadena de caracteres).
- Guarda el correo electrónico del cliente.
- También es obligatorio.

telefono: {

type: DataTypes.STRING,

allowNull: false

}

- Texto para almacenar el número telefónico.
- Aunque el teléfono es numérico, es común guardarlo como STRING por:
  - Códigos de área (+52, 011).
  - Guiones (555-1234) o espacios.
- También es obligatorio.

direccion: {

type: DataTypes.STRING,

allowNull: false

}

- Texto.
- Guarda la dirección del cliente.
- También obligatorio.

Return Cliente

- Devuelve el modelo Cliente para poder usarlo en el resto del proyecto.

Tercera parte Empleado:

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

- Exporta una función que define el modelo.
- Recibe:
  - sequelize: instancia de conexión a la base de datos.
  - DataTypes: objeto con los tipos de datos que Sequelize puede usar (INTEGER, STRING, FLOAT, DATEONLY, etc.).

sequelize.define("Empleado", { ... })

- Crea un modelo Sequelize llamado "Empleado", que Sequelize asociará a una tabla llamada Empleados (plural automático).
- Dentro del objeto se definen los campos de la tabla.

Campos del modelo:

id: {

type: DataTypes.INTEGER,

autoIncrement: true,

primaryKey: true

}

- Entero autoincremental.
- Clave primaria (identificador único del empleado).

nombre: {

type: DataTypes.STRING,

allowNull: false

}

- Cadena de texto.
- Obligatorio (allowNull: false).

telefono: {

type: DataTypes.STRING,

allowNull: false

}

- Cadena de texto para el número de teléfono.
- Se usa STRING en lugar de INTEGER para poder incluir:
  - Códigos de país (+52, 011)
  - Guiones o espacios (555-1234)
- Obligatorio.

fecha_de_nacimiento: {

type: DataTypes.DATEONLY,

allowNull: false

}

- Tipo de dato DATEONLY guarda solo la fecha (sin hora), útil para fechas de nacimiento.

- Obligatorio.

sueldo: {

type: DataTypes.FLOAT,

allowNull: false

}

- Número decimal con decimales (FLOAT), ideal para sueldos.
- Obligatorio.

Return Empleado

- Devuelve el modelo Empleado para que pueda ser usado en otras partes del proyecto (rutas, controladores, etc.).

Cuarta parten Proveedor:

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

- Exporta una función que define el modelo.
- Recibe dos parámetros:
  - sequelize: instancia activa de la conexión con la base de datos.
  - DataTypes: objeto con los tipos de datos que Sequelize permite usar (STRING, INTEGER, DATE, etc.).

sequelize.define("Proveedor", {...})

- Define el modelo Sequelize con el nombre "Proveedor".
- Sequelize automáticamente creará la tabla Proveedors (sí, el plural es imperfecto en inglés, pero puedes corregirlo con freezeTableName: true si quieres que quede como Proveedor).
- Dentro del objeto se indican los campos (columnas) del modelo.

Campos definidos:

id: {

type: DataTypes.INTEGER,

autoIncrement: true,

primaryKey: true

}

- Entero.
- Clave primaria (primaryKey: true).
- Se incrementa automáticamente con cada nuevo registro (autoIncrement: true).

nombre: {

type: DataTypes.STRING,

allowNull: false

}

- Texto.
- Campo obligatorio (allowNull: false).
- Guarda el nombre del proveedor (por ejemplo: "Lácteos del Sur").

direccion: {

type: DataTypes.STRING,

allowNull: false

}

- Texto.
- Campo obligatorio.
- Guarda la dirección física del proveedor.

Return Proveedor

- Devuelve el modelo Proveedor para usarlo en otras partes del proyecto (rutas, consultas, etc.).