Explicación del Código

Importaciones

```
const express = require('express')

const bodyParser = require('body-parser')

const monedas = require('./monedas') // Modelo para interactuar con la base de datos
```

- Usa **Express** para crear el servidor.
- Usa **body-parser** para leer datos del cuerpo de las peticiones POST.
- Importa el modelo monedas.js, que probablemente define una tabla con conversiones.

Inicialización del servidor

```
const app = express()

const puerto = 3000


app.use(bodyParser.json())


app.listen(puerto, () => {
   console.log('servicio iniciado')
})
```

- El servidor escucha en el puerto 3000.
- Permite que el cuerpo de las peticiones se reciba como JSON.

Ruta /convertir (POST)

```
app.post('/convertir', async (req, res) => {
   const { origen, destino, cantidad } = req.body;


   const data = await monedas.findOne({ where: { origen, destino } });


   if (!data) {
```

```
    res.sendStatus(404)
  }

  const { valor } = data;
  const resultado = cantidad * valor;

  res.send({ origen, destino, cantidad, resultado })
})
```

- Recibe los datos: moneda de origen, destino y cantidad.
- Busca la tasa de conversión en la base de datos.
- Si no la encuentra, responde con 404.
- Si la encuentra, multiplica la cantidad por el valor y devuelve el resultado.

Ruta /modificar (POST)

```
app.post('/modificar', async (req, res) => {
  const { origen, destino, valor } = req.body;

  const data = await monedas.findOne({ where: { origen, destino } });

  if (!data) {
    const createConversion = await monedas.create({ origen, destino, valor });
    res.send("Conversión registrada con éxito!")
  } else {
    if (data.valor !== valor) {
      await data.update({ valor });
      return res.json({ mensaje: 'Registro actualizado', data });
    } else {
      return res.send("El valor es el mismo que ya está registrado")
    }
```

```
    }
})
```

- Permite **agregar o actualizar** tasas de conversión.
- Si no existe la conversión, la crea.
- Si ya existe pero el valor es diferente, lo actualiza.
- Si es igual, lo notifica.

Ruta /monedas (GET)

```
app.get('/monedas', async (req, res) => {
    const data = await monedas.findAll();
    res.send(data);
})
```

- Devuelve todas las conversiones registradas en la base de datos.

Explicación del Código (monedas.js)

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

- Importa DataTypes desde Sequelize para definir los tipos de datos.
- Importa una instancia de conexión a la base de datos desde el archivo conexion.js (que aún no hemos visto, pero probablemente existe).

Definición del modelo

```
const monedas = sequelize.define('monedas', {
    id: { type: DataTypes.INTEGER, primaryKey: true },
    origen: { type: DataTypes.STRING },
    destino: { type: DataTypes.STRING },
    valor: { type: DataTypes.DOUBLE }
}, {
    timestamps: false
```

});

Define una tabla llamada monedas con los siguientes campos:

Campo	Tipo	Descripción
id	INTEGER	Clave primaria (no auto-incremental)
origen	STRING	Código de la moneda de origen (ej: USD)
destino	STRING	Código de la moneda de destino (ej: EUR)
valor	DOUBLE	Tasa de conversión

timestamps: false indica que Sequelize **no** agregará campos automáticos como createdAt y updatedAt.

Exportación

module.exports = monedas;

- Exporta el modelo para poder usarlo en otros archivos (como en app.js).