Tarea: Desarrollo de un Backend para una Aplicación Bancaria

Objetivo

Desarrollar una aplicación backend utilizando Node.js con Express, PostgreSQL y TypeORM, que permita la gestión de usuarios y transacciones bancarias. Los estudiantes deberán implementar autenticación, registro de usuarios, transferencias entre usuarios, consulta de transacciones, envío de correos de confirmación y el uso de Pug para la generación de correos.

Requisitos Técnicos

Tecnologías a utilizar:

- Node.js con Express.
- PostgreSQL como base de datos.
- TypeORM para la gestión de la base de datos.
- Nodemailer para el envío de correos electrónicos.
- Pug para la plantilla del correo de confirmación.
- JSON Web Tokens (JWT) para la autenticación.
- Bcrypt para el cifrado de contraseñas.

Funcionalidades requeridas:

- Registro de usuarios con validación de datos.
- Inicio de sesión mediante JWT.
- Transferencias entre usuarios autenticados.
- Consulta del historial de transacciones del usuario.
- Envío de correo de confirmación de registro utilizando Pug.

Base de Datos:

- Se deberá definir un esquema adecuado utilizando TypeORM, con tablas para usuarios y transacciones.

Entrega:

- Código en un repositorio de GitHub.

Documentación de la API con una tabla de endpoints.

Capturas de pantalla del correo de confirmación.

Tabla de Endpoints

Método	Endpoint	Descripción
POST	/api/auth/register	Registra un nuevo usuario
POST	/api/auth/login	Inicia sesión y devuelve un JWT
GET	/api/users/me	Obtiene información del usuario autenticado
POST	/api/transactions	Realiza una transferencia a otro usuario
GET	/api/transactions	Obtiene el historial de transacciones
GET	/api/transactions/:id	Obtiene los detalles de una transacción

Consideraciones

Implementar manejo de errores adecuado.

Proteger los endpoints con autenticación JWT.

Utilizar validadores para los datos de entrada.

Configurar Nodemailer para el envío de correos.

Modelo Entidad-Relación

Entidad: Usuario

Campo	Tipo	Descripción
id	UUID	Identificador único del usuario
name	VARCHAR(100)	Nombre del usuario
email	VARCHAR(100)	Correo electrónico (debe ser único)
password	TEXT	Contraseña encriptada
account_number	VARCHAR(20)	Número de cuenta bancario generado aleatoriamente
balance	DECIMAL(10,2)	Saldo disponible
created_at	TIMESTAMP	Fecha de creación

Entidad: Transacción

Campo	Tipo	Descripción
id	UUID	Identificador único de la transacción

sender_id	UUID	Referencia al usuario que envía dinero
receiver_id	UUID	Referencia al usuario que recibe dinero
amount	DECIMAL(10,2)	Monto de la transferencia
transaction_date	TIMESTAMP	Fecha de la transacción

Script para generar numero de cuenta aleatorio

```
function generateAccountNumber() {
    const timestamp = Date.now().toString().slice(-8); // Últimos 8 dígitos del timestamp
    const randomDigits = Math.floor(100000 + Math.random() * 900000).toString(); // 6 dígitos
    return timestamp + randomDigits; // Número de cuenta de 14 dígitos
}
```