npm i --save @nestjs/serve-static

----------------------------------------------------------------

**Comando que genera los recursos para cada modulo del proyecto**

nest g resource usuarios

---------------------------------------------------------------------------

npm i --save @nestjs/typeorm typeorm mysql2

----------------------------------------------------------------

npm install class-validator

----------------------------------------------------------------

npm i bcrypt

-------------------------------------------------------------------

npm i --save-dev @types/bcrypt

-------------------------------------------------------------------

TypeOrmModule.forRoot({

type: 'mysql',

host: 'localhost',

port: 3306,

username: 'root',

password: '\*\*',

database: 'rapicreditos',

entities: [\_\_dirname + '/\*/.entity{.ts,.js}'],

synchronize: true,

}),

**Genero modulos sin DTO y Entities**

nest g module auth

nest g service auth

nest g controller auth

**1. Access Token (Token de acceso)**

Es un token de corta duración que se usa para autenticar y autorizar al usuario en las peticiones al servidor o a la API.

Contiene información del usuario y permisos (a veces en formato JWT).

Cada vez que el cliente hace una petición protegida, envía el access token para demostrar que está autenticado.

Se expira rápido (por ejemplo, 15 minutos o 1 hora) para minimizar riesgos si alguien roba el token.

Ejemplo:

Cuando un usuario inicia sesión, recibe un access token.

Para consultar su perfil o hacer acciones, envía ese token en las cabeceras.

Si el token expiró, no puede seguir usando ese token.

2. Refresh Token (Token de refresco)

Es un token de mayor duración (por ejemplo, días o semanas).

Se usa exclusivamente para pedir al servidor un nuevo access token cuando el original expiró.

El refresh token no se envía en cada petición, solo en la llamada para renovar el access token.

Generalmente es más seguro, se almacena de forma segura (por ejemplo, HttpOnly cookies o almacenamiento seguro).

Puede ser invalidado manualmente (logout) o por el servidor.

¿Por qué el refresh token tiene mayor tiempo de expiración?

Para no obligar al usuario a iniciar sesión constantemente.

El access token expira rápido para limitar el riesgo en caso de robo.

**El refresh token permite obtener nuevos access tokens sin volver a pedir las credenciales (usuario y contraseña).**

Si el refresh token durara poco, la experiencia de usuario sería mala porque tendría que reingresar muchas veces.

Pero si dura mucho, debe manejarse con cuidado, porque si alguien lo roba puede generar access tokens nuevos.

**¿Cuándo se usan?**

**Access token:** Se usa en todas las peticiones protegidas al backend para validar la sesión y permisos.

**Refresh token:** Se usa cuando el access token expiró y queremos obtener uno nuevo sin que el usuario vuelva a loguearse.

**Resumen rápido:**

Token Duración Uso principal Seguridad

Access Token Corta (minutos) Autorizar peticiones al API Se renueva frecuentemente, riesgo bajo

Refresh Token Larga (días/semanas) Obtener nuevos access tokens Se almacena seguro, riesgo mayor si se pierde

Si quieres, te puedo dar un ejemplo práctico con código o flujos típicos. ¿Te interesa?