**Ejemplo de Microservicios en NestJS con Base de Datos PostgreSQL**

Este ejemplo muestra una arquitectura basada en microservicios que utiliza **PostgreSQL** como base de datos. Los microservicios son los siguientes:

1. **User Service**: Administra usuarios y se conecta directamente a PostgreSQL.
2. **Order Service**: Administra órdenes y también se conecta a PostgreSQL.
3. **Notification Service**: Envía notificaciones cuando se crea una orden, utilizando mensajes entre microservicios.
4. **API Gateway**: Proporciona un punto único de entrada para interactuar con los microservicios.

Usaremos **gRPC** como protocolo de comunicación entre los microservicios.

**Arquitectura General**

* **User Service** y **Order Service** se conectan a PostgreSQL con sus propias tablas.
* El **Notification Service** escucha eventos relacionados con órdenes a través de gRPC.
* Un **API Gateway** expone endpoints HTTP para interactuar con los servicios.

**Preparar PostgreSQL**

Crea una base de datos llamada microservices\_example y define las tablas para usuarios y órdenes:

CREATE DATABASE microservices\_example;

-- Tabla de usuarios

CREATE TABLE users (

id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) NOT NULL UNIQUE

);

-- Tabla de órdenes

CREATE TABLE orders (

id SERIAL PRIMARY KEY,

user\_id INT NOT NULL,

product VARCHAR(100) NOT NULL,

amount NUMERIC(10, 2) NOT NULL,

created\_at TIMESTAMP DEFAULT NOW(),

FOREIGN KEY (user\_id) REFERENCES users (id)

);

**1. User Service**

**Instalar dependencias:**

nest new user-service

cd user-service

npm install @nestjs/typeorm typeorm pg

**Archivo: src/main.ts**

import { NestFactory } from '@nestjs/core';

import { AppModule } from './app.module';

async function bootstrap() {

const app = await NestFactory.create(AppModule);

await app.listen(3001);

console.log('User Service is running on port 3001');

}

bootstrap();

**Archivo: src/app.module.ts**

import { Module } from '@nestjs/common';

import { TypeOrmModule } from '@nestjs/typeorm';

import { User } from './user.entity';

import { UserService } from './user.service';

import { UserController } from './user.controller';

@Module({

imports: [

TypeOrmModule.forRoot({

type: 'postgres',

host: 'localhost',

port: 5432,

username: 'your\_username',

password: 'your\_password',

database: 'microservices\_example',

entities: [User],

synchronize: true,

}),

TypeOrmModule.forFeature([User]),

],

controllers: [UserController],

providers: [UserService],

})

export class AppModule {}

**Archivo: src/user.entity.ts**

import { Entity, Column, PrimaryGeneratedColumn } from 'typeorm';

@Entity()

export class User {

@PrimaryGeneratedColumn()

id: number;

@Column()

name: string;

@Column({ unique: true })

email: string;

}

**Archivo: src/user.service.ts**

import { Injectable } from '@nestjs/common';

import { InjectRepository } from '@nestjs/typeorm';

import { Repository } from 'typeorm';

import { User } from './user.entity';

@Injectable()

export class UserService {

constructor(

@InjectRepository(User)

private userRepository: Repository<User>,

) {}

findAll() {

return this.userRepository.find();

}

create(user: Partial<User>) {

const newUser = this.userRepository.create(user);

return this.userRepository.save(newUser);

}

}

**Archivo: src/user.controller.ts**

import { Controller, Get, Post, Body } from '@nestjs/common';

import { UserService } from './user.service';

@Controller('users')

export class UserController {

constructor(private readonly userService: UserService) {}

@Get()

findAll() {

return this.userService.findAll();

}

@Post()

create(@Body() user: { name: string; email: string }) {

return this.userService.create(user);

}

}

**2. Order Service**

**Instalar dependencias:**

nest new order-service

cd order-service

npm install @nestjs/typeorm typeorm pg

**Archivo: src/app.module.ts**

Configura la conexión a PostgreSQL y registra la entidad Order.

import { Module } from '@nestjs/common';

import { TypeOrmModule } from '@nestjs/typeorm';

import { Order } from './order.entity';

import { OrderService } from './order.service';

import { OrderController } from './order.controller';

@Module({

imports: [

TypeOrmModule.forRoot({

type: 'postgres',

host: 'localhost',

port: 5432,

username: 'your\_username',

password: 'your\_password',

database: 'microservices\_example',

entities: [Order],

synchronize: true,

}),

TypeOrmModule.forFeature([Order]),

],

controllers: [OrderController],

providers: [OrderService],

})

export class AppModule {}

**Archivo: src/order.entity.ts**

import { Entity, Column, PrimaryGeneratedColumn } from 'typeorm';

@Entity()

export class Order {

@PrimaryGeneratedColumn()

id: number;

@Column()

user\_id: number;

@Column()

product: string;

@Column('decimal')

amount: number;

@Column({ type: 'timestamp', default: () => 'CURRENT\_TIMESTAMP' })

created\_at: Date;

}

**Archivo: src/order.service.ts**

import { Injectable } from '@nestjs/common';

import { InjectRepository } from '@nestjs/typeorm';

import { Repository } from 'typeorm';

import { Order } from './order.entity';

@Injectable()

export class OrderService {

constructor(

@InjectRepository(Order)

private orderRepository: Repository<Order>,

) {}

findAll() {

return this.orderRepository.find();

}

create(order: Partial<Order>) {

const newOrder = this.orderRepository.create(order);

return this.orderRepository.save(newOrder);

}

}

**Archivo: src/order.controller.ts**

import { Controller, Get, Post, Body } from '@nestjs/common';

import { OrderService } from './order.service';

@Controller('orders')

export class OrderController {

constructor(private readonly orderService: OrderService) {}

@Get()

findAll() {

return this.orderService.findAll();

}

@Post()

create(@Body() order: { user\_id: number; product: string; amount: number }) {

return this.orderService.create(order);

}

}

**3. Notification Service**

**Descripción**

Este servicio se suscribe a eventos de órdenes utilizando gRPC para enviar notificaciones.

**4. API Gateway**

El API Gateway unifica los servicios y proporciona un único punto de entrada para interactuar con ellos.

**Instalar dependencias**

nest new api-gateway

cd api-gateway

npm install @nestjs/microservices

**Archivo: src/main.ts**

import { NestFactory } from '@nestjs/core';

import { AppModule } from './app.module';

async function bootstrap() {

const app = await NestFactory.create(AppModule);

await app.listen(3000);

console.log('API Gateway is running on port 3000');

}

bootstrap();

**Archivo: src/app.module.ts**

Configura los clientes para conectarse a los servicios de usuario y órdenes mediante gRPC o TCP.

import { Module } from '@nestjs/common';

@Module({

imports: [],

controllers: [],

providers: [],

})

export class AppModule {}

**Pruebas**

1. Levanta los servicios:
2. npm run start
3. Interactúa con el **API Gateway** en http://localhost:3000.

Este ejemplo puede ampliarse con más servicios y lógica según las necesidades de tu sistema. Si deseas detalles adicionales, avísame.