САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа

Выполнил:

Чебан Илья

Группа К3341

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

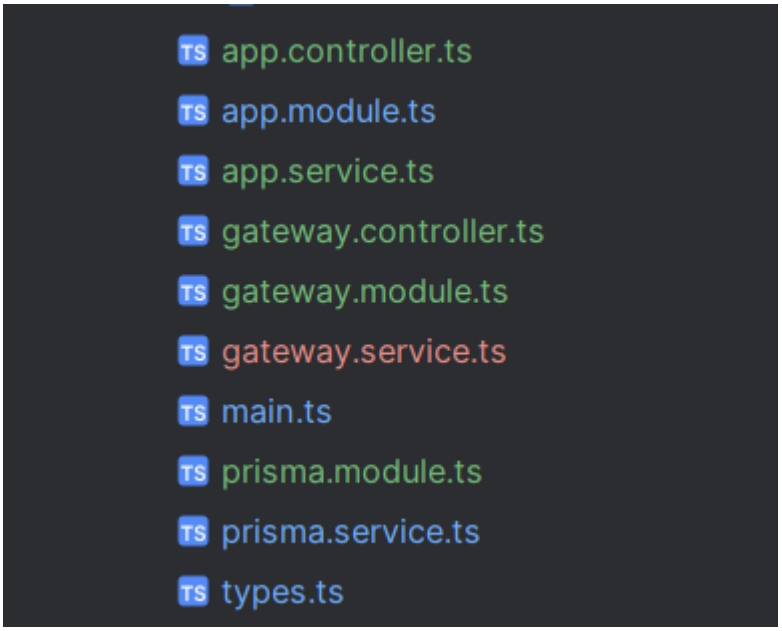
Задача

Реализация Микросервисной Архитектуры

Ход работы

Нужно реализовать Микросервисы на nest.js + PrismaORM

Структура:



Gateway:

```
import { Module } from '@nestjs/common';
import { ClientsModule, Transport } from '@nestjs/microservices';
import { GatewayService } from './gateway.service';
import { UsersGatewayController } from './gateway.controller';
import { PrismaModule } from './prisma.module';
import { UsersModule } from './users/users.module';
import { MessageModule } from './message/message.module';
import { PropertyModule } from './property/property.module';
import { DealModule } from './deal/deal.module';

@Module({
 imports: [
   ClientsModule.register([
     PrismaModule,
     UsersModule,
     MessageModule,
```

```
    PropertyModule,
    DealModule,

    {
      name: 'USERS_SERVICE',
      transport: Transport.TCP,
      options: { port: 8877 },
    },
    {
      name: 'PROPERTY_SERVICE',
      transport: Transport.TCP,
      options: { port: 8878 },
    },
    {
      name: 'MESSAGE_SERVICE',
      transport: Transport.TCP,
      options: { port: 8879 },
    },
    {
      name: 'DEAL_SERVICE',
      transport: Transport.TCP,
      options: { port: 8880 },
    },
  ]),
 ],
 providers: [GatewayService],
 controllers: [UsersGatewayController],
})
export class GatewayModule {}
```

```
import { Injectable } from '@nestjs/common';
import {
 ClientProxy,
 ClientProxyFactory,
 Transport,
} from '@nestjs/microservices';

@Injectable()
export class GatewayService {
 private usersClient: ClientProxy;
 private propertyClient: ClientProxy;
 private messageClient: ClientProxy;
 private dealClient: ClientProxy;

 constructor() {
   this.usersClient = ClientProxyFactory.create({
     transport: Transport.TCP,
     options: { host: 'users-service', port: 8877 },
   });

   this.propertyClient = ClientProxyFactory.create({
     transport: Transport.TCP,
     options: { host: 'property-service', port: 8878 },
```

```
    });

    this.messageClient = ClientProxyFactory.create({
      transport: Transport.TCP,
      options: { host: 'message-service', port: 8879 },
    });

    this.dealClient = ClientProxyFactory.create({
      transport: Transport.TCP,
      options: { host: 'deal-service', port: 8880 },
    });
  }

  async sendToUsers(pattern: string, data: any): Promise<any> {
    return this.usersClient.send({ cmd: pattern }, data).toPromise();
  }

  async sendToProperty(pattern: string, data: any): Promise<any> {
    return this.propertyClient.send({ cmd: pattern }, data).toPromise();
  }

  async sendToMessage(pattern: string, data: any): Promise<any> {
    return this.messageClient.send({ cmd: pattern }, data).toPromise();
  }

  async sendToDeal(pattern: string, data: any): Promise<any> {
    return this.dealClient.send({ cmd: pattern }, data).toPromise();
  }
}
```

```
import { Body, Controller, Post } from '@nestjs/common';
import { GatewayService } from './gateway.service';

@Controller('api/users')
export class UsersGatewayController {
  constructor(private readonly gatewayService: GatewayService) {}

  @Post('create')
  createUser(@Body() data: any) {
    return this.gatewayService.sendToUsers('user_create', data);
  }

  @Post(':id/update')
  updateUser(@Body() data: any) {
    return this.gatewayService.sendToUsers('user_update', data);
  }

  @Post(':id/delete')
  deleteUser(@Body() data: any) {
    return this.gatewayService.sendToUsers('user_delete', data);
  }

  @Post('create')
```

```
  createProperty(@Body() data: any) {
    return this.gatewayService.sendToProperty('property_create', data);
  }

  @Post(':id/update')
  updateProperty(@Body() data: any) {
    return this.gatewayService.sendToProperty('property_update', data);
  }

  @Post(':id/delete')
  deleteProperty(@Body() data: any) {
    return this.gatewayService.sendToProperty('property_delete', data);
  }

  @Post('create')
  createMessage(@Body() data: any) {
    return this.gatewayService.sendToMessage('message_create', data);
  }

  @Post(':id/update')
  updateMessage(@Body() data: any) {
    return this.gatewayService.sendToMessage('message_update', data);
  }

  @Post(':id/delete')
  deleteMessage(@Body() data: any) {
    return this.gatewayService.sendToMessage('message_delete', data);
  }

  @Post('create')
  createDeal(@Body() data: any) {
    return this.gatewayService.sendToDeal('deal_create', data);
  }

  @Post(':id/update')
  updateDeal(@Body() data: any) {
    return this.gatewayService.sendToDeal('deal_update', data);
  }

  @Post(':id/delete')
  deleteDeal(@Body() data: any) {
    return this.gatewayService.sendToDeal('deal_delete', data);
  }
}
```

Вывод: Реализовали Микросервисную Архитектуру