**Hướng dẫn sử dụng Backend PAPAOPTOM**

* 1. **Cấu trúc source code**

**C:.**

**│ app.controller.spec.ts**

**│ app.controller.ts**

**│ app.module.ts**

**│ app.service.ts**

**│ main.ts**

**│**

**├───auth -> Module chứng thực**

**│ │ auth.module.ts**

**│ │**

**│ ├───decorators**

**│ │ roles.decorator.ts**

**│ │**

**│ ├───guards**

**│ │ roles.guard.ts**

**│ │**

**│ ├───interfaces**

**│ │ jwt-payload.interface.ts**

**│ │ refresh-token.interface.ts**

**│ │**

**│ ├───schemas**

**│ │ refresh-token.schema.ts**

**│ │**

**│ ├───services**

**│ │ auth.service.spec.ts**

**│ │ auth.service.ts**

**│ │**

**│ └───strategies**

**│ jwt.strategy.ts**

**│**

**├───forsage -> Module lấy dữ liệu từ forsage**

**│ │ axios.config.ts**

**│ │ forsage.module.ts**

**│ │**

**│ └───services**

**│ forsage.service.spec.ts**

**│ forsage.service.ts**

**│**

**├───order -> Module tạo đơn hàng**

**│ │ order.module.ts -> Quản lý các thành phần trong module**

**│ │**

**│ ├───dto -> Định nghĩa các Data Transfer Object để giao tiếp trong dự án**

**│ │ ├───req -> Định nghĩa các input nhận vào của GraphQL**

**│ │ │ BrandReq.ts**

**│ │ │ CategoryChildrenReq.ts**

**│ │ │ CategoryReq.ts**

**│ │ │ characteristicsReq.ts**

**│ │ │ CustomerReq.ts**

**│ │ │ GalleryReq.ts**

**│ │ │ OrderReq.ts**

**│ │ │ PaymentReq.ts**

**│ │ │ ProductReq.ts**

**│ │ │ ShippingReq.ts**

**│ │ │ SupplierReq.ts**

**│ │ │**

**│ │ └───res -> Định nghĩa cấu trúc trả về của GraphQL**

**│ │ OrderRes.ts**

**│ │**

**│ ├───interface -> Định nghĩa toan bộ schema của database**

**│ │ Bill.ts**

**│ │ Customer.ts**

**│ │ Order.ts**

**│ │ Payment.ts**

**│ │ Product.ts**

**│ │ Shipping.ts**

**│ │**

**│ ├───resolvers -> Expose một GrapQL gồm query và mutation**

**│ │ order.resolver.spec.ts**

**│ │ order.resolver.ts**

**│ │**

**│ ├───schemas -> Schema của Database**

**│ │ OrderSchema.ts**

**│ │**

**│ └───services -> Danh sách các dịch vụ sử dụng cho thành phần order**

**│ OrderService.ts**

**│**

**├───product -> Module quản lý sản phẩm**

**│ │ product.module.ts**

**│ │**

**│ ├───constants**

**│ │ ProductConstant.ts**

**│ │**

**│ ├───controllers -> Expose rest API nhưng trong dự án không sử dụng RestAPI nên không cần đào sâu phần này**

**│ │ product.controller.spec.ts**

**│ │ product.controller.ts**

**│ │**

**│ ├───dto -> Tham khảo phần order đã giải thích**

**│ │ │ create-ref-book-characteristics.dto.ts**

**│ │ │ ProductCsvDto.ts**

**│ │ │**

**│ │ ├───req**

**│ │ │ ProductFilterReq.ts**

**│ │ │**

**│ │ └───res**

**│ │ BrandRes.ts**

**│ │ CategoriesChildrenRes.ts**

**│ │ CategoriesRes.ts**

**│ │ CategoryChildrenRes.ts**

**│ │ CategoryRes.ts**

**│ │ CharacteristicRes.ts**

**│ │ GalleryRes.ts**

**│ │ ProductDetailRes.ts**

**│ │ ProductFilterRes.ts**

**│ │ ProductRes.ts**

**│ │ SupplierRes.ts**

**│ │**

**│ ├───helpers -> Định nghĩa các function hỗ trợ cho class service Product**

**│ │ ProductHelper.ts**

**│ │**

**│ ├───interfaces**

**│ │ Categories.ts**

**│ │ CronJob.ts**

**│ │ NotFoundProduct.ts**

**│ │ product.interface.ts**

**│ │ ref-book-characteristics.interface.ts**

**│ │ Sku.ts**

**│ │**

**│ ├───resolvers**

**│ │ └───product**

**│ │ product.resolver.spec.ts**

**│ │ product.resolver.ts**

**│ │**

**│ ├───schedules**

**│ │ ProductSchedule.ts**

**│ │**

**│ ├───schemas**

**│ │ CategoriesSchema.ts**

**│ │ CronJobSchema.ts**

**│ │ NormalizeProductSchema.ts**

**│ │ NotFoundProduct.ts**

**│ │ product.schema.ts**

**│ │ ref-book-characteristics.ts**

**│ │ SkuSchema.ts**

**│ │**

**│ └───services**

**│ product.service.spec.ts**

**│ product.service.ts**

**│**

**├───shared -> Thành phần dùng chung trong dự án**

**│ CurrencyRate.ts**

**│**

**└───user**

**│ user.module.ts**

**│**

**├───controllers**

**│ user.controller.spec.ts**

**│ user.controller.ts**

**│**

**├───dto**

**│ │ create-forgot-password.dto.ts**

**│ │ create-user.dto.ts**

**│ │ login-user.dto.ts**

**│ │ refresh-access-token.dto.ts**

**│ │ reset-password.dto.ts**

**│ │ verify-uuid.dto.ts**

**│ │**

**│ ├───req**

**│ │ AddressReq.ts**

**│ │ CardReq.ts**

**│ │ ContactReq.ts**

**│ │ CustomersReq.ts**

**│ │**

**│ └───res**

**│ AddressRes.ts**

**│ CardRes.ts**

**│ ContactRes.ts**

**│ CustomerDtoRes.ts**

**│ CustomerRes.ts**

**│ DeletedAddressRes.ts**

**│**

**├───interfaces**

**│ Customer.ts**

**│ forgot-password.interface.ts**

**│ user.interface.ts**

**│**

**├───resolvers**

**│ └───customer**

**│ customer.resolver.spec.ts**

**│ customer.resolver.ts**

**│**

**├───schemas**

**│ CustomerSchema.ts**

**│ forgot-password.schema.ts**

**│ user.schema.ts**

**│**

**└───services**

**user.service.spec.ts**

**user.service.ts**

* 1. **Giải thích một số decorator**

1. @Module({ => Khai báo một module (1)  
    imports: [  
    MongooseModule.*forFeature*([  
    { name: 'NotFoundProduct', schema: ***NotFoundProductSchema*** },  
    ]), => Khai báo một collection trong mongoDB (2)  
    MongooseModule.*forFeatureAsync*([  
    {  
    name: 'sku',  
    useFactory: () => {  
    const schema = ***SkuSchema***;  
    schema.plugin(mongoosePaginate);  
    return schema;  
    },  
    },  
    ]),  
    MongooseModule.*forFeature*([{ name: 'CronJob', schema: ***CronJobSchema*** }]),  
    MongooseModule.*forFeatureAsync*([  
    {  
    name: 'Product',  
    useFactory: () => {  
    const schema = ***NormalizeProductSchema***;  
    schema.plugin(mongoosePaginate);  
    return schema;  
    },  
    },  
    ]),  
    MongooseModule.*forFeatureAsync*([  
    {  
    name: 'ForsageProduct',  
    useFactory: () => {  
    const schema = ***ProductSchema***;  
    schema.plugin(mongoosePaginate);  
    return schema;  
    },  
    },  
    ]),  
    MongooseModule.*forFeatureAsync*([  
    {  
    name: "Categories",  
    useFactory: () => {  
    const schema = ***CategoriesSchema***;  
    schema.plugin(mongoosePaginate);  
    return schema;  
    }  
    }  
    ]),  
    MongooseModule.*forFeature*([  
    { name: 'RefbookCharacteristics', schema: ***RefBookCharacteristics*** },  
    ]),  
    ForsageModule,  
    HttpModule,  
    ],  
    providers: [ -> Khai báo các service sử dụng (3)  
    ProductService,  
    ProductResolver,  
    ProductSchedule,  
    CurrencyRate,  
    ProductHelper,  
    ],  
    controllers: [ProductController], -> Khai báo các controller để expose RestAPI (4)  
   })  
   export class ProductModule {} -> Định nghĩa tên của module (5)

**1.3. Định nghĩa Schema cho mongo db**

import \* as mongoose from "mongoose";  
  
export const ***CronJobSchema*** = new mongoose.Schema({  
 times: {  
 type: ***Number***,  
 default: 0  
 },  
 status: {  
 type: ***String***,  
 default: "INITIAL"  
 },  
 start: {  
 type: ***Date***,  
 default: ***Date***.now  
 },  
 end: {  
 type: ***Date***,  
 default: ***Date***.now  
 },  
 domain: {  
 type: ***String***,  
 default: "PRODUCT"  
 }  
}, { versionKey: false, timestamps: true})

**Bước 2: Định nghĩa 1 đối tượng trong code chứa trong folder interface**

import { Document } from "mongoose";  
  
export interface Categories extends Document {  
 id: string;  
 title: string;  
 slug: string;  
 icon: string;  
 children: Array<any>;  
}

**Bước 3: Vào module khai báo trong phần imports khai báo giống (1) như hình dưới**

@Module({  
 imports: [  
 MongooseModule.*forFeature*([ -> Khai báo giống như trên (1)  
 { name: 'NotFoundProduct', schema: ***NotFoundProductSchema*** },  
 ]),  
 MongooseModule.*forFeatureAsync*([  
 {  
 name: 'sku',  
 useFactory: () => {  
 const schema = ***SkuSchema***;  
 schema.plugin(mongoosePaginate);  
 return schema;  
 },  
 },  
 ]),  
 MongooseModule.*forFeature*([{ name: 'CronJob', schema: ***CronJobSchema*** }]),  
 MongooseModule.*forFeatureAsync*([  
 {  
 name: 'Product',  
 useFactory: () => {  
 const schema = ***NormalizeProductSchema***;  
 schema.plugin(mongoosePaginate);  
 return schema;  
 },  
 },  
 ]),  
 MongooseModule.*forFeatureAsync*([  
 {  
 name: 'ForsageProduct',  
 useFactory: () => {  
 const schema = ***ProductSchema***;  
 schema.plugin(mongoosePaginate);  
 return schema;  
 },  
 },  
 ]),  
 MongooseModule.*forFeatureAsync*([  
 {  
 name: "Categories",  
 useFactory: () => {  
 const schema = ***CategoriesSchema***;  
 schema.plugin(mongoosePaginate);  
 return schema;  
 }  
 }  
 ]),  
 MongooseModule.*forFeature*([  
 { name: 'RefbookCharacteristics', schema: ***RefBookCharacteristics*** },  
 ]),  
 ForsageModule,  
 HttpModule,  
 ],  
 providers: [  
 ProductService,  
 ProductResolver,  
 ProductSchedule,  
 CurrencyRate,  
 ProductHelper,  
 ],  
 controllers: [ProductController],  
})  
export class ProductModule {}

**1.4. Khai báo một GraphQL**

**Bước 1: Khai báo input (chú ý input là param điều kiện lấy dữ liệu được truyền vào) vào thư mục dto/req.**

import { Field, InputType, ***Int*** } from '@nestjs/graphql';  
  
@InputType() -> Định nghĩa một input object phần này là bắt buộc (1)  
export class ProductFilterReq {  
 @Field(() => ***String***, { nullable: true })-> Khai báo field request  
 readonly id?: string;  
 @Field(() => ***String***, { nullable: true })  
 readonly supplier?: string;  
 @Field(() => ***String***, { nullable: true })  
 readonly category?: string;  
 @Field(() => ***String***, { nullable: true })  
 readonly brand?: string;  
 @Field(() => ***String***, { nullable: true })  
 readonly name?: string;  
 @Field(() => ***Int***, { nullable: true })  
 readonly pageSize?: number;  
 @Field(() => ***Int***, { nullable: true })  
 readonly pageNumber?: number;  
}

**Bước 2: Khai báo response GraphQL**

import { Field, ***Int***, ObjectType } from '@nestjs/graphql';  
  
@ObjectType() -> Định nghĩa response trả về bắt buộc (1)  
export class BrandRes {  
 @Field(() => ***Int***, { nullable: true })  
 id?: number;  
 @Field(() => ***String***, { nullable: true })  
 name?: string;  
}

**Bước 3: Tạo một resolver trong thư mục resolvers**

import {Args, ***Int***, Query, Resolver} from '@nestjs/graphql';  
import {ProductFilterReq} from '../../dto/req/ProductFilterReq';  
import {ProductRes} from '../../dto/res/ProductRes';  
import {HttpStatus} from '@nestjs/common';  
import {ProductService} from '../../services/product.service';  
import \* as \_ from 'lodash';  
import {ProductDetailRes} from '../../dto/res/ProductDetailRes';  
import {ProductHelper} from '../../helpers/ProductHelper';  
import {CategoriesRes} from "../../dto/res/CategoriesRes";  
import {***InjectModel***} from "@nestjs/mongoose";  
import {***Model***} from "mongoose";  
import {Categories} from "../../interfaces/Categories";  
  
@Resolver() -> Khai báo resolvers bắt buộc (1)  
export class ProductResolver {  
 constructor(  
 private readonly productService: ProductService,  
 private readonly productHelper: ProductHelper,  
 @InjectModel("Categories") private readonly categoriesModel: ***Model***<Categories> -> Định nghĩa database muốn thao tác (2)  
 ) {}  
  
 @Query(() => [CategoriesRes]) -> Khai báo một query sữ dụng cho trường hợp lấy dữ liệu (3)  
 async categories(@Args({ name: "type", type: () => ***String***, nullable: true}) type?: string) {  
 const categories = await this.categoriesModel.find({});  
 return categories;  
 }  
 @Query(() => ProductDetailRes)  
 async getProduct(@Args({ name: 'slug', type: () => ***String*** }) slug: string) {  
 if (!slug) {  
 return {  
 code: HttpStatus.*BAD\_REQUEST*,  
 message: 'требуется пуля',  
 };  
 }  
 const product = await this.productService.findProductById(slug);  
  
 if (\_.isEmpty(product)) {  
 return {  
 code: HttpStatus.*NOT\_FOUND*,  
 message: 'никаких записей не найдено',  
 };  
 }  
  
 const characteristics = this.productHelper.convertPrice(product);  
  
 return {  
 code: HttpStatus.*OK*,  
 message: '1 результат найден',  
 data: {  
 ...product,  
 id: product?.id ?? 0,  
 vcode: product?.vcode ?? null,  
 name: product?.name ?? null,  
 quantity: product?.quantity ?? 0,  
 category: product?.category ?? {},  
 supplier: product?.supplier ?? {},  
 brand: product?.brand ?? {},  
 createdAt: product?.createdAt ?? new ***Date***(),  
 updatedAt: product?.updatedAt ?? new ***Date***(),  
 characteristics,  
 slug: product?.sku,  
 },  
 };  
 }  
  
 @Query(() => ProductRes)  
 async searchShoes(  
 @Args({ name: 'searchTerm', type: () => ***String***, nullable: true}) searchTerm?: string,  
 @Args({ name: 'pageSize', type: () => ***Int***, nullable: true}) pageSize?: number,  
 @Args({ name: 'pageNumber', type: () => ***Int***, nullable: true}) pageNumber?: number,  
 ) {  
 const product = await this.productService.searchShoes(  
 { searchTerm, pageSize, pageNumber });  
 const data = product?.docs.map((product) =>  
 this.productHelper.presentProduct(product),  
 );  
 if (product !== null) {  
 return {  
 code: HttpStatus.*OK*,  
 message: `${product?.totalDocs ?? 0} результат найден`,  
 data,  
 totalDocs: product?.totalDocs ?? 0,  
 hasPrevPage: product?.hasPrevPage ?? false,  
 hasNextPage: product?.hasNextPage ?? false,  
 totalPages: product?.totalPages ?? 0,  
 prevPage: product?.prevPage ?? 0,  
 nextPage: product?.nextPage ?? 0,  
 pageNumber: product?.page ?? 0,  
 pageSize: product?.limit ?? 0,  
 };  
 }  
 return {  
 code: HttpStatus.*NOT\_FOUND*,  
 message: 'результатов не найдено',  
 };  
 }  
  
 @Query(() => ProductRes)  
 async filterProduct(  
 @Args({ name: 'filter', type: () => ProductFilterReq })  
 filter?: ProductFilterReq,  
 ) {  
 // eslint-disable-next-line @typescript-eslint/ban-ts-comment  
 // @ts-ignore  
 const product = await this.productService.filterProduct(filter);  
 const data = product?.docs.map((product) =>  
 this.productHelper.presentProduct(product),  
 );  
 if (product !== null) {  
 return {  
 code: HttpStatus.*OK*,  
 message: `${product?.totalDocs ?? 0} результат найден`,  
 data,  
 totalDocs: product?.totalDocs ?? 0,  
 hasPrevPage: product?.hasPrevPage ?? false,  
 hasNextPage: product?.hasNextPage ?? false,  
 totalPages: product?.totalPages ?? 0,  
 prevPage: product?.prevPage ?? 0,  
 nextPage: product?.nextPage ?? 0,  
 pageNumber: product?.page ?? 0,  
 pageSize: product?.limit ?? 0,  
 };  
 }  
 return {  
 code: HttpStatus.*NOT\_FOUND*,  
 message: 'результатов не найдено',  
 };  
 }  
}

**Tài liệu tham khảo**

**[1].** [**https://docs.nestjs.com/graphql/quick-start**](https://docs.nestjs.com/graphql/quick-start) **-> nắm kỹ phần GraphQL**

**[2].** [**https://docs.nestjs.com/techniques/mongodb**](https://docs.nestjs.com/techniques/mongodb) **-> Nắm kỹ thao tác mongodb**

**[3].** [**https://docs.nestjs.com/fundamentals**](https://docs.nestjs.com/fundamentals) **-> nắm cấu trúc cơ bản của NestJs**