



NESTJS

<https://cybersoft.edu.vn/>

CYBERSOFT

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ MẠNG MÁY TÍNH



Mục Lục

01 Giới thiệu

02 Cài đặt

03 Module

04 Controller

05 Provider, dto

06 Database, cors, .env

07 JWT

08 Upload

09 Swagger

10 Docker compose

Why NestJS

Structure

Rest api

Typescript

Graphql

Documentation

Popular/Community

Module

CRUD

Giới thiệu

- Nest (NestJS) là một framework để xây dựng các ứng dụng phía máy chủ (Back end) NodeJS hiệu quả và có thể mở rộng.
- Được xây dựng và hỗ trợ đầy đủ TypeScript (nhưng vẫn cho phép các nhà phát triển viết mã bằng JavaScript).
- Kết hợp các yếu tố OOP (Object Oriented Programming), FP (Functional Programming), and FRP (Functional Reactive Programming).
- Ứng dụng có khả năng kiểm tra cao, có thể mở rộng, kết hợp chặt chẽ và dễ bảo trì.
- Cảm ứng từ Angular.

Cài đặt - structure

- CLI: `npm i -g @nestjs/cli`
- Kiểm tra: `nest -v`
- Tạo dự án mới: `nest new [tên dự án]`
- Lưu ý: Phiên bản NodeJs thích hợp (≥ 12 , trừ v13)

```
git clone https://github.com/nestjs/typescript-starter.git project
```

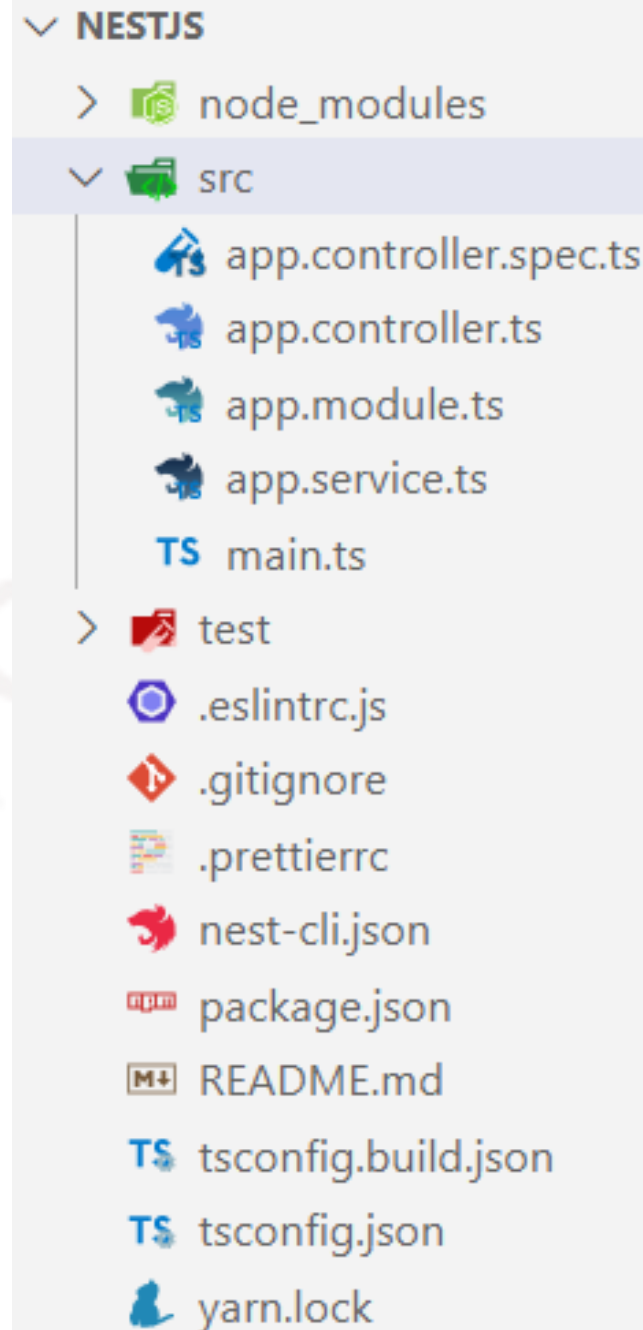
Structure and restruct

Run project: (auto build folder dist)

`yarn start` => Không debug

`yarn start:dev` => Debug (watching) giống nodemon

Tham khảo: <https://docs.nestjs.com/first-steps>



Module

- Imports: Tích hợp các module khác.
- Controllers: Điều hướng, Chứa các Restful API (GET, POST, PUT, DELETE,...)
- Provider (service): Gọi database, thực hiện các tác vụ tính toán,...

Lệnh tạo module nhanh: **nest g module [tên module] --no-spec**

```
..
import { Module } from '@nestjs/common';
import { AppController } from '../app.controller';
import { AppService } from '../app.service';
```

```
@Module({
  imports: [],
  controllers: [AppController],
  providers: [AppService],
})
```

Controller

- Định nghĩa API: **@Controller("user")**.
- Dữ liệu trả về (string, number, object,...).
- Lấy dữ liệu từ client: **@Req()**, **@Body()**, **@Param()**.
- Import service xử lý.
- Lệnh tạo nhanh:

nest g controller [tên module] --no-spec

```
import {
  Controller,
  Get, Post, Put,
  Req, Body, Param, HttpStatusCode } from '@nestjs/common';
import { Request } from "express";
import { AppService } from '../app.service';

@Controller("user")
export class AppController {
  constructor(private appService: AppService) { }

  @Get()
  getHello() {
    return "Get Hello"
  }
  @Get("getId/:id")
  getId(@Req() req: Request) {
    // req.params.id
    // req.body
    return "Get Hello id"
  }
  @Post("postId/:email")
  postHello(@Body() body, @Param("email") params) {
    console.log(params)

    return "Post Hello param"
  }
  @Put()
  @HttpCode(200)
  putHello() {
    return "Put Hello"
  }
}
```

Provider (service) - DTO

- Làm việc với database, xử lý tính toán,...
- Lệnh tạo nhanh:

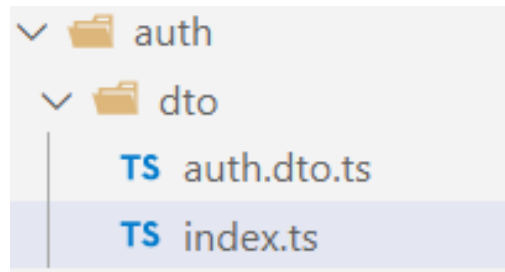
nest g service [tên module] --no-spec

- DTO: (Data transfer object)

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

```
@Injectable()
export class AppService {
  constructor() {}

  getHello(): string {
    return 'Hello World!';
  }
}
```



```
export interface UserDto {
  id: number;
  hoTen: string;
}
```

```
export * from './auth.dto';
```


Connect database – Prisma:

Setup prisma như buổi 10

Enable CORS:

```
import * as express from 'express';

async function bootstrap() {
  const app = await NestFactory.create(AppModule);
  app.enableCors();
  app.use(express.static("."))
  await app.listen(8080);
}
bootstrap();
```

Config .env:

yarn add @nestjs/config

```
import { ConfigModule } from '@nestjs/config';
```

```
@Module({
  imports: [
    ConfigModule.forRoot({
      isGlobal: true
    }),
  ],
})
```

```
import { ConfigService } from '@nestjs/config';
```

```
@Controller('auth')
export class AuthController {
  constructor(
    private config: ConfigService) {
  }
}
```

Authentication - JWT

Cài đặt:

yarn add @nestjs/config

yarn add @nestjs/passport passport passport-local

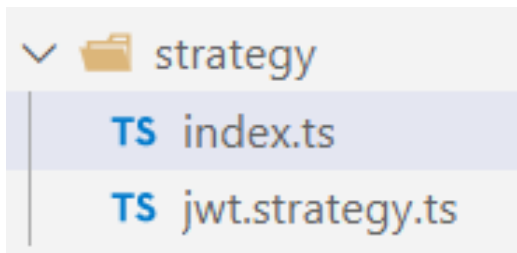
yarn add @nestjs/jwt passport-jwt

yarn add -D @types/passport-jwt

Tham khảo:

<https://docs.nestjs.com/security/authentication>

=> **Strategy, Guards and Structure:**



```
import { ExtractJwt, Strategy } from 'passport-jwt';
import { PassportStrategy } from '@nestjs/passport';
import { Injectable } from '@nestjs/common';
import { ConfigService } from '@nestjs/config';
```

```
@Injectable()
export class JwtStrategy extends
  PassportStrategy(Strategy) {
  constructor(config: ConfigService) {
    super({
      jwtFromRequest:
        ExtractJwt.fromAuthHeaderAsBearerToken(),
      ignoreExpiration: false,
      secretOrKey: config.get("SECRET_KEY"),
    });
  }

  async validate(payload: any) {

    return payload;
  }
}
```

Trả về sau khi kiểm tra token
req.user

Authentication - JWT

```
import { JwtModule } from '@nestjs/jwt';  
import { JwtStrategy } from './strategy/jwt.strategy';
```

```
@Module({  
  imports: [JwtModule.register({})],  
  controllers: [AuthController],  
  providers: [AuthService, JwtStrategy]  
})  
export class AuthModule { }
```

Guards

```
import { Controller, Req, UseGuards } from '@nestjs/common';  
import { AuthGuard } from '@nestjs/passport';
```

```
@UseGuards(AuthGuard("jwt"))  
@Post()  
signin(@Req() req: Request) {  
  console.log(req.user)  
  
  return "signin";  
}
```

Strategy

```
export class JwtStrategy extends  
  PassportStrategy(Strategy, "myjwt") {  
  constructor(config: ConfigService) {  
    super({  
      jwtFromRequest:
```

Upload

Cài đặt: **yarn add -D @types/multer**

```
import { Controller, Get, Param, Post, Req, UploadedFile, UseGuards, UseInterceptors } from '@nestjs/common';
import { diskStorage } from 'multer';
import { FileInterceptor } from '@nestjs/platform-express';
```

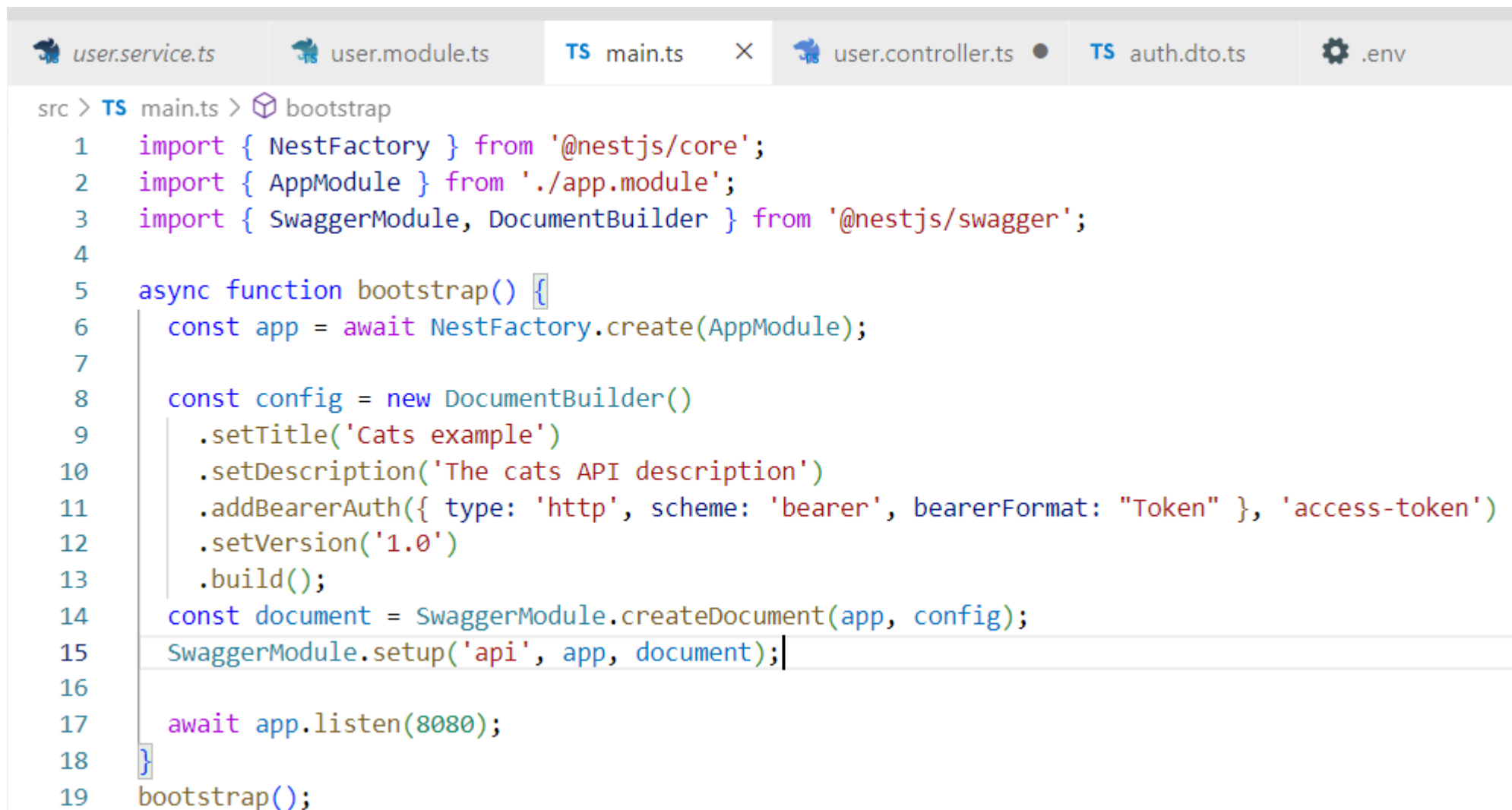
```
@Post('upload')
@UseInterceptors(FileInterceptor('file', {
  storage: diskStorage({
    destination: './img'
  },
  filename: (req, file, cb) => {
    console.log(file);
    cb(null, `abc.jpg`)
  }
}))
uploadFile(@UploadedFile() file) {
  console.log(file);
}
```

```
import { ServeStaticModule } from '@nestjs/serve-static';
```

```
@Module({
  imports: [
    ServeStaticModule.forRoot({
      rootPath: ".",
    }),
  ],
})
```

Swagger

Cài đặt: **yarn add @nestjs/swagger swagger-ui-express**



```
src > TS main.ts > bootstrap
1  import { NestFactory } from '@nestjs/core';
2  import { AppModule } from './app.module';
3  import { SwaggerModule, DocumentBuilder } from '@nestjs/swagger';
4
5  async function bootstrap() {
6    const app = await NestFactory.create(AppModule);
7
8    const config = new DocumentBuilder()
9      .setTitle('Cats example')
10     .setDescription('The cats API description')
11     .addBearerAuth({ type: 'http', scheme: 'bearer', bearerFormat: 'Token' }, 'access-token')
12     .setVersion('1.0')
13     .build();
14    const document = SwaggerModule.createDocument(app, config);
15    SwaggerModule.setup('api', app, document);
16
17    await app.listen(8080);
18  }
19  bootstrap();
```

Swagger

Setup: Controller, DTO

```
import { ApiBody, ApiParam, ApiBearerAuth, ApiTags } from '@nestjs/swagger';  
import { UserDto } from '../dto';
```

```
@ApiTags("user")  
@Controller('user')  
export class UserController {  
  constructor(private userService: UserService) {}  
  
  @ApiParam({ name: "hoTen" })  
  @ApiBody({ type: UserDto })  
  @ApiBearerAuth()  
  @UseGuards(AuthGuard("jwt"))  
  @Post("/:id")  
  signin(@Req() req: Request, @Body() body) {  
    console.log(body)  
  
    return "signin";  
  }  
}
```

```
import { ApiProperty } from '@nestjs/swagger';  
  
export class UserDto {  
  @ApiProperty({ description: "id", type: Number })  
  id: number;  
  
  @ApiProperty({ description: "họ tên", type: String })  
  hoTen: string;  
}
```

Docker compose, prisma

Dockerfile

```
FROM node:16

WORKDIR /app

COPY package*.json ./
COPY prisma ./prisma/

RUN npm install

COPY . .

RUN npm run build

EXPOSE 8080
CMD [ "npm", "run", "start:prod" ]
```

docker-compose.yml

.env

```
version: '3.8'
services:
  nest-api:
    container_name: nest-api
    build:
      context: .
    ports:
      - 3000:8080
    depends_on:
      - mysql_db
    env_file: .env
    networks:
      - node-network

  mysql_db:
    image: mysql
    container_name: mysql_db
    ports:
      - 3306:3306
    environment:
      - MYSQL_ROOT_PASSWORD=1234
      - MYSQL_DATABASE=db_food
    env_file:
      - .env
    volumes:
      - mysql_db:/var/lib/mysql
    networks:
      - node-network

volumes:
  mysql_db:
    driver: local
networks:
  node-network:
```

```
DATABASE_URL="mysql://root:1234@mysql_db:3306/db_food?schema=public"
```