

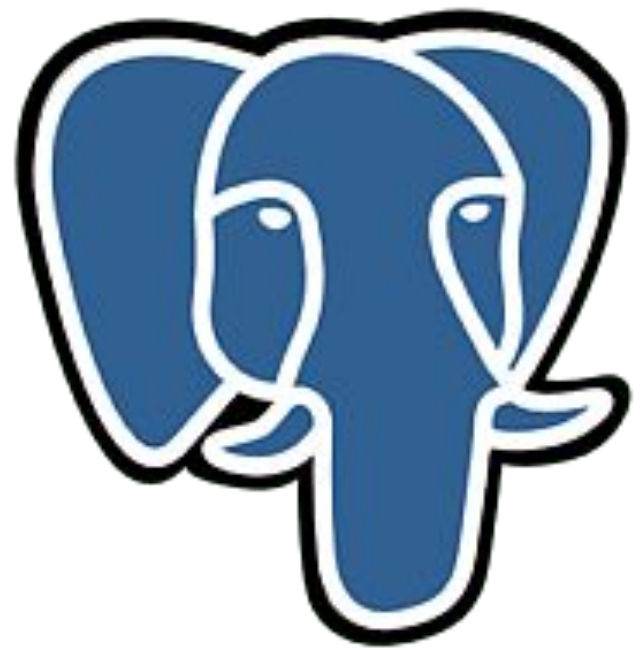


Capacitación Backend

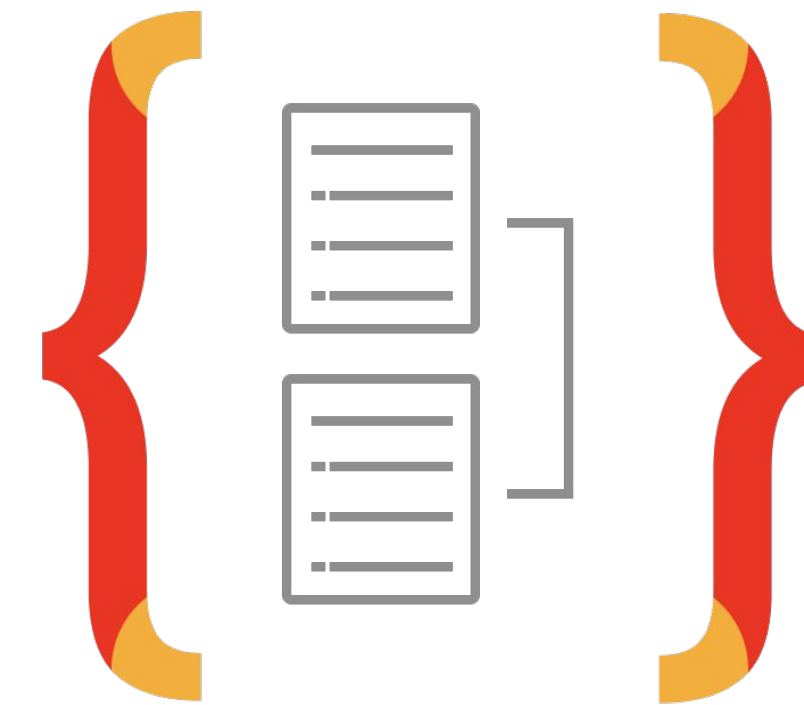
Bootcamp Desarrollo de Software

AGETIC - Octubre 2024

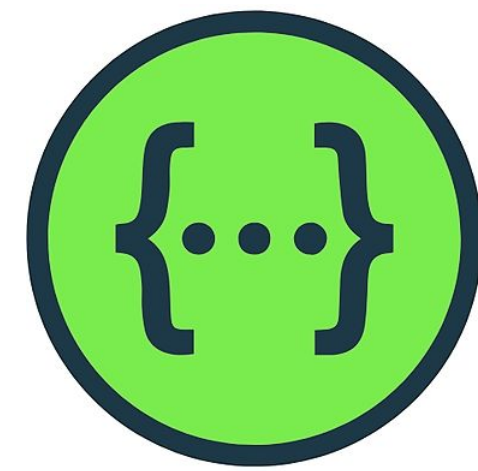
Base Backend - Tecnologías



PostgreSQL



TYPEORM



SwaggerTM



Passport



casbin[★]

Nest Js

Framework para construir aplicaciones del lado del servidor con Node.js. Construido sobre TypeScript y utiliza conceptos de POO y programación funcional.

Basado en módulos, controladores y servicios, lo que permite una organización limpia y escalable del código.

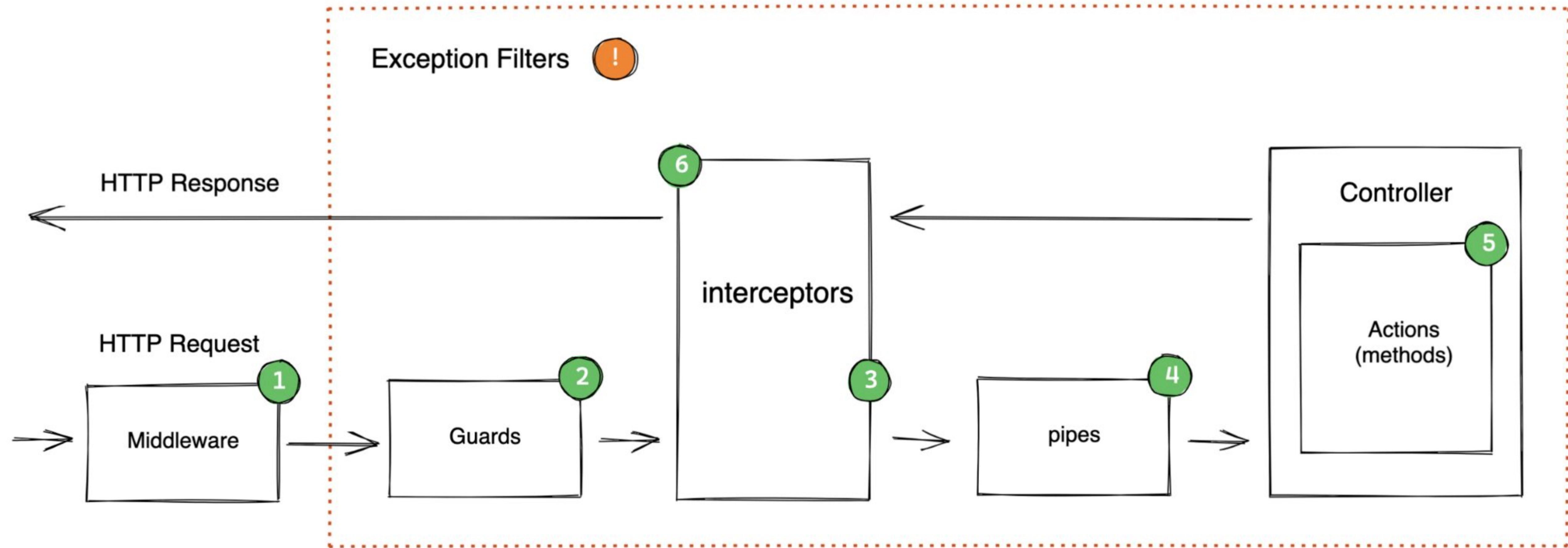
```
npm install -g @nestjs/cli
```

Nest JS - Ventajas

- Estructura Modular
- Soporte para Typescript
- Enfoque en la Escalabilidad
- Interoperabilidad
- Ecosistema Rico
- Soporte para Microservicios
- Documentación Exhaustiva

Nest JS - Requests

Ciclo de vida



Módulos

Agrupación lógica de funcionalidades relacionadas

```
nest g module <module_name>
```


Controladores

Manejo de solicitudes entrantes y respuestas al cliente.
Se definen rutas y métodos HTTP (GET, POST, etc.)

```
nest g controller <controller_name>
```

Servicios

Contienen la lógica de negocio y son inyectables en los controladores.

nest g service <service_name>

Pipes

Permiten validar y transformar datos de entrada en las solicitudes.

Ventajas:

- Mejora robustez de la aplicación al validar datos
- Transformación de datos previo tratamiento en servicios

Paquetes:

- class-validator
- class-transformer

Interceptors

Permiten manipular las solicitudes y respuestas, así como realizar acciones antes y después de la ejecución de un controlador.

Ejemplos de uso:

- Registro de logs o manejo de excepciones.

Ventajas:

- Facilitan el manejo de aspectos transversales de la aplicación.

Middlewares

Funciones que se ejecutan durante el ciclo de vida de una solicitud, antes de llegar al controlador.

Ejemplos de uso:

- Autenticación y autorización.

Ventajas:

- Permiten agregar lógica común a múltiples rutas

Base de datos

Persistencia

@nestjs/typeorm typeorm pg

Typeorm

ORM (Object-Relational Mapping) para TypeScript y JavaScript que permite interactuar con bases de datos de manera más sencilla y eficiente.

Características:

- Soporte para múltiples bases de datos (PostgreSQL, MySQL, SQLite, etc.).
- Soporte para migraciones y seeding (datos semilla)
- Integración fácil

Repositories

Objetos que encapsulan la lógica de acceso a datos. Proporcionan métodos para interactuar con la base de datos, como crear, leer, actualizar y eliminar (CRUD) entities.

DataSources:

Representa una conexión con la base de datos y contiene la configuración necesaria para interactuar con ella.

Entities

Clases que representan tablas en la base de datos. Cada instancia de una entidad corresponde a una fila en la tabla.

Decoradores comunes:

- `@Entity`
- `@Column`
- `@PrimaryGeneratedColumn`

Nest Js - Otros conceptos

- **Módulo de configuración y variables de entorno**

@nestjs/config

- **Documentación de los endpoints**

@nestjs/swagger

Proyecto Base Backend

Scaffold de proyecto para desarrollo de servicios

<https://gitlab.softwarelibre.gob.bo/capacitacion/nestjs-base-backend>

Base Backend - Funcionalidades

- **Usuarios:** Gestión de usuarios (CRUD)
- **Roles:** Gestión de roles (CRUD) y manejo de roles de usuarios
- **Módulos:** Gestión de módulos (CRUD)
- **Paramétricas:** Gestión de paramétricas (CRUD)
- **Autenticación:** JWT y Ciudadanía (proveedor de identidad)
- **Manejo de logs para “debug”**
- **Validaciones mediante DTO's**

Integración servicios externos
















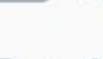









- Ciudadanía Digital
- IOP
- Mensajería (Alertin)



Árbol de directorios

Basado en
Clean Architecture
y
Screaming Architecture

▼ AGETIC-NESTJS-BASE-BACKEND

- >  .husky
- >  backups
- >  database
- >  docs
- >  public
- >  scripts
- ▼  **src**
 - ▼  application
 - >  parametro
 -  application.module.ts
 - >  common
 - ▼  core
 - >  authentication
 - >  authorization
 - >  config
 - >  external-services
 - >  logger
 - >  usuario
 -  core.module.ts
 - >  templates
 - >  types
 -  app.controller.spec.ts
 -  app.controller.ts
 -  app.module.ts
 -  **main.ts**

Base Backend - Tecnologías probadas

 RabbitMQ



redis

 BULLMQ



socket.io



**Firebase Push
Notifications**

Documentación extra

- **Documentación proyecto base:**
<https://gitlab.softwarelibre.gob.bo/capacitacion/nestjs-base-backend/-/blob/main/README.md>
- **Documentación instalación:**
<https://gitlab.softwarelibre.gob.bo/capacitacion/nestjs-base-backend/-/blob/main/INSTALL.md>
- **NestJs:** <https://docs.nestjs.com/>
- **Typeorm:** <https://typeorm.io/>
- **Documentación de API:** <https://docs.nestjs.com/openapi/introduction>

- **Validaciones:** <https://docs.nestjs.com/techniques/validation>
- **Conventional commits:**
<https://www.conventionalcommits.org/en/v1.0.0/#summary>
- **Subida de archivos:** <https://docs.nestjs.com/techniques/file-upload>
- **Websockets:** <https://docs.nestjs.com/websockets/gateways>
- **Microservicios:** <https://docs.nestjs.com/microservices/basics>
- **Rabbitmq:** <https://docs.nestjs.com/microservices/rabbitmq>
- **BullMq:** <https://docs.nestjs.com/techniques/queues>



Gracias por tu atención

AGETIC - Octubre 2024