**Documentación del Proyecto WasteManager**

**Descripción del Proyecto**

El proyecto final cuenta con 5 microservicios distribuidos en 4 aplicaciones. Se utilizó Maven y Java 17.

1. **waste-manager**

Cuenta con dos servicios:

* WasteManagerService: Servicio que gestiona la entidad WasteManagerEntity junto con sus entidades relacionadas. Tiene su propio CRUD y exporta un API Rest con las funciones create, update y findById. Para el manejo de la entidad WasteManagerAddressEntity, hace uso del CRUD del servicio WasteManagerAddressService.
* WasteManagerAddressService: Servicio que gestiona los datos correspondientes a la entidad WasteManagerAddressEntity mediante su propio CRUD. Se le agregó además los métodos REST para en caso que se quiera hacer una consulta puntual a estos datos, o se requiera una actualización.

Se utilizó la base de datos embebida H2 en memoria, y se le hicieron las configuraciones pertinentes, poniéndole como nombre “waste”. Ambos servicios trabajan en esta misma base de datos.

Fueron creados 6 paquetes siguiendo el patrón MVC:

* Paquete controller: Contiene las clases que manejan las solicitudes del cliente, en este caso solicitudes Rest.
* Paquete dto: Contiene la representación de los datos de manera estructurada y simple para su transferencia entre el cliente y el servidor.
* Paquete entity: Contiene las entidades que luego se persisten en la base de datos.
* Paquete model: Contiene la estructura de los datos de la aplicación
* Paquete repository: Contiene las interfaces para interactuar con la capa de persistencia de datos.
* Paquete service: Contiene la lógica de la aplicación y coordina las operaciones necesarias para llevar a cabo las solicitudes del cliente.

Decidí utilizar la librería *Lombok* para reducir la cantidad de código poco relevante, la dependencia JPA para persistir a la base de datos y la dependencia Bootstrap para poder cargar la configuración de la aplicación desde un servidor de configuración externo antes de que la aplicación arranque.

Se configuro con el 0 porque al utilizar Eureka y Spring Gateway se facilita la integración, ya que estos permiten que los servicios se registren y se enruten dinámicamente sin tener que preocuparse de puertos específicos.

El servicio fue desplegado bajo el perfil dev.

1. **waste-config-server**

Servidor que se encarga de externalizar y gestionar la configuración de las las aplicaciones de forma centralizada.

Decidí cargar las configuraciones desde GitHub, ya que permite mantener las configuraciones en un solo lugar, facilita la gestión y el seguimiento de los cambios estos archivos, y es uno de los más utilizados en la comunidad de desarrollo de software

En este caso utilice mi repositorio personal [*https://github.com/Pachuco5077/service-configuration*](https://github.com/Pachuco5077/service-configuration)*,* donde almacené las configuraciones de la aplicación waste-manager con dos perfiles (dev y prod).

1. **waste-naming-service**

Servidor que se encarga de implementar el descubrimiento y registro de los servicios.

Se utilizo la herramienta Eureka ya que esta se encarga de localizar y registrar servicios, y ayuda con el balanceo de cargas y la tolerancia a fallos.

1. **waste-gateway-service**

Servidor que se encarga del enrutamiento dinámico.

Se utilizo spring cloud Gateway, ya que permite tener un punto de acceso centralizado a nuestros microservicios.

Se realizo la configuración de una ruta, con un Path que permite acceder a los servicios de waste-manager.

Herramientas y Tecnologías Utilizadas

* Java: Utilicé Java 17.
* Spring Boot: Framework de desarrollo.
* Spring Cloud Config: Herramienta para la gestión centralizada de la configuración de aplicaciones distribuidas.
* GitHub: Plataforma de desarrollo colaborativo para alojar proyectos utilizando el control de versiones Git.
* H2 Database: Base de datos en memoria utilizada para el almacenamiento de datos en entornos de desarrollo.
* Maven: Herramienta de gestión de proyectos y construcción de software para Java.

Configuración y Ejecución

Clonar el Repositorio

Configurar Spring Cloud Config Server

Configurar un repositorio en GitHub para almacenar los archivos de configuración de la aplicación.

Configurar WasteManager para Usar Spring Cloud Config

Especificar el perfil bajo el que se desea desplegar en el archivo de configuración “bootstrap.properties”.

Ejecutar la Aplicación con Maven en Windows

1. Abre la consola de comandos de Windows.
2. Navega hasta el directorio raíz del proyecto WasteManager.
3. Ejecuta el siguiente comando para compilar y empaquetar la aplicación:

*mvn clean install*

1. Después de que Maven termine de construir el proyecto, navega al directorio target dentro del directorio del proyecto:
2. Corra el siguiente comando Java desde la carpeta del proyecto

*java -war target/entumovil-service-backend-account-0.0.1-SNAPSHOT.war*