Instrucciones para la ejecución.

Es necesario poseer JDK 17 o una version mayor que esta.

Instrucciones:

* Contar con una herramienta para probar APIs (Postman, ...)
* Ejecutar los microservicios en este orden:
    - microservice-config.
    - microservice-eureka.
    - microservice-wastemanager.
    - microservice-wastemanager-address.
    - microservice-gateway.
* Para ejecutar los microservicios seguir la siguiente ruta: microservice-name/src/main/java/MicroserviceNameAplication. Una vez en esta ejecutar esa clase.

Urls de los endpoints:

* localhost:8091/api/manager/...
    - create:(POST)

```
{
    "name": "El manager",
    "nif": "566",
    "version": 2,
    "wasteManagerAddressId": 1,
    "wasteAuthorizationList": [
        {
            "authorizationNumber": "1"
        },
        {
            "authorizationNumber": "3"
        }
    ],
    "isEnabled": true,
    "createdDate": "2021-03-04",
    "lastModifiedDate": "2021-03-04"
}
```

    - update:(PUT)

```
{
    "id": 1,
    "name": "El manager",
    "nif": "566",
    "version": 2,
    "wasteManagerAddressId": 1,
    "wasteAuthorizationList": [
        {
```

```
                "id": 1,
                "authorizationNumber": "1"
            },
            {
                "id": 2,
                "authorizationNumber": "3"
            }
        ],
        "isEnabled": true,
        "createdDate": "2021-03-04",
        "lastModifiedDate": "2021-03-04"
    }
```

- search/{id}.(GET)
- all.(GET)
- all/auth. (GET)

* localhost:8081/api/address/...
    - create:(POST)

```
    {
        "direction": "España",
        "version": 1,
        "isEnabled": true,
        "createdDate": "2020-03-04",
        "lastModifiedDate": "2021-03-04"
    }
```

    - update:(PUT)

```
    {
        "id": 1,
        "direction": "España",
        "version": 1,
        "isEnabled": true,
        "createdDate": "2020-03-04",
        "lastModifiedDate": "2021-03-04"
    }
```

    - search/{id}.(GET)
    - all.(GET)

* localhost:8080/api/"Nombre del Api a consultar (manager, address)"

Detalles de la implementación:

Microservices WasteManager, WasteManagerAddress.

Como bien se indica en la descripción de la prueba fue necesario crear dos microservicios: microservice-wastemanager y microservice-wastemanager-address. El primero contiene entities, services, repositories, controllers, dtos y client necesarios para implementar toda la funcionalidad de este microservicio.

En la carpeta entities se encuentran dos entidades: WasteManager y WasteManagerAuthorization. Se valoró la idea de crear un microservicio solo para la entidad WasteManagerAuthorization pero esta no fue la opción utilizada pues se entiende que un WasteManagerAuthorization tiene una relación Muchos-Muchos con el WasteManager y tenerlos en el mismo microservicio representa una ventaja, pues se podría establecer esta relación directamente en la entidad. Además no se especifica que sea necesario crear un microservicio distinto para esta, y se entiende que un WasteManagerAuthorization solo se utiliza en un WasteManager, por tanto no se cree necesario tener dos microservicios distintos para manejarlos
(Es importante destacar que cada microservicio que posee una entidad tiene su propia base de datos).

Se decidió que el campo wasteManagerAddressEntity no sea del tipo WasteManagerAddress, en su lugar sea del tipo Long, representando a los id de la tabla WasteManagerAddress. Se pensó en tener una dependencia entre microservicios, pero, al tener una referencia directa, se estará acoplando una entidad a la entidad del otro microservicio, lo cual no es la mejor práctica en arquitecturas de microservicios. Esto puede generar problemas de mantenibilidad y escalabilidad. Es por esto, que es necesario que exista un WasteManagerAddress con el id que es otorgado en el campo wasteManagerAddressId antes de poder crear un WasteManager.

En los repositorios de este microservicio se utiliza el CrudRepository para mayor facilidad. A los servicios se le inyectan los repositorios y un client, y a los controladores se le inyectan los servicios.

En el directorio client tenemos un cliente de WasteManagerAddress el cual puede ordenar a ejecutar los metodos implementados en el controlador de WasteManagerAddress, en este caso solo es utilizado para controlar que el id del campo wasteManagerAddressId existe. Para esto se utilizó la dependencia OpenFeign la cual permite anotar las clases con @FeignClient.

También se tiene una carpeta dto, la cual implementa los dto de las entidades. Estos poseen todos los campos excepto el campo id. Para este proyecto no fue necesario usar un dto de entrada y uno de salida, ya que ambos tendrían los mismos campos.

Microservices Config, Eureka, Gateway.

El microservico-eureka es el encargado de conocer los microservicios que se encuentran levantados para que el microservicio-gateway conozca hacia donde dirigir las peticiones que a este se le hagan.

Y por último el microservicio-Config es el encargado de brindarle a cada uno de los microservicios la configuración necesitada por estos.

Notas:

Si el ordenador en la que se probará el proyecto, se encuentra utilizando alguno de los puertos en los cuales los microservicios se levantan, cambiar este puerto en el Config, accediendo a la ruta: src/main/resources/configurations/msvc-nombre del microservicio.