Actividad 02. Desarrollo de un sistema software basado en microservicios

En esta actividad el estudiante debe diseñar, desarrollar y probar un sistema basado en microservicios.

- El sistema basado en microservicios propuesto estará basado en el proyecto de Ingeniería del Software o bien en el proyecto de Ingeniería de Requisitos. Así, la primera parte de la actividad debe llevarnos a realizar un análisis del proyecto definido e identificar los microservicios que deben implementarse. Para ello, utilizaremos la técnica de identificar en el diagrama de casos de uso las distintas capacidades del negocio (business capabilities), identificando, al menos, 3 microservicios a implementar.
- Para cada uno de los microservicios, será necesario identificar los datos con los que trabajará y, en consecuencia, definir el esquema de base de datos concreto de cada uno de ellos. Ello supone que un microservicio podrá ser responsable de una o varias tablas.
- Será necesario definir un servicio central encargado de ofrecer toda la funcionalidad del sistema, invocando para ello a los distintos microservicios base que se han definido.
- Para el registro de los microservicios desplegados en el sistema se utilizará el servidor de registro denominado Eureka.
- Para la configuración del sistema utilizaremos un servidor de configuraciones Configuration Server de Spring, ello facilita la configuración de todos los servicios.

El esquema de funcionamiento del sistema puede ser similar al siguiente, aunque las relaciones entre los microservicios base definidos (microservicio1, microservicio2 y microservicio3) puede ser distinta en función de cada uno de los proyectos propuestos.

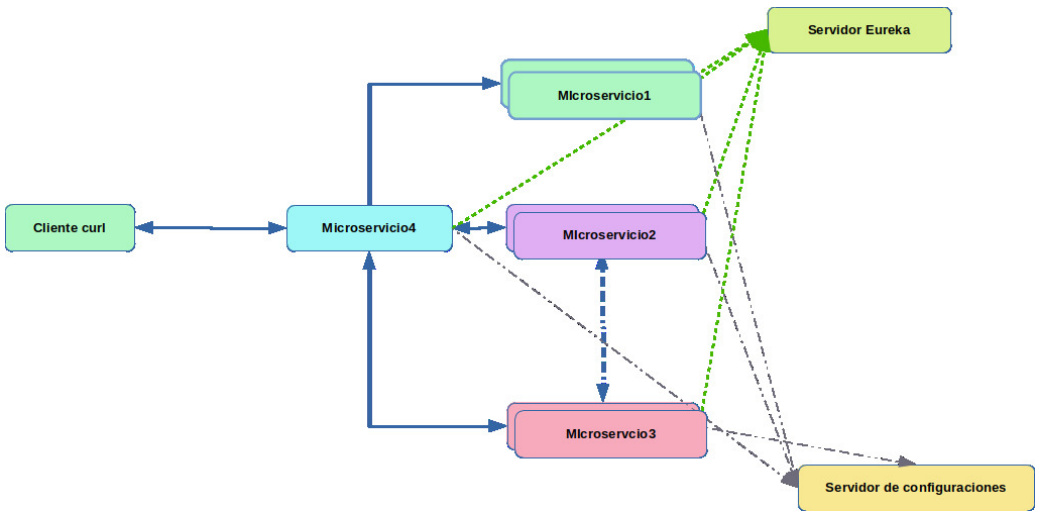

Figure 1: Esquema general de la propuesta de estructura

La estructura del código puede ser la siguiente:

```
microservicio1_eureka_server/
microservicio2_configuration_server/
microservicio3/
microservicio4/
microservicio5/
microservicio6/

script_compilacion_empaquetado.sh  #Script para ejecutar `mvn clean package` en todos los proyectos
script_ejecución_sistema.sh  #Script para ejecutar cada uno de los `mvn spring-boot:run`
scripts_ejecución_pruebas.sh # Script que utilizando `curl` permita la ejecución de las pruebas sobre los distintos microservicios
```

Estructura de la documentación

- Introducción del sistema a desarrollar.
- Descripción de los casos de uso y su división en microservicios. Descripción de la base de datos que tendrá cada uno de los microservicios.
- Esquema general del sistema propuesto.
- Definición de la api de cada uno de los microservicios. La definición de la api puede generarse utilizando openapi.
- Scripts de compilación, empaquetado y ejecución.
- Problemas encontrados y conclusiones

```
# Entregar un archivo comprimido con .zip que incluya lo siguiente. Antes de comprimir, elimina la carpeta target/ de cada microservicio:

actividad1/
    doc/
    codigo/
```

Checklist

- ☐ Tengo identificados, al menos 3 microservicios base (microservicio1, microservicio2 y microservicio3 del esquema).
- ☐ Tengo definido el servicio central, encargado de invocar a los servicios base definidos.

- ☐ Todos los servicios se registran y utilizan Eureka.
- ☐ Todos los servicios utilizan de forma adecuada el servidor de configuraciones de Spring (Spring Configuration Server)
- ☐ Cada microservicio base se ejecutará de forma redundante, es decir, al menos dos instancias para cada uno de ellos.
- ☐ Para facilitar la ejecución de los microservicios puede ser interesante crear un script que levante todos o bien que pare todos.
- ☐ Tengo definido un script de pruebas para ejecutar pruebas sobre los distintos microservicios
- ☐ He documentado la entrega siguiendo las pautas indicadas en este documento (concretamente en su sección `Estructura de la documentación`).

Tecnologías

- curl
- Maven
- JDK 17
- SpringBoot
- Spring EurekaServer
- Spring LoadBalancer
- Spring Configuration Server

Nota. Se recomienda usar Springboot, pero el alumno puede elegir otro stack tecnológico respetando la funcionalidad y características de cada uno de los microservicios propuestos.

Fecha de entrega: Definida en el aula virtual de la asignatura.

Apéndice I. Añadir documentación openapi a un proyecto SpringBoot - versiones nuevas

La configuarción de `springdoc` ha cambiado. Toda la información está actualizada en https://springdoc.org/

Para generar automáticamente la documentación de las API REST de un proyecto SpringBoot será necesario incluir la siguiente dependencia en el archivo `pom.xml`.

```
<dependency>
   <groupId>org.springdoc</groupId>
   <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
   <version>2.6.0</version>
</dependency>
```

Una vez añadida esta dependencia, podremos acceder al url del microservicio a la `/swagger-ui/index.html`. Por ejemplo, `http://localhost:8020/swagger-ui/index.html` si el microservicio estuviese desplegado en `http://localhost:8020`. La definición openapi correspondiente se encuentra en `/v3/api-docs`, por ejemplo `http://localhost:8020/v3/api-docs`.

(Deprecated) Apéndice I. Añadir documentación openapi a un proyecto SpringBoot

Para generar automáticamente la documentación de las API REST de un proyecto SpringBoot será necesario incluir la siguiente dependencia en el archivo `pom.xml`.

```
<!-- https://mvnrepository.com/artifact/org.springdoc/springdoc-openapi-ui -->
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-ui</artifactId>
    <version>1.7.0</version>
</dependency>
```

Una vez añadida esta dependencia, podremos acceder al url del microservicio a la `/swagger-ui/index.html`. Por ejemplo, `http://localhost:8020/swagger-ui/index.html` si el microservicio estuviese desplegado en `http://localhost:8020`. La definición openapi correspondiente se encuentra en `/v3/api-docs`, por ejemplo `http://localhost:8020/v3/api-docs`.