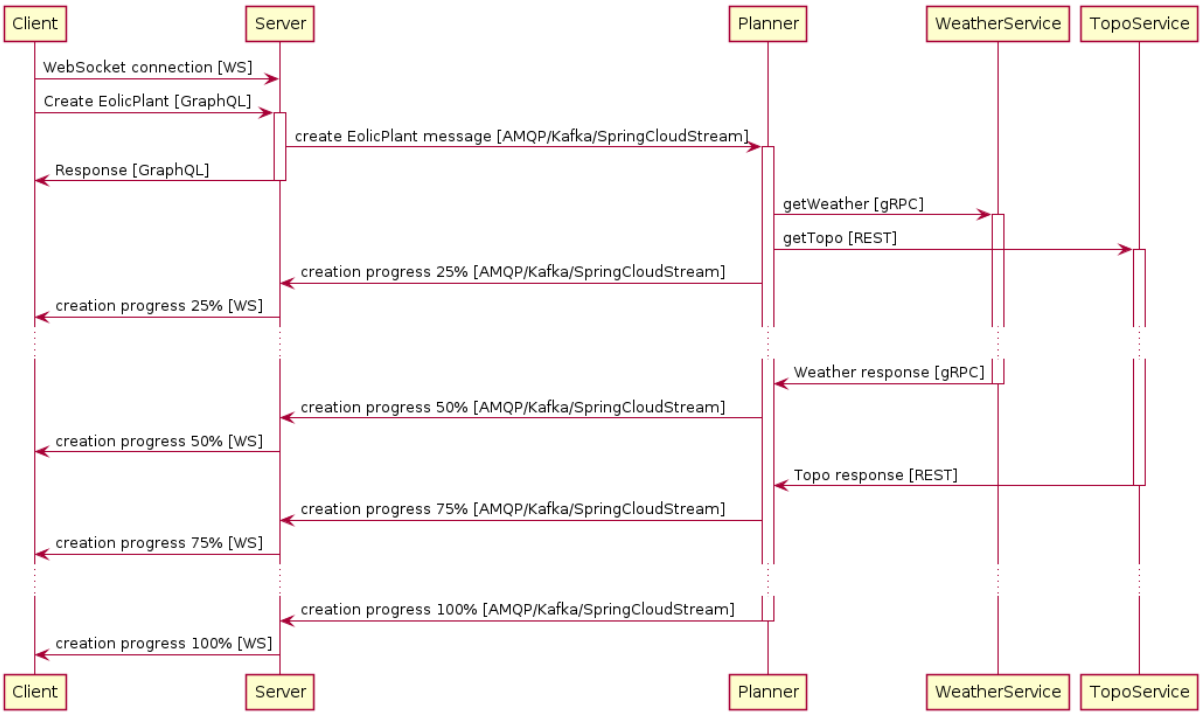
Práctica 1. Dockerizar una aplicación

Enunciado

Dadas las ventajas que proporciona Docker para el empaquetado, distribución y ejecución de aplicaciones, vamos a ampliar y “dockerizar” una aplicación implementada como parte de una práctica previa en el máster.

En concreto, se va a ampliar y empaquetar en diferentes contenedores Docker la aplicación de la Práctica 4 de la asignatura “Tecnologías de Servicios de Internet” del Módulo II - Servicios de Internet.

La aplicación está formada por 4 servicios que interactúan entre sí usando diferentes tipos de protocolos, dos bases de datos (MySQL y MondoDB) y un broker de mensajería. Los servicios son Server, Planner, WeatherService y TopoService. La comunicación entre ellos se representa en el siguiente diagrama:



Se debe empaquetar la aplicación siguiendo estos criterios:

- Cada servicio deberá empaquetarse en su propio contenedor.
- Para las bases de datos Mongo y MySQL y para el broker RabbitMQ se usarán los contenedores publicados en DockerHub.
- Para los servicios *Server*, *Planner*, *WeatherService* y *TopoService* habrá que crear contenedores propios que deberán publicarse en una cuenta de **DockerHub** del alumno.
- Todos los contenedores de la aplicación estarán coordinados usando **Docker Compose** que usará las imágenes publicadas en DockerHub (las oficiales para las bases de datos y el broker y las personalizadas para el resto de servicios).
- Las bases de datos **MySQL y Mongodb** deberán utilizar **volúmenes** para que la información se almacene en el host. La cola **RabbitMQ** también deberá utilizar un **volumen** para persistir la información.
- Las bases de datos **MySQL**, **Mongodb** y la cola **RabbitMQ** deben estar **securizadas** con **usuario y contraseña**.
- Creación de contenedores para los servicios:
    - **Server:** La imagen se creará utilizando el plugin de *Docker* para Quarkus. La espera de los servicios debe hacerse utilizando la etiqueta **restart**.
    - **WeatherService:** La imagen se creará utilizando un *Multistage Dockerfile* con cacheo de dependencias Maven.
    - **Planner:** La imagen se creará utilizando *JIB* para *Spring.* La espera de los servicios debe hacerse utilizando la herramienta **wait-for-it.sh** o similares.
    - **TopoService:** La imagen se creará utilizando Spring *Buildpacks* y deberá ser un compilado *GraalVM Native.* La espera de los servicios debe hacerse utilizando **HEALTHCHECK** junto a la etiqueta **depens_on**.
    - Se deberá crear un **script bash** para generar las imágenes y publicarlas en una cuenta de DockerHub. En Windows se deberá usar **WSL2** con Docker para ejecutar este script o en su defecto **Git BASH**.
    - Cuando un servicio tenga que conectarse a otro servicio, la configuración de conexión (nombre del host, puertos y credenciales) debería especificarse como **variable de entorno** para el servicio.
- Desarrollo en VSCode:
    - Se creará un fichero **.devcontainer** por cada servicio. De forma que pueda desarrollarse sin necesidad de tener las herramientas de desarrollo en la máquina (Maven).
    - Para acceder a los servicios MySQL, Mongo y RabbitMQ, se creará un *docker-compose-dev.yml* en el que sólo estén configurados estos servicios. Se deberá configurar el **.devcontainer** en modo **network=host**.
- En la raíz del repositorio deberá incluirse un fichero README.md que describa los diferentes modos de desarrollo y ejecución de la aplicación.

- Junto con el código se deberá crear un vídeo capturando la pantalla del ordenador en el que se muestran los siguientes aspectos:
    - Breve repaso por las configuraciones **Docker** de los diferentes servicios.
    - Arranque de los servicios desde **Docker Compose**.
    - Correcto funcionamiento de la web desde el navegador.
    - Entradas de log en el **Docker Compose**.
    - Visualización del desarrollo con **VSCode**.

Formato de entrega

La práctica se entregará teniendo en cuenta los siguientes aspectos:

- La práctica se entregará como un fichero .zip con el código de los diferentes servicios y los ficheros auxiliares. El nombre del fichero .zip será el correo URJC del alumno (sin @alumnos.urjc.es).

Las prácticas se podrán realizar de forma individual o por parejas. En caso de que la práctica se haga por parejas:

- Sólo será entregada por uno de los alumnos
- El nombre del fichero .zip contendrá el correo de ambos alumnos separado por guión. Por ejemplo `p.perezf2019-z.gonzalez2019.zip`