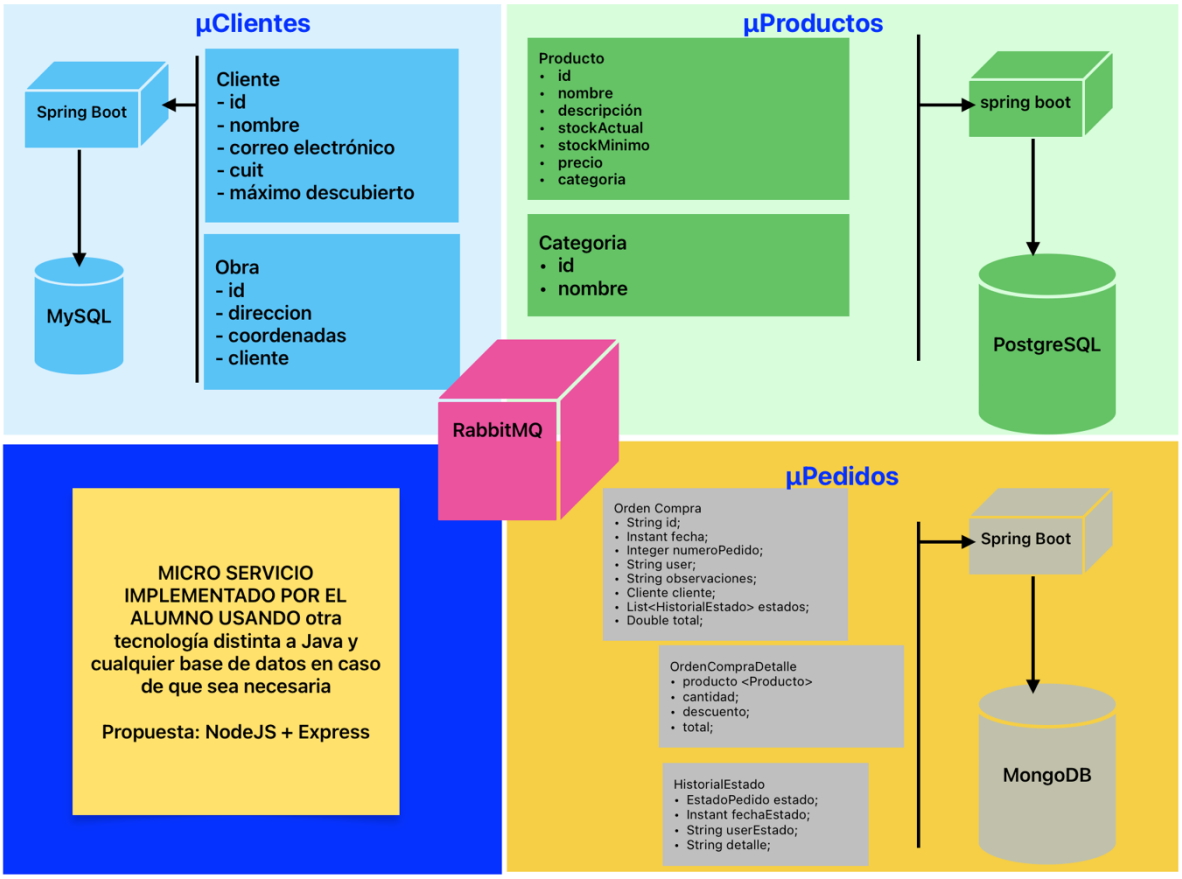
TP FINAL DAN

Modelo propuesto



Objetivos

El objetivo del presente trabajo práctico es incorporar de manera práctica los conocimientos brindados en la materia completando las actividades prácticas propuestas y agregando la implementación de un conjunto de funcionalidades opcionales a elección de cada grupo.

Al finalizar el trabajo práctico se pretende que el alumno haya comprendido:

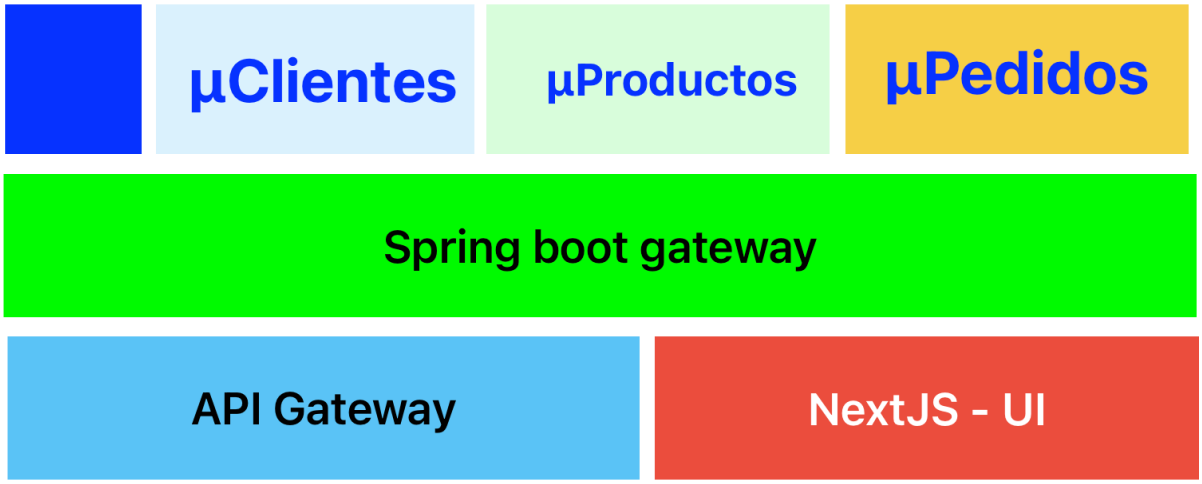
- Funcionamiento básico de una aplicación spring boot
- Funcionamiento básico de una aplicación Express
- Funcionamiento básico de una aplicación ReactJS / NextJS
- Ejecución de aplicaciones en entornos virtualizados
- Comunicación entre aplicaciones distribuidas de manera sincrónica y asincrónica.
- Como modularizar aplicaciones en servicios de menor tamaño (microservicios) independientes del lenguaje de programación, plataforma de ejecución y base de datos subyacentes.
- Como aplicar técnicas para garantizar la tolerancia a fallas, y la auto recuperación y autoconfiguración de aplicaciones distribuidas virtualizadas (API Gateway – Service Discovery – Load Balancer – Reverse Proxy – Circuit Breaker – Retry – etc)
- Como configurar loggers y herramientas para lograr observabilidad

- Como configurar herramientas de telemetría para obtener información del rendimiento de los componentes de las aplicaciones.
- Configuraciones básicas de seguridad.

Tareas obligatorias

- Deberá entregar una aplicación que representa un marketplace B2B. En este Marketplace se registran clientes, que tienen una o más obras de construcción y usaran el sistema para solicitar materiales de construcción a través de una interface web llevar a cabo las siguientes funcionalidades:
  - Debo poder ingresar a la aplicación con mi usuario y mi contraseña, el cual puede tener 2 roles:
    - Vendedor
    - Cliente
  - Como vendedor puedo:
    - Dar de alta clientes y obras y poder consultarlos por distintos criterios
    - Dar de alta y actualizar producto y poder consultarlos por distintos criterios
    - Consultar los pedidos creados.
    - Actualizar el estado de los pedidos creados.
  - Como usuario cliente puedo
    - Buscar productos.
    - Ver mis datos de perfil (datos de cliente y obras asociadas)
    - Crear un pedido y ver el estado del pedido.
    - Ver el historial de pedidos.
    - Ver el historial de estados por los que paso un pedido.
  - El sistema debe permitirme realizar un logout y luego entrar con otro usuario

Arquitectura propuesta



Requisitos técnicos obligatorios

- La aplicación debe tener una UI desarrollada en React y NextJS
- La aplicación debe contener los 3 módulos ya creados y un 4to módulo que resuelva un problema que el grupo determine y que se integre a la aplicación.
- La aplicación debe ejecutar en contenedores virtuales.

- La aplicación debe usar las bases de datos propuestas en los 3 módulos trabajados hasta aquí, y el 4to modulo que agregue puede usar una base existente, una nueva, o ninguna. Las bases de datos no es necesario que ejecuten en Docker, o que ejecuten en el mismo proyecto o en la misma red, si por necesidad necesita distribuirlas o sacarlas de un contenedor puede hacerlo.
- La interface de usuario debe ejecutar en un contenedor y debe ser accedida a través de un Gateway o ProxyReverso.
- De los cuatro módulos de microservicios que utiliza **al menos 1 debe tener más de una instancia** y debe implementar alguna forma de balanceo de carga.
- Deberá implementar en alguna de las llamadas que ocurren entre microservicios, el patrón Circuit Breaker **O** el patrón retry (puede implementar ambos, pero al menos uno de los dos) para solventar la no disponibilidad de un servicio.
- Al menos un 1 unit test de cada uno de los siguientes tipo
    - Unit test que compruebe un endpoint rest usando mockMvc
    - Unit test que compruebe un servicio mockeando dependencia
    - unit test que compruebe un repositorio usando una base de datos embebida o usando https://testcontainers.com/

Tareas opcionales

De las siguientes funcionalidades **deberá** implementar **al menos 2**

- Configurar alguna herramienta que permita obtener telemetría de las aplicaciones (por ejemplo, Prometheus y Graphana puede ser otra)
- Configurar alguna herramienta que permita obtener observabilidad de las aplicaciones (por ejemplo, Graylog Opensearch o cualquier otra)
- Utilizar alguna implementación de kubernetes para desplegar las aplicaciones.
- Utilizar imágenes nativas (GraalVM)
- Utilizar serverless.
- Utilizar keycloak para seguridad.
- Utilizar un servicio async (Rabbit / Kafka)
- Utilizar Eureka para service Discovery
- Utilizar keycloak para autenticacion
- Alguna otra funcionalidad cloud no cubierta en la materia es válida previa consulta con la cátedra.

Información importante:

- Modalidad: grupos de 1 a 3 personas (MAXIMO 3)
- Fecha de entrega: AGOSTO 2024