Arquitectura de Software II 2023

Trabajo Final Integrador

Objetivo

Implementar un sistema de microservicios, mediante el cual una cadena de hoteles disponibiliza su oferta de forma local y se integra con un proveedor central para validar los pedidos al momento de la reserva.

Arquitectura

La aplicación debe contar con al menos 4 microservicios:

1. **Microservicio frontend**
2. **Microservicio de ficha de hotel**
3. **Microservicio de búsqueda de hotel**
4. **Microservicio de usuarios, reserva y disponibilidad**

Microservicio frontend

Debe tener 4 pantallas:

- Pantalla inicial con 3 campos de búsqueda: ciudad, fecha desde y fecha hasta.
- Pantalla de resultado de la búsqueda con el listado de hoteles con disponibilidad para esa ciudad: nombre, descripción y thumbnail.
- Pantalla de detalle del hotel con la información completa: nombre, descripción, fotos, amenities y botón de reserva.
- Congrats, con confirmación exitosa o rechazo de la reserva.

Microservicio de ficha de hotel

Microservicio RESTful en Go que almacena la información de los hoteles con formato documental en una base de datos no relacional (MongoDB) y disponibiliza esa información de forma RESTful. Soporta y valida la información de creación y modificación de hoteles así también como la obtención de la información por id de hotel. Notifica a una cola de mensajes (RabbitMQ) cuando se crea o se modifica un hotel.

Microservicio de búsqueda de hotel

Microservicio RESTful en Go que contiene un motor de búsqueda (Solr) que disponibiliza la información de los hoteles en base a los criterios de búsqueda que se le especifiquen. Escucha la cola de mensajes de creación y actualización del servicio de ficha de hoteles y sincroniza los cambios haciendo un GET por ID. Consulta concurrentemente la disponibilidad de los resultados para filtrar en la búsqueda mediante un atributo dinámico que indique si tiene disponibilidad o no, esto significa tener un atributo por ej “availability”, pero que no se persista, sino que devuelva los resultados de la consulta al servicio de disponibilidad de forma interna y concurrente para el listado de resultados.

Microservicio de usuarios, reserva y disponibilidad

Microservicio RESTtful en Go que contiene una base de datos relacional (MySQL) con la información de los usuarios/clientes del sitio y sus reservas. El microservicio tiene que tener la capacidad de retornar la disponibilidad de los hoteles con un caché distribuido (Memcached) con TTL de 10 segundos que evita su cálculo subsiguiente. De esa forma, se optimizan las consultas suponiendo una consistencia eventual de 10 segundos. Al momento de realizar las reservas se tiene que validar externamente con el sitio Amadeus (anexo 1), dado que la oferta puede estar distribuída y su validación debe estar concentrada por este proveedor externo de servicios de booking. Este microservicio debe proveer todos los endpoints necesarios para la operación del cliente y gestión de sus reservas. El mapping entre el ID interno del hotel y el ID de Amadeus lo conoce este microservicio. Los nuevos hoteles se generan de forma síncrona en el POST inicial mediante el servicio de ficha de hoteles.

Anexos

Anexo 1 - Operatoria con Amadeus

1. Registrarse en el sitio https://developers.amadeus.com/ con el email de la UCC para obtener acceso a la API.
2. Seguir los pasos para obtener el token de operación, como se indica en https://developers.amadeus.com/get-started/get-started-with-self-service-apis-335
3. Parametrizar el microservicio para la obtención automática del token de operación mediante la inyección de los secrets:

```
curl "https://test.api.amadeus.com/v1/security/oauth2/token" -H
"Content-Type: application/x-www-form-urlencoded" -d
"grant_type=client_credentials&client_id=[KEY]&client_secret=[SECRET]"
```

Ejemplos

```
curl
'https://test.api.amadeus.com/v1/reference-data/locations/hotels/by-city?c
ityCode=PAR' -H 'Authorization: Bearer [TOKEN]'
```

```
curl
'https://test.api.amadeus.com/v3/shopping/hotel-offers?hotelIds=YXPARKPR&c
heckInDate=2024-01-22' -H 'Authorization: Bearer [TOKEN]'
```

API de disponibilidad

La API de disponibilidad se encuentra en https://developers.amadeus.com/self-service/category/hotels/api-doc/hotel-search/api-reference

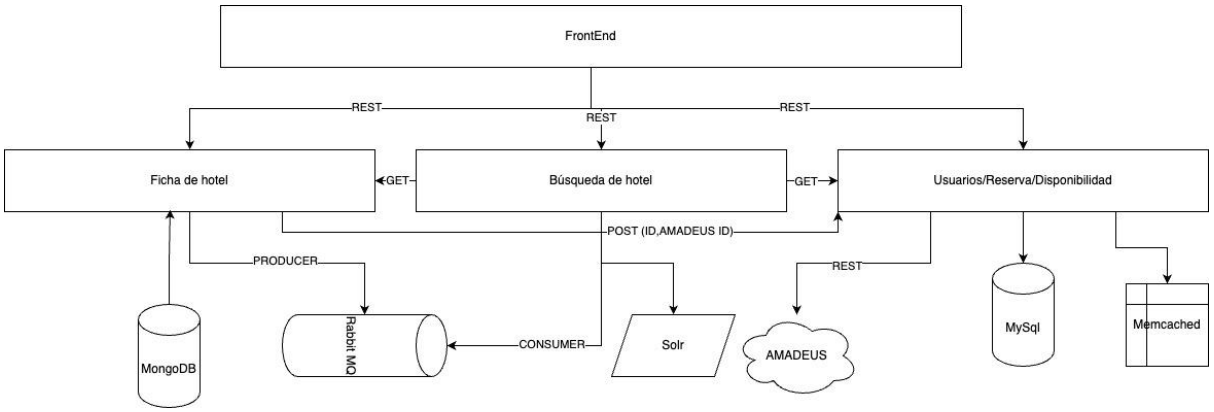
API de hoteles

La API de hoteles se encuentra en https://developers.amadeus.com/self-service/category/hotels

Anexo 2 - Puntos adicionales para el final

- Implementar Load Balancer en todos los microservicios de backend.
- Implementar el escalado automático para al menos 1 microservicio y su correspondiente test de carga o downtime para forzar el escalado.
- Implementar los test unitarios para todos los microservicios de backend.
- Implementar Docker Compose como orquestador para levantar la solución.
- Desarrollar un Frontend (de administración de infraestructura) que visualice el estado de los contenedores de los servicios y permita crear nuevos contenedores y destruirlos.
- Implementar una Caché Local en algún microservicio para incrementar la escalabilidad.

Anexo 3 - Arquitectura



Evaluación

Es condición de aprobación (nota 4) que todos los criterios descritos estén funcionando correctamente. Trabajo que no está completo, no se aprueba.