Università di Catania

BiobankSPREC Microservices

Progetto Sistemi Cloud e Laboratorio (LM-18)
Università degli Studi di Catania - A.A 2021/2022

Danilo Leocata - 1000022576
Docenti: Giuseppe Pappalardo, Andrea Fornaia

July 22, 2022

1 Introduzione

È stato preso in esame un progetto a microservizi (inserisci di cosa parla) realizzato precedentemente dal collega. L'intero progetto è disponibile al seguente link GitHub: (link progetto vecchio).

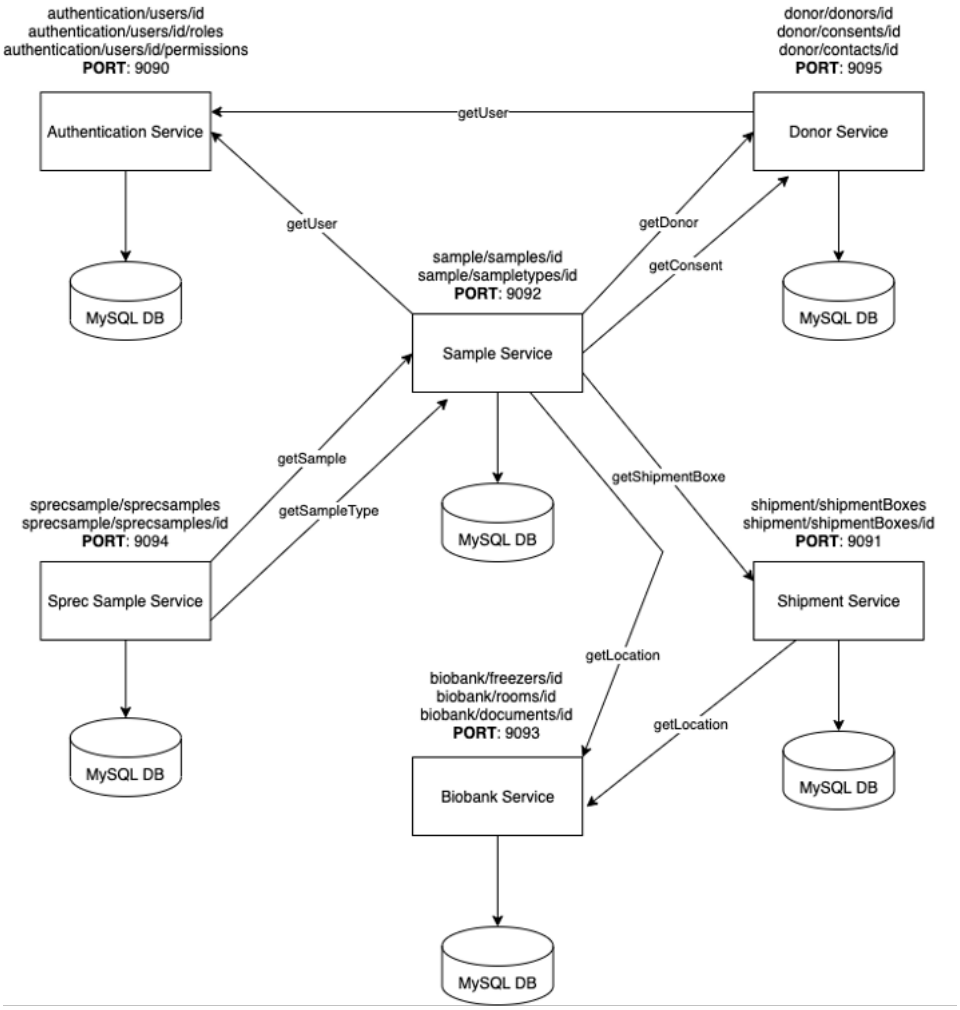

Figure 1: Architettura iniziale

È stato trovato più opportuno, iniziare ad effettuare un primo refactoring del progetto. In particolare è stata prevista la suddivisione dei microservizi, ed i relativi database, in repository separate in modo da scindere il deploy di database e relativo microservizio associato. Tutte le configurazioni (come quella di Keycloak, ad esempio) e le documentazioni generali sono state inserite all'interno della repository `main`. Per agevolare lo sviluppo ed i test sono stati implementati due script bash, nel dettaglio:

1. `setup-dev.sh`: crea una cartella `development`, al cui interno effettua i clone delle repository, prepara i file in modo da renderli eseguibili per il `docker-compose`, effettua la build del microservizi con `maven` e copia tutti i file necessari per la configurazione dentro la cartella `imports`
2. `update-repo.sh`: script che effettua pull del `main` in tutti i branch e ne effettua le build

In generale, sono stati effettuati i seguenti miglioramenti su ogni microservizio:

1. Il microservizio di authentication è stato sostituito in favore di Keycloak;
2. Sono stati estratti, dai rispettivi container docker, le tabelle ed i dati su file `.sql` (inizialmente il db veniva inizializzato a run-time durante l'avvio del microservizio);
3. I file `pom.xml` sono stati stato revisionati ed aggiornati;

4. Il formato delle API è stato cambiato da *camelCase* ad *hypenate*
5. I file `application.properties` sono stati stato convertiti in formato `.yml` e splittato in `dev`, per lo sviluppo, e `prod` per il deploy.
6. Sono state implementate le **Git Actions**
7. Orchestrazione dei microservizi con **Kubernetes**

2 Keycloak

Il primo step è stato quello di sostituire il microservizio di autenticazione con Keyacloak. Keycloak è un prodotto software open source che abilita il Single Sign-On (IdP) con Identity Management e Access Management per applicazioni e servizi moderni. Questo software è scritto in Java e supporta i protocolli di federazione delle identità per impostazione predefinita SAML v2 e OpenID Connect (OIDC) / OAuth2. Lo scopo dello strumento è quello di facilitare la protezione di applicazioni e servizi con poca o nessuna crittografia. Un IdP consente a un'applicazione (Service Provider) di delegare la propria autenticazione

Si riassumono brevemente alcuni dei concetti principali:

1. **Users**: entità che sono in grado di accedere al tuo sistema. Possono avere attributi associati a se stessi come e-mail, nome utente, indirizzo, numero di telefono e giorno di nascita. È possibile assegnare loro l'appartenenza a un gruppo e assegnare loro ruoli specifici.
2. **Roles**: identificano un tipo o una categoria di utente: sono tutti ruoli tipici che possono esistere in un'organizzazione. Le applicazioni spesso assegnano l'accesso e le autorizzazioni a ruoli specifici piuttosto che a singoli utenti, poiché la gestione degli utenti può essere difficile da gestire.
3. **User role mapping** Una mappatura dei ruoli utente definisce una mappatura tra un ruolo e un utente. Un utente può essere associato a zero o più ruoli. Queste informazioni sulla mappatura dei ruoli possono essere incapsulate in token e asserzioni in modo che le applicazioni possano decidere le autorizzazioni di accesso su varie risorse che gestiscono.
4. **Realms** un realm gestisce un insieme di utenti, credenziali, ruoli e gruppi. Un utente appartiene e accede a un realm che sono isolati l'uno dall'altro e possono solo gestire e autenticare gli utenti che controllano.
5. **Clients**: i client sono entità che possono richiedere a Keycloak di autenticare un utente. Nella maggior parte dei casi, i client sono applicazioni e servizi che desiderano utilizzare Keycloak per proteggersi e fornire una soluzione single sign-on. I client possono anche essere entità che desiderano semplicemente richiedere informazioni sull'identità o un token di accesso in modo da poter invocare in modo sicuro altri servizi sulla rete protetti da Keycloak.
6. **Client scope**: quando un client viene registrato, è necessario definire i mappatori di protocollo e i mapping dell'ambito del ruolo per quel client. Spesso è utile memorizzare un ambito client per semplificare la creazione di nuovi client condividendo alcune impostazioni comuni, è utile anche per richiedere che alcuni ruoli, ad esempio, siano condizionalmente basati sul valore del parametro scope
7. **Client role**: i clients possono definire ruoli specifici.

Brevemente, di seguito, saranno spiegati di seguito i passi necessari per la configurazione di Keycloak

2.1 Creazione di un realm

2.2 Creazione di un client

2.3 Creazione di un ruolo

2.4 Dimostrazione autenticazione

3 Git actions

Successivamente è stato ritenuto opportuno ai fini di un buon flusso di implementazione integrare delle git actions nei vari repository. In particolare: - deploy con maven - push dell'immagine in un registry di docker in modo tale da poter effettuare i test su minikube prima del deploy su AWS

Link della repository https://hub.docker.com/repositories

4 Kubernetes

Una volta messe a disposizione le immagini, è stato trovato opportuno iniziare ad utilizzare kubernetes [... inserisci motiviazioni]. È stato trovato efficiente ed opportuno utilizzare k8s utilizzando immagini docker.

Per prima cosa è stata dedicata attenzione alla creazione del db. Infatti, come accadeva nel docker-compose, l'obbiettivo sarebbe quello di istanziare dei db, indipendenti dal be, con dei dati pre caricati. L'opzione migliore è stata quella di montare un volume ... download dei dati raw da git (in modo da poter facilmente prendere le eventuali modifiche successive) ...

È stato creato un file di configurazione unico, in quanto le secret sono tutte uguali ...

References

[1] Getting started with Keycloak

[2] Keycloak server installation

[3] Easily secure Spring Boot app with Keycloak

[4] Spring boot with keycloak