Architettura dei sistemi software

Progetto per l'A.A. 2019/2020

30 maggio 2020

Premessa

Il progetto del corso di Architettura dei sistemi software è relativo alla sperimentazione pratica di alcune tecnologie studiate durante il corso, e riguarda la realizzazione di una semplice applicazione a servizi, nonché il rilascio di questa applicazione in un opportuno ambiente di esecuzione.

Il progetto può essere realizzato in diverse varianti, descritte nelle seguenti sezioni.

Progetto (punto di partenza)

Instagnam è un semplice social-network per la condivisione di ricette di cucina. Il codice di **Instagnam** è disponibile sul repository GitHub del corso (https://github.com/aswroma3/asw/tree/master/projects/asw-instagnam).

Instagnam può essere usato come segue. Gli utenti del sistema possono pubblicare delle ricette. Possono poi seguire altri utenti del sistema. Quando un utente accede la pagina delle ricette che segue, gli vengono mostrate le ricette degli altri utenti che segue.

L'applicazione **Instagnam** è composta dai seguenti microservizi:

- Il servizio **ricette** gestisce le ricette di cucina dei suoi utenti. Ogni ricetta ha un autore, un titolo e una preparazione. Operazioni:
  - POST /ricette aggiunge una nuova ricetta (dato autore, titolo e preparazione)
  - GET /ricette/{id} trova una ricetta dato l'id
  - GET /ricette trova tutte le ricette (in formato breve, solo id, autore e titolo)
  - GET /ricette?autore={autore} trova tutte le ricette di autore (solo id, autore e titolo)
- Il servizio **connessioni** gestisce le connessioni tra gli utenti. Ogni connessione è una coppia follower-followed, in cui follower e followed sono due utenti del sistema. Operazioni:
  - POST /connessioni aggiunge una nuova connessione (dato follower e followed)
  - GET /connessioni/{id} trova una connessione dato l'id
  - GET /connessioni trova tutte le connessioni
  - GET /connessioni?follower={utente} trova tutte le connessioni di utente (quelle in cui l'utente è follower)
- Il servizio **ricette-seguite** consente a un utente di trovare le ricette di tutti gli utenti che segue. Operazioni:
  - GET /ricetteseguite/{utente} trova tutte le ricette seguite da utente, ovvero le ricette di utenti di cui l'utente è follower (ricette in formato breve, solo id, autore e titolo)
- Il servizio **api-gateway** (esposto sulla porta 8080) è l'API gateway dell'applicazione, che:
  - espone il servizio **ricette** sul path /ricette; ad esempio, GET /ricette/ricette
  - espone il servizio **connessioni** sul path /connessioni;
    ad esempio, GET /connessioni/connessioni
  - espone il servizio **ricette-seguite** sul path /ricette-seguite;
    ad esempio, GET /ricette-seguite/ricetteseguite/{utente}

Sul repository GitHub del corso, l'implementazione dell'operazione GET /ricetteseguite/U del servizio **ricette-seguite**, per trovare le ricette seguite dall'utente U, è basata su invocazioni remote REST ai servizi **connessioni** e **ricette**:

- prima viene invocata l'operazione GET /connessioni?follower=U di **connessioni** per trovare l'insieme AA di tutti gli utenti seguiti dall'utente U
- poi, ripetutamente, per ciascun utente A nell'insieme AA, viene invocata l'operazione GET /ricette?autore=A di **ricette**, in modo da trovare, complessivamente, le ricette degli autori nell'insieme degli utenti AA seguiti da U

Progetto (attività)

Il progetto consiste nel modificare il codice dell'applicazione presente sul repository GitHub del corso, svolgendo le seguenti attività:

- Nei servizi **ricette** e **connessioni**, usare una base di dati MySQL o PostgreSQL al posto di HSQLDB (ciascuna da eseguire in un contenitore Docker separato).
- Probabilmente va cambiata l'inizializzazione delle basi di dati.
- Invertire la logica del servizio **ricette-seguite** (descritto nel seguito).

Come invertire la logica del servizio **ricette-seguite**:

- Quando viene aggiunta una nuova ricetta, il servizio delle **ricette** deve notificare un evento **RicettaCreatedEvent** (solo con id, autore, e titolo della ricetta), su un canale Kafka oppure RabbitMQ (da eseguire in un contenitore Docker).
- Quando viene aggiunta una nuova connessione, il servizio delle **connessioni** deve notificare un evento **ConnessioneCreatedEvent**, su un canale Kafka oppure RabbitMQ.
- Il servizio **ricette-seguite** deve gestire una propria base di dati (separata dalle precedenti, in un contenitore Docker separato), con una tabella per le ricette e una per le connessioni.
- Ogni volta che il servizio **ricette-seguite** riceve un evento **RicettaCreatedEvent**, deve aggiornare di conseguenza la propria tabella delle ricette.
- Ogni volta che il servizio **ricette-seguite** riceve un evento **ConnessioneCreatedEvent**, deve aggiornare di conseguenza la propria tabella delle connessioni.
- Il servizio **ricette-seguite** deve poi rispondere alle richieste GET /ricetteseguite/{utente} accedendo solo alla propria base di dati.

Una variante più complessa:

- Il servizio **ricette-seguite** deve gestire una propria base di dati (separata dalle precedenti, in un contenitore Docker separato), con una tabella per le ricette, una per le connessioni, e una tabella ricetteseguite che memorizza righe (utenteFollower, idRicetta, autoreRicetta, titoloRicetta).
- Ogni volta che il servizio **ricette-seguite** riceve un evento **RicettaCreatedEvent**, deve aggiornare di conseguenza la propria tabella delle ricette e la tabella ricetteseguite.
- Ogni volta che il servizio **ricette-seguite** riceve un evento **ConnessioneCreatedEvent**, deve aggiornare di conseguenza la propria tabella delle connessioni e la tabella ricetteseguite.
- Il servizio **ricette-seguite** deve poi rispondere alle richieste GET /ricetteseguite/{utente} consultando solo la propria tabella ricetteseguite.

Ulteriori varianti:

- Eseguire i diversi servizi in contenitori Docker, usando Docker Compose.
- Eseguire i diversi servizi in contenitori Docker, usando Kubernetes.
- Mandare in esecuzione più istanze di ciascun servizio.

Altri progetti

Ciascun gruppo può formulare, se vuole, una propria proposta progetto (che deve comunque avere finalità simili a quelle dei progetti illustrati in questo documento). In ogni caso, queste proposte di progetti alternativi devono essere autorizzate dal docente.

Modalità di svolgimento e di consegna del progetto

Il progetto va svolto in gruppi di 3-5 studenti (preferibilmente di 4 studenti).

Ciascun gruppo dovrà interagire con il docente in questo modo:

- Ciascun gruppo dovrà comunicare al docente, per posta elettronica, la composizione del proprio gruppo, le tecnologie che si pensano di utilizzare (se diverse da quelle proposte), oppure la propria proposta di progetto (se diversa da quelle illustrate in questo documento).
- Poi, ciascun gruppo dovrà realizzare e verificare (sul proprio computer) la propria applicazione distribuita.
- Infine, ciascun gruppo dovrà caricare la propria applicazione distribuita su GitHub (o altro servizio di condivisione del codice). In particolare, dovrà caricare tutto il codice sorgente dell'applicazione, oltre ad ogni script necessario per la compilazione e costruzione dell'applicazione (Maven o Gradle), i file Dockerfile e gli eventuali file per Docker Compose o Kubernetes, nonché gli script per mandare in esecuzione l'applicazione.
- Al completamento del progetto, il gruppo dovrà comunicare al docente, per posta elettronica, l'URI su GitHub del codice dell'applicazione.