Contents

List of Figures

0.1 Introduzione

Durante la terza iterazione ci siamo concentrati principalmente sulla creazione di un gateway che mettesse in comunicazione l'applicazione mobile con il backend creato nella precedente iterazione. Abbiamo inoltre implementato, attraverso le GitHub Actions, un meccanismo di Continuous Integration e Deplyment per gestire in automatico le release della nostra architettura. Infine abbiamo lavorato sugli use case a media priorità e abbiamo aggiornato alcune parti della documentazione e dei diagrammi scritti nelle precedenti iterazioni.

0.2 Continuous Integration e Continuous Deployment (CI/CD)

Attraverso le GitHub Actions abbiamo implementato un meccanismo di Continuous Integration e Deployment che ci permette di lavorare sul nostro codice sorgente e non dover provvedere manualmente alla compilazione e pubblicazione dell'immagine Docker con la quale abbiamo pubblicato il nostro backend. La parte di CI avviene direttamente su GitHub, appunto grazie alle Actions, mentre la parte di CD avviene sulla macchina virtuale di Digital Ocean che abbiamo usato per pubblicare l'architettura.

0.2.1 Continuous Integration (CI)

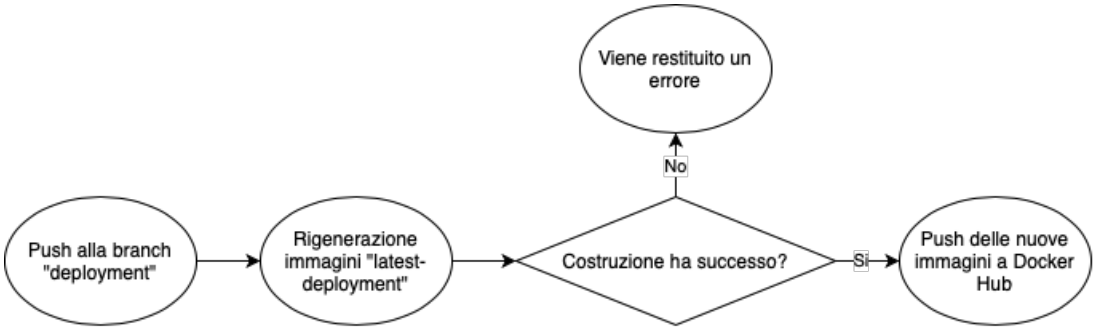

Figure 1: Continuous Integration della branch main

Il nostro backend è organizzato in 2 immagini:

1. "ventura-boulevard", contenente il manager degli utenti e degli eventi;
2. "radio-nowhere", contenente l'API Gateway usato per comunicare con la mobile app.

Entrambe le immagini sono organizzate in 2 tag:

1. "latest-development", l'ultima versione del codice usato per lo sviluppo;
2. "latest-deployment", l'ultima versione del codice usato in produzione.

Infine la nostra repository è organizzata attorno a 2 branch principali (oltre a quelle sviluppate al bisogno da ogni membro, ma che non vengono gestite dal sistema CI/CD):

1. “main”, rappresentante la nostra branch di sviluppo, dove poniamo il codice a cui stiamo ancora lavorando ma che non vogliamo pubblicare sul server di produzione;

2. “production”, ovvero la nostra branch di produzione dove è contenuto il codice che attualmente è pubblicato e viene usato dal backend e dalla mobile app.

Ad ogni push alla branch main o production viene rigenerata e ri-pubblicata su Docker Hub, rispettivamente, l’immagine taggata come “latest-development” e “latest-production”. Per pubblicare l’immagine compilata sul relativo registry di Docker Hub è necessario accedere con le credenziali del proprietario. Per permette di usarle senza che vengano mai divulgate al pubblico, GitHub mette a disposizione “Secrets”, che possiamo vedere come un dizionario dove ogni coppia chiave-valore rappresenta una costante che può essere richiamata all’interno delle Actions attraverso la sua chiave. Una volta creata una entry non è più possibile, neppure per l’autore, vederne il contenuto, ma unicamente modificarla o eliminarla.

0.2.2 Continuous Deployment (CD)

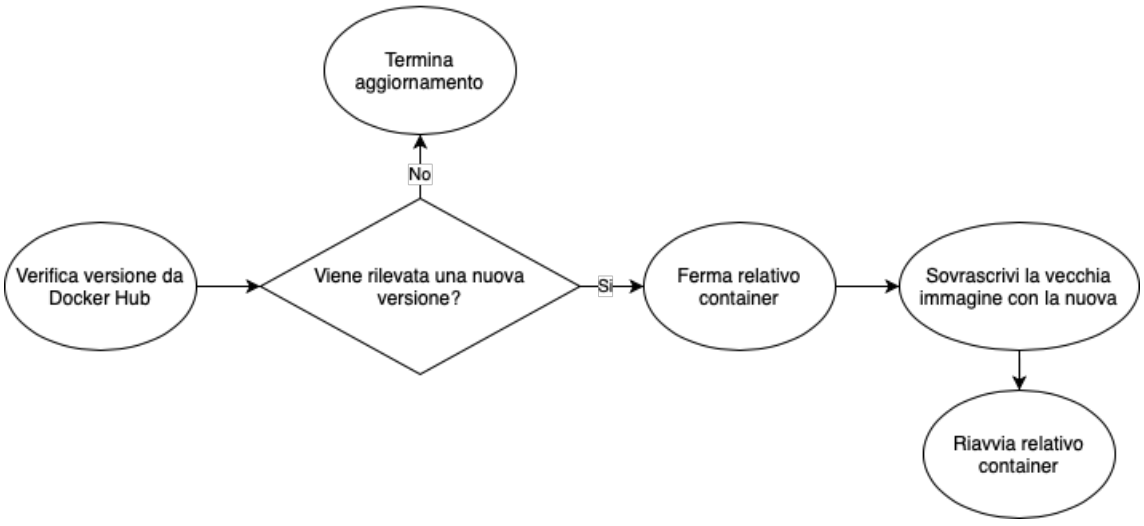

Figure 2: Continuous Deployment di un immagine

Una volta che l’immagine su Docker Hub è stata rigenerata è necessario che questi cambiamenti vengano rispecchiati sul server di produzione. Per fare ciò abbiamo usato “Watchtower”, un programma open-source usato per gestire le immagini Docker

salvate su una macchina aggiornandole in automatico quando viene pubblicata una nuova versione. Il controllo e l'aggiornamento avvengono a intervalli regolari impostabili dall'utente. Per evitare che il server venisse riavviato troppe volte durante la giornata abbiamo deciso di limitare l'aggiormamento ad una volta al giorno, verso mezzanotte. Rimane poi la possibilità di accedere alla macchina virtuale e riavviare manualmente il container, oppure, scelta meno consigliata, riavviare per intero il server.

0.3 Creazione dell'API Gateway

0.4 Casi d'uso

0.4.1 UC-5 Visualizzazione Meteo

Breve introduzione.

Breve Descrizione:

Attori Coinvolti:

Precondizione:

Postcondizione:

Procedimento:

1. item1;
2. item2;
3. item3.

Eccezioni:

- E1:
    1. item1;
    2. item2;
    3. item3;

0.4.2 UC-6 Visualizza eventi consigliati

Breve introduzione.

Breve Descrizione:

Attori Coinvolti:

Precondizione:

Postcondizione:

Procedimento:

1. item1;
2. item2;
3. item3.

Eccezioni:

- E1:
    1. item1;
    2. item2;
    3. item3;

0.4.3 UC-13 Visualizza il profilo di un utente

Breve Descrizione: Il sistema consente la visualizzazione dei profili del singolo utente, al fine di verificarne il livello e le escursioni svolte in precedenza.

Attori Coinvolti: Utente, sistema

Precondizione: L'utente deve essere registrato e deve aver eseguito l'accesso alla piattaforma.

Postcondizione: Il sistema mostra una nuova schermata in cui le informazioni generali dell'utente vengono mostrate.

Procedimento:

1. L'utente seleziona un profilo;
2. Il sistema mostra il profilo dell'utente selezionato, con annessa tipologia di utente;

Eccezioni: Durante il procedimento non si possono verificare eccezioni in quanto solamente i profili accessibili potranno essere visualizzati.

0.4.4 UC-14 visualizza il profilo di un organizzatore

Breve introduzione.

Breve Descrizione:

Attori Coinvolti:

Precondizione:

Postcondizione:

Procedimento:

1. item1;
2. item2;
3. item3.

Eccezioni:

- E1:
    1. item1;
    2. item2;
    3. item3;