Contents

0.1 Introduzione 4

0.2 Continuous Integration e Continuous Deployment (CI/CD) 5

0.2.1 Continuous Integration (CI) 5

0.2.2 Continuous Deployment (CD) 6

0.3 API Gateway 8

0.3.1 Creazione dell'API Gateway 8

0.3.2 Testing del gateway 9

0.4 Casi d'uso 10

0.4.1 UC-4 Visualizza iscrizioni 11

0.4.2 UC-5 Visualizzazione Meteo 12

0.4.3 UC-6 Visualizza eventi consigliati 13

0.4.4 UC-13 Visualizza il profilo di un utente 14

0.4.5 UC-14 visualizza il profilo di un organizzatore 15

0.4.6 UC-6 Visualizza dettagli evento 16

List of Figures

1 Continuous Integration della branch main 5
2 Continuous Deployment di un immagine 6
3 Risultati testing gateway 9

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)

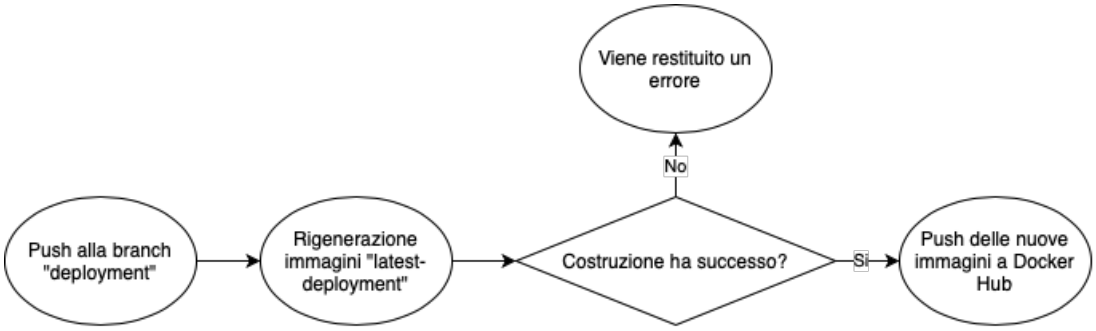

Figure 1: Continuous Integration della branch main

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)

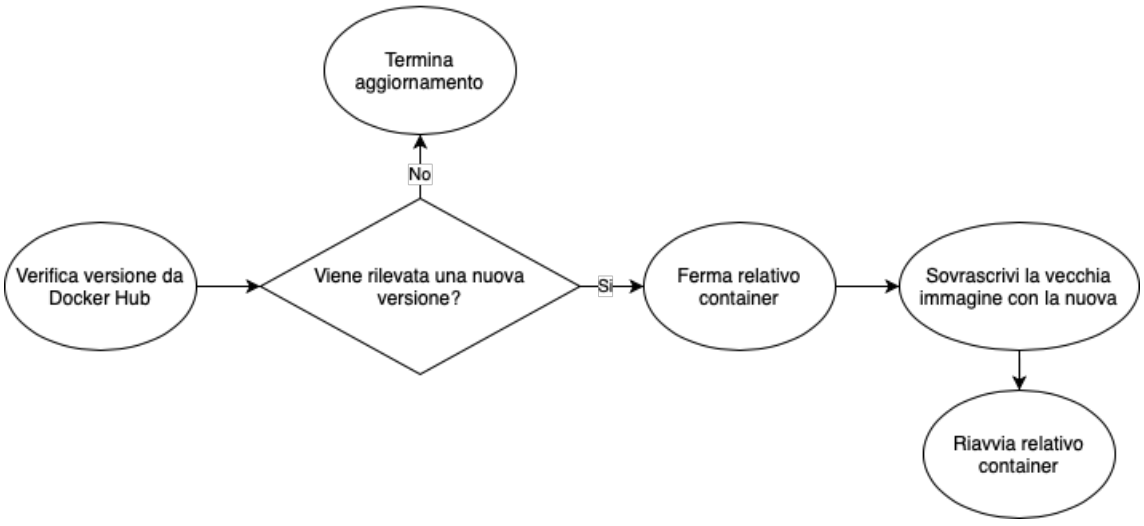

Figure 2: Continuous Deployment di un immagine

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 API Gateway

0.3.1 Creazione dell'API Gateway

Per completare la nostra architettura e disporre di un modo per rispondere in maniera più efficace ed efficiente alle richieste del client abbiamo deciso di creare un API Gateway che facesse da tramite tra il backend scritto in Spring (il "sistema complesso") e l'applicazione mobile scritta in Dart. Per fare ciò abbiamo optato per un terzo linguaggio di programmazione, NodeJS, poiché volevamo realizzare un componente che fosse veloce e leggero. Il maggior compito del nostro gateway è quello di interrograre il sistema complesso e comporre le risposte da inviare al client. Grazie al gateway abbiamo riscontrato molti vantaggi mentre lavoravamo all'architettura, tra questi troviamo:

- La possibilità di modellare la risposta proveniente dal server in modo da fornire particolari header necessari affinché le risposte fossero accetate dal client. Per esempio includendo l'header 'Access-Control-Allow-Origin', richiesto da Dart per accettare una risposta, ma che non era
  fornito dl sistema complesso;

- Permettere a client di ottenere oggetti composti dal sistema complesso che senza l'utilizzo del gateway avrebbero necessitato di numerose chiamate da parte del client. Un esempio è la chiamata di login, la quale, quando l'utente viene autenticato, restituisce informazioni su di esso come nome e cognome. Nella risposta fornita dal Managaer degli Utenti tuttavia non viene fornito il livello di esperirenza. Tuttavia è disponibile una chiamata che soddisfa questa richiesta, dato l'id di un utente. Chiamando questo secondo metodo, dopo aver ottenuto i dati dell'utente, ci ha permesso di includere il campo "experience" nella risposta al client.

0.3.2 Testing del gateway

Per verificare il corretto funzionamento del gateway abbiamo scritto una serie di test per ognuna delle chiamate verificando che la risposta attesa fosse corretta. Seguono i risultati della test suite (3).

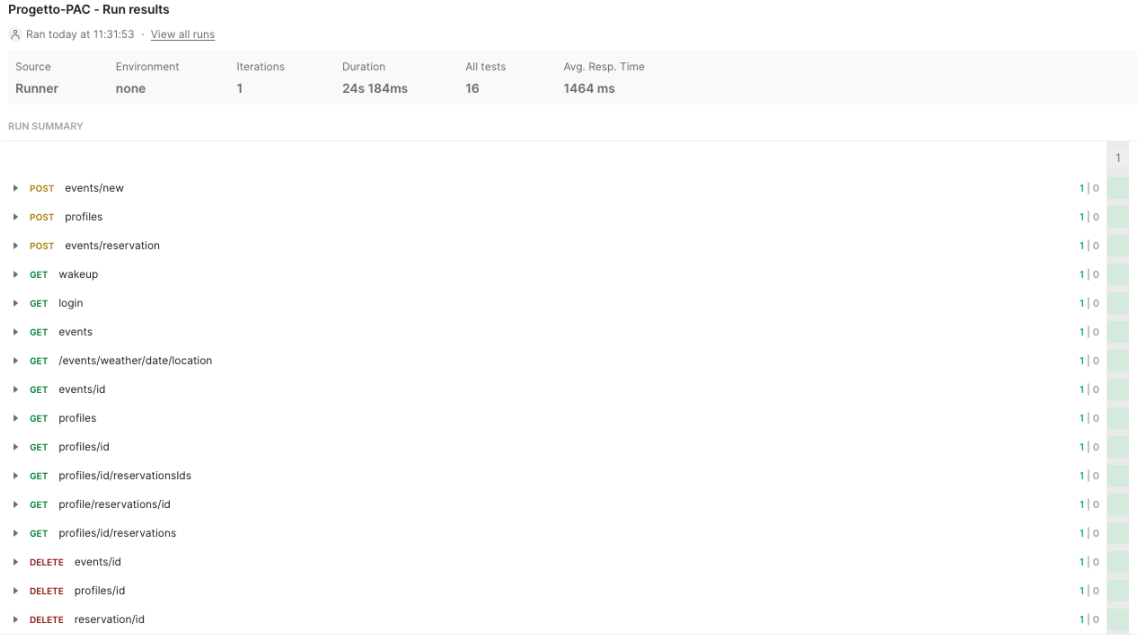

Figure 3: Risultati testing gateway

0.4 Casi d'uso

0.4.1 UC-4 Visualizza iscrizioni

Breve Descrizione: il sistema consente all'utente di visualizzare gli eventi a cui è attualmente iscritto.

Attori Coinvolti: Sistema, Utente

Precondizione: L'utente è registrato e ha effettuato l'accesso.

Postcondizione: L'utente visualizza le proprie iscrizioni agli eventi

Procedimento:

1. L'utente accede alla app e visualizza la homepage;
2. L'utente entra nel proprio profilo e seleziona gli eventi in programma;
3. Il sistema mostra una lista contente gli eventi a cui l'utente è iscritto e a cui non ha ancora partecipato. [E1: Pagina vuota];

Eccezioni:

- E1: Pagina vuota
    1. Viene mostrato all'utente un messaggio indicante che al momento non ci sono iscrizioni in corso;

0.4.2 UC-5 Visualizzazione Meteo

Per ogni evento, entro 5 giorni dalla sua data, l'utente può visualizzare una breve del meteo.

Breve Descrizione: Usando la posizione e il giorno dell'evento si ottengono le condizioni meteo da mostrare per ogni evento solo prima che manhino 5 giorni allo stesso.

Attori Coinvolti: Sistema, Visual Crossing Weather

Precondizione: Mancano almeno 5 giorni da un evento

Postcondizione: Viene mostrato il meteo per il giorno e il luogo dell'evento

Procedimento:

1. Quando l'escursione viene preparata dal gateway si verifica se manchino meno di 5 giorni;
2. In caso affermativo viene invocata l'API di Visual Crossing Weather per ottenere le previsioni [E1: Errore nell'ottenere il meteo];
3. In caso negativo il campo del meteo viene lasciato vuoto.

Eccezioni:

- E1: Errore nell'ottenre il meteo
    1. Viene rilevato l'errore attraverso lo statusCode ritornato dall'API;
    2. Il campo del meteo viene lasciato vuoto, come se mancassero più di 5 giorni all'evento.

0.4.3 UC-6 Visualizza eventi consigliati

Da una pagiona dedicata sulla app, l'utente può visualizzare una lista di escursioni curata automaticamente in base alla sua esperienza.

Breve Descrizione: Visualizzando la pagina "per te" viene presentata una lista di tutti gli eventi disponibili filtrata in base alla loro difficoltà

Attori Coinvolti: Sistema, Utente

Precondizione: L'utente è loggato nella app

Postcondizione: L'utente visualizza gli eventi consigliati

Procedimento:

1. L'utente accede alla app e si ritrova sulla homepage;
2. L'utente seleziona la pagina "per te" dal menu in basso nella pagina;
3. Il sistema filtra la lista di eventi scaricata dalla hompage;
4. Il sistema mostra una lista contente solo gli eventi con una diffocoltà<=all'esperienza dell'utente [E1: Pagina vuota];

Eccezioni:

- E1: Pagina vuota
    1. Viene mostrato all'utente un messaggio indicante che al momento non ci sono eventi consigliati;

0.4.4 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.

0.4.5 UC-14 visualizza il profilo di un organizzatore

Similmente ad UC-13 (0.4.4) è possibile visualizzare il profilo di un utente.

Breve Descrizione: La differenza con UC-13 sta nei dati aggiuntivi che vegnono mostrati se l'utente è organizzatore, per esempio il badge di "organizzatore"

Attori Coinvolti: Sistema, Utente

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.

0.4.6 UC-6 Visualizza dettagli evento

L'utente può visualizzare informazioni dettagliate riguardo ad un singolo evento.

Breve Descrizione: Dalla pagina dell'evento, è possibile consultare dettagli come il luogo, l'ora, la distanza, ecc.

Attori Coinvolti: Sistema, utente

Precondizione: L'utente è registrato e ha effettuato l'accesso all'app

Postcondizione: Vengono mostrati tutti i dettagli dell' evento

Procedimento:

1. L'utente accede alla app;
2. L'utente accede alla sezione 'per te' e seleziona un evento di interesse o ne ricerca e seleziona uno dalla barra di ricerca;
3. L'utente visualizza la pagina dell'evento con tutti i dettagli;

Eccezioni:

- E1: Problema di connessione
    1. Messaggio di errore;
    2. Viene chiesto all'utente di riprovare;