

**Distributed Systems and Big Data 2023-2024**

Weather Event Notifier

Alessandro Genovese 1000002043

Francesco Pennisi 1000055702

INDICE

[INTRODUZIONE 5](#_Toc114680447)

[1. Abstract 8](#_Toc114680448)

[2. Architettura del sistema 14](#_Toc114680455)

[3. Sistema realizzato 21](#_Toc114680462)

[3.1. Comunicazioni 21](#_Toc114680463)

[3.2. Gestione repliche 22](#_Toc114680464)

[3.3. Sicurezza Secrets 22](#_Toc114680465)

[3.4. 24](#_Toc114680466)

[3.5. 24](#_Toc114680467)

[3.6. Serie temporali analizzate 25](#_Toc114680468)

[3.7. Acquisizione ed elaborazione dei dati 26](#_Toc114680469)

[3.8. Memorizzazione 29](#_Toc114680470)

[4. Monitoraggio QoS 32](#_Toc114680471)

[4.1. Prometheus 32](#_Toc114680472)

[4.2. Cadvisor 32](#_Toc114680473)

[4.3. SLA\_manager 32](#_Toc114680474)

[4.4. Forecasting 33](#_Toc114680475)

[4.5. Serie temporali analizzate 25](#_Toc114680468)

5. BUILD E DEPLOY SU README.md

[INDICE DELLE FIGURE 34](#_Toc114680476)

[RINGRAZIAMENTI 35](#_Toc114680477)

[BIBLIOGRAFIA 36](#_Toc114680478)

1. **Abstract**

Lo scopo di questo progetto è la progettazione e lo sviluppo di un sistema distribuito che possa essere eseguito su piattaforme architecture-independent.

Il sistema in oggetto è realizzato seguendo un pattern architetturale a microservizi, i quali vengono impacchettati in appositi container, facendo uso della tecnologia di containerizzazione Docker, che permette di raccogliere e isolare i microservizi in ambienti runtime completi, corredati di tutti i file necessari per l’esecuzione in modo da garantire la portabilità su qualunque infrastruttura (hardware e software) che sia docker-enabled.

Il sistema in esame ha l’obiettivo di permettere agli utenti registrati di indicare, per ogni località di interesse, dei parametri meteorologici, ad esempio la massima temperatura o l’eventuale presenza di pioggia. Tali parametri saranno monitorati dal sistema stesso. In caso di violazione delle condizioni meteo specificate, gli utenti verranno notificati dal sistema tramite e-mail.

A tale scopo, ad intervalli regolari, anch’essi a discrezione dell’utente, l’applicativo recupera le informazioni meteorologiche appoggiandosi al servizio terzo OpenWeather (<https://openweathermap.org/api>) mediante richieste REST API. Tali informazioni vengono filtrate e opportunamente elaborate sulla base delle condizioni sottomesse dagli utenti.

1. **Architettura del sistema**

Il sistema, dal punto di vista funzionale, è composto dai seguenti microservizi: Weather Management Service (abbreviato in WMS), User Manager Service (UM), Worker Service e Notifier Service.

Inoltre, è stata prevista l’introduzione di un’attività di QoS management che prevede l’inserimento nell’architettura del sistema di tre ulteriori microservizi: Prometheus, Cadvisor e Service Level Agreement (SLA) Manager, i quali sono trattati in seguito (#TODO: indicare nome paragrafo).

Immagine che contiene testo, diagramma, schermata, Piano

Descrizione generata automaticamente

Figura 1 Architettura del sistema

In breve sono spiegati i microservizi utilizzati:

* **User Manager Service (UM):** garantisce la registrazione e la successiva autenticazione degli utenti nel sistema. L’autenticazione sfrutta la tecnologia JWT token, dunque l’utente attraverso l’utilizzo di un client (e.g. *Postman*) effettua il login tramite richiesta REST API e ottiene un token JWT, il quale viene memorizzato e inserito nell’header *HTTP Authorization* in tutte le successive richieste effettuate dall’utente.

Nel momento in cui l’utente volesse inserire/modificare dei parametri meteorologici per una determinata località contatterà il microservizio WMS inviando il token di autorizzazione.

Quest’ultimo microservizio autenticherà l’utente passando il token allo User Manager, il quale dopo aver validato il token restituirà al WMS le informazioni necessarie affinché la richiesta dell’utente possa essere soddisfatta con successo.

L’UM è collegato a un suo Database MySQL che include la tabella Users. Tale tabella contiene le informazioni relative all’ utente: indirizzo e-mail, password e codice identificativo univoco dell’utente.

* **Weather Management Service (WMS):** permette all’utente l’inserimento, la modifica e l’eliminazione dei parametri meteorologici relativi a una o più località a scelta dell’utente.

Il WMS è collegato a un suo Database MySQL che include la tabella *Locations*, in cui vengono memorizzate le informazioni relative alle località di interesse degli utenti, e la tabella *UserConstraints*.

Quest’ultima contiene le regole specificate dagli utenti, un *trigger\_period* che indica ogni quanti minuti l’utente desidera che il sistema monitori le regole sottoscritte, infine un timestamp che viene aggiornato ogni volta che il WMS prende in considerazione la specifica entry.

Il WMS ciclicamente, attraverso l’utilizzo di un timer, controlla la tabella UserConstraints e di volta in volta prende in considerazione solo le entries per cui è trascorso il *trigger\_period*.

Il compito del WMS in sostanza è quello di costruire dei messaggi Kafka da pubblicare sul topic *event\_update*, del quale il microservizio Worker è sottoscrittore.

Tali messaggi includono le regole degli utenti di cui il Worker dovrà controllare le eventuali violazioni. La costruzione del messaggio è approfondita al paragrafo ()

* **Worker Service:** ha la responsabilità di contattare il servizio OpenWeather per ottenere i dati metereologici attuali e verificare l’eventuale violazione dei parametri specificati dagli utenti.

Ciò avviene seguendo questo ordine procedurale: il worker preleva dal topic Kafka *event\_update* i messaggi pubblicati dal WMS, estrarre le regole da controllare, contatta il servizio OpenWeather, confronta i risultati ottenuti con quelli ricevuti dal WMS.

Nel caso in cui si presentino delle violazioni il worker le pubblicherà in un messaggio sul topic Kafka *event\_to\_be\_notified*. All’interno del messaggio ogni parametro violato è associato all’utente che ne ha richiesto il monitoraggio tramite il suo codice identificativo univoco.

* **Notifier Service:** il suo compito è quello di prelevare eventuali messaggi Kafka pubblicati sul topic *event\_to\_be\_notified* dal worker.

Una volta estratto il messaggio si occupa di avvisare gli utenti delle avvenute violazioni mediante l’invio di una mail.

Il notifier è in grado di recuperare l’e-mail dell’utente contattando lo User Manager tramite il meccanismo di comunicazione gRPC, è possibile identificare univocamente l’utente grazie al suo codice identificativo contenuto nel messaggio Kafka.