Progetto prova in itinere DSBD aa2023-2024

Nota: l'elaborato prevede due fasi.
La prima fase, prevede la progettazione e l'implementazione di una applicazione distribuita composta da diversi microservizi, deployati come docker container.

La seconda fase prevede un'evoluzione dello stesso progetto. Lo scopo sarà quello:

- irrobustire le comunicazioni e garantire un determinato livello di consistenza con pattern specifici (e.g., Circuit Breaker, SAGA)
- introdurre le attività di QoS Management e aggiungere algoritmi predittivi per anticipare comportamenti anomali del sistema.
- deploy in ambiente kubernetes

Descrizione delle attività della prima fase.

Lo studente (ovvero il gruppo) può decidere se sviluppare un elaborato con un tema di proprio interesse, oppure fare riferimento ad uno degli esempi di applicazioni forniti in questo documento.
In ogni caso, prima di sviluppare l'elaborato, è necessario inviare una bozza dell'idea progettuale al gruppo di docenti. (antonella.distefano@unict.it, giovanni.morana@unict.it, tutor massimo.gollo@phd.unict.it)

Esempi di applicazione
Applicazione 1 - weather event notifier: l'applicazione prevede agli utenti di sottoscriversi ad eventi meteorologici nelle città di interesse. L'utente può sottoscriversi ad un insieme di città, fornendo per ognuno una serie di constraint (es. temperatura massima, temperatura minima, quantità di pioggia, presenza di neve). Qualora si verifichino le condizioni specificate, il sistema restituisce un alert all'utente (la modalità di notifica è a scelta, ad esempio: SMS, telegram, email, sfruttando eventualmente servizi esterni che offrono tali funzionalità). A tale scopo, ad intervalli regolari (a discrezione dello studente), recupera le informazioni meteorologiche tramite rest API da un servizio esterno (es. https://openweathermap.org/api), le filtra e le elabora sulla base delle condizioni sottomesse dagli utenti

Applicazione 2 - Traffic event notifier: Analogo al caso precedente ma sul contesto del controllo del traffico. Gli utenti si sottoscrivono agli eventi sulla quantità di traffico presente in una tratta definita da una locazione di partenza ad una di arrivo in determinate fasce orarie. Sfruttando le API offerte da servizi esterni (es. Google Maps), l'utente può specificare un percorso, un relativo livello di traffico e un tempo soglia oltre al quale si deve essere notificati. Il sistema, sulla base delle preferenze dell'utente, ottiene ed elabora periodicamente i dati e notifica l'utente se nella fascia oraria sarà presente un livello di traffico superiore a quello specificato.

Applicazione 3: Stock Portfolio Management: il sistema consente la gestione dei portafogli utente dove vengono inserite un certo numero di stock/azioni/crypto con una data e un valore di acquisto. Il sistema ottiene le informazioni sulle azioni a cui l'utente è sottoscritto, interroga il sistema esterno per ottenere i risultati, li elabora e mostra guadagno/perdita dell'interno portafoglio sulla base dei guadagni/perdite delle singole azioni.

Note di progetto

Tutte le tracce sono predisposte per la composizione in microservizi (es. notifier, user management, scraper). Le scelte progettuali (database utilizzati, decomposizione in microservizi, gestione dei topic kafka, eventuali pattern di comunicazione) sono oggetto di valutazione e devono essere opportunamente motivate. Ogni gruppo può liberamente personalizzare il progetto, considerando che la valutazione delle scelte progettuali prevale sull'implementazione del codice. **NON** si richiede alcuna interfaccia grafica o qualsiasi forma di frontend e non incide in alcun modo sulla valutazione. E' sufficiente interagire tramite REST con il sistema. La scelta dei linguaggi di programmazione non è influente ma per maggior supporto si consiglia java,python o golang per il backend.

Descrizione delle attività della seconda fase.

La seconda fase è dedicata alla gestione della qualità del servizio dell'applicazione realizzata nella prima fase. Si configuri un server Prometheus per fare scraping delle metriche esposte da eventuali exporter. La decisione di quali elementi monitorare è a discrezione dello studente. Ad esempio

- performance del nodo sulla quale è deployata l'app (node exporter)
- performance dei container (cAdvisor)
- performance applicative tramite custom exporter (richiede la strumentazione dell'applicazione instrumented per esporre metriche custom generate da/dai microservizio/i)

SLA manager

Servizio che permette di definire un SLA dell'applicazione come composizione di un set di metriche collezionate da prometheus al punto precedente. Deve quindi esporre un endpoint per aggiungere/rimuovere una metrica ad SLA. Per ogni metrica in SLA, SLA Manager deve essere in grado di fornire:

- valore attuale attuale della metrica
- valore desiderato
- stato della metrica rispetto l'SLA - > violazione (true/false)
- numero di violazioni in un arco temporale predeterminato
- probabilità di violazione nel prossimo intervallo di tempo X.

Le informazioni gestite da SLA Manager sono memorizzate su un DB a scelta e rese persistenti.

SLA manager deve esporre una interfaccia (REST) che permette di recuperare le suddette informazioni. Si rendano dunque disponibili, almeno, le seguenti API:

- CREATE/UPDATE del SLA (definizione del set di metriche e, per ogni metrica, range di valori ammissibili)
- QUERY dello stato del SLA
- QUERY del numero di violazioni nelle ultime 1,3, 6 ore.
- QUERY della probabilità di violazioni nei prossimi X minuti.

Modalità e tempi di consegna

Il progetto completo va consegnato improrogabilmente entro il 31/01/2024 (hard deadline). Chiunque desideri una revisione della prima fase del progetto, dovrà consegnare quanto svolto prima del 02/01/2024 (soft deadline), notificando i docenti tramite email richiedendo la revisione.

Il codice del progetto dovrà essere consegnato come repository Git (es. su Github).
Assieme al progetto va consegnata:

- una relazione che illustri le scelte progettuali (e.g., descrizione dell'applicazione, schema architetturale - micro-servizi e relative comunicazioni - lista delle API implementate). La relazione deve includere
  - un abstract (circa 250 parole) che descriva l'idea sviluppata;
  - un diagramma architetturale che mostri i micro-servizi coinvolti;
  - un diagramma che mostri le interazioni, sia tra componenti dell'applicazione che tra l'applicazione e il mondo esterno.
- un documento con le informazioni necessarie per build & deploy, da consegnare in formato Markdown o PDF.

Il progetto può essere presentato anche se incompleto.
E' possibile realizzare delle slide da utilizzare durante la discussione del progetto (10-15 min).
La data della discussione verrà stabilità dopo la consegna e la valutazione del progetto.