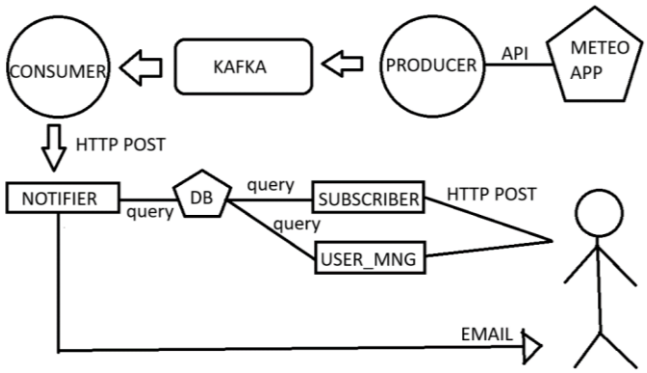
IDEA PROGETTUALE

Nel contesto del progetto di prova in itinere per il corso DSBD per l'anno accademico 2023-2024, vorremmo presentarvi un'architettura proposta per l'applicazione 1 - Weather Event Notifier.



Architettura Proposta
La nostra idea si basa su un'architettura di microservizi, utilizzando **Flask** come framework principale per lo sviluppo, **Kafka** per la messaggistica asincrona, **MySQL** per la gestione dei dati persistenti, **Prometheus** per il monitoraggio, **Docker** per la containerizzazione e **Kubernetes** per l'orchestrazione dei container.

Producer
A intervalli regolari, recupera le informazioni meteorologiche tramite REST API da un servizio esterno (ad esempio, OpenWeatherMap) e li pubblica su dei topic **Kafka,** uno per ogni città.

Consumer
Il **consumer** è progettato per ricevere messaggi dai topic, filtrare i dati di interesse, inserirli all'interno di un JSON e inviarli al microservizio **Notifier**.

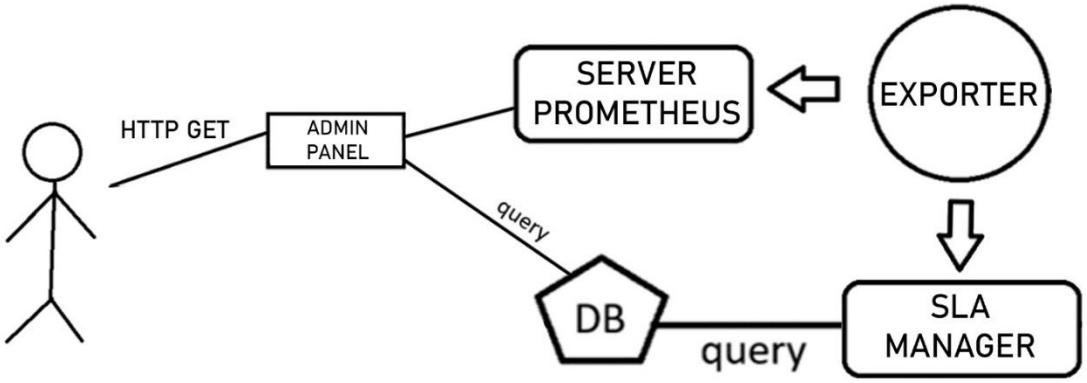
N.B
Consumer e producer sono autenticati con il broker Kafka mediante il meccanismo SASL/PLAIN; quindi, sono richiesti username e password per pubblicare e consumare i topic.

Microservizio "Notifier"
Il **Notifier** riceve i dati meteorologici solo dal **consumer** e invia agli utenti delle e-mail in caso di violazione dei vincoli (verificate tramite query al DB). Esempio: quando l'umidità rilevata supera una certa soglia definita dall'utente.

Microservizio "User Management"
Gestisce le informazioni sugli utenti e la loro autenticazione;

Microservizio "Subscriber"
Questo microservizio consente agli utenti di sottoscriversi a eventi meteorologici specificando vincoli sulle condizioni meteorologiche per le location desiderate (temperatura, umidità, precipitazioni, ecc), e gestisce l'inserimento di queste sottoscrizioni nel database MySQL.
Inoltre, fornisce un servizio per ottenere l'elenco delle città monitorabili.

Client
L'utente interagisce con i microservizi tramite un client, una volta loggato avrà a disposizione un menu e potrà scegliere se visualizzare le sottoscrizioni attive, la lista delle città che può monitorare, le sue informazioni personali (magari desidera cambiarle) e un servizio per effettuare il logout.



La seconda fase del progetto prevede l'utilizzo di un exporter per fornire le performance applicative più importanti al server prometheus e allo sla manager.

Exporter

Riceve dati riguardanti le performance dei nodi (producer,notifier,subscriber,consumer,user manager,exporter), come l'utilizzo della ram e della cpu. Per alcuni nodi come il notifier sono previste anche metriche custom, come per esempio il numero di notifiche giornaliere riguardanti una determinata città.
Possibili metriche da esporre:

- **notifier**: numero notifiche giornaliere, monitoraggio performance container (ram e cpu)
- **producer**: performance (ram e cpu)
- **user management**: numero login giornalieri, numero nuovi utenti giornalieri, accessi non autorizzati giornalieri, performance (ram e cpu)
- **subscriber**: numero di sottoscrizioni per una città in un giorno, performance (ram e cpu)
- **exporter**: performance (ram e cpu)

Server Prometheus

Prometheus si occupa di recuperare le informazioni fornite dall'exporter per visualizzarle (Grafana) e analizzarle.

SLA Manager

Lo SLA Manager ha il compito di monitorare e controllare le possibili violazioni sui contratti SLA. Permette all'amministratore di aggiornare e aggiungere i contratti SLA e definire i valori ammessi.
Memorizza tutte le sue informazioni sul database Mysql.
Possibili metriche SLA:

- L'utilizzo massimo della CPU dei nodi non deve superare il 90%
- L'utilizzo massimo della RAM dei nodi non deve superare il 75%
- La CPU deve avere un utilizzo orario medio che non superi il 70%
- Il servizio deve rispondere entro un certo lasso di tempo

Admin Panel

E' un modulo che permette all'amministratore di sistema di poter monitorare i grafici di prometheus, recuperare e modificare i contratti SLA

Circuit Breaker

E' un modulo che si interpone con tutti i moduli che devono comunicare con il database Mysql (admin panel, sla manager, notifier, subscriber e user manager). Nello specifico viene utilizzato per prevenire il ripetersi di richieste che hanno una maggiore probabilità di fallire, riducendo così il carico su un servizio o un componente e migliorando la gestione delle situazioni di errore.