



UNIVERSITÀ DELLA CALABRIA

DIPARTIMENTO DI
INGEGNERIA INFORMATICA,
MODELLISTICA, ELETTRONICA
E SISTEMISTICA

DIMES

Corso di Laurea in
Ingegneria Informatica

Progetto di Sistemi Distribuiti e Cloud Computing

Professori

Prof. Domenico Talia
Prof. Loris Belcastro

Studente

Carmen Licciardi
235353

Anno Accademico 2021/22

Indice

1. Introduzione.....	3
2. Analisi e specifica dei requisiti	4
2.1 Stakeholders.....	5
2.2 Requisiti funzionali	5
2.3 Requisiti non funzionali	7
3. Progettazione	9
3.1 Formato del file e dei dati	9
3.2 Lo stack ELK.....	10
3.3 Microsoft Azure	11
3.4 Applicativi Python	11
4. Architettura.....	13
5. Configurazione.....	14
5.1 Configurazione Azure Storage	14
5.2 Configurazione Azure Virtual Machines (VM).....	15
5.3 Installazione e configurazione di Elasticsearch e Kibana	15
6. Implementazione.....	17
6.1 Validazione file JSON.....	17
6.2 Collegamento allo Storage di Azure.....	18
6.3 Normalizzazione coordinate	19
6.4 Connessione ad Elasticsearch.....	20
6.5 Definizione indice	20
6.6 Caricamento dati	23
7. Visualizzazione e analisi dei tweet	24
7.1 Grafici interattivi	24
7.2 Filtri.....	28
7.3 Contatori e metriche	30
8. Conclusioni.....	31

1. Introduzione

Il presente progetto ha avuto come obiettivo la progettazione e l'implementazione di un sistema distribuito per la visualizzazione su mappa di grandi moli di tweet geolocalizzati. Si è cercato di offrire agli utenti un'interfaccia intuitiva e potente che permetesse di esplorare e analizzare i dati relativi ai tweet geolocalizzati, consentendo loro di impostare query per il filtraggio dei dati in base a criteri come tag, lingua o parole specifiche.

Nell'ambito di questo progetto, è stata utilizzata la dashboard interattiva Kibana, una piattaforma versatile che ha permesso di creare visualizzazioni personalizzate, quali grafici, mappe e widget, al fine di facilitare la comprensione e l'analisi dei dati. Kibana ha offerto una vasta gamma di funzionalità che hanno consentito agli utenti di esplorare i tweet geolocalizzati in modo efficace e interattivo.

Per la gestione dei dati, è stato utilizzato il database NoSQL Elasticsearch, noto per le sue prestazioni elevate e la scalabilità. I dati dei tweet sono stati importati in batch da file e memorizzati in Elasticsearch, permettendo un accesso rapido ed efficiente alle informazioni.

Durante lo sviluppo dell'applicazione, sono stati presi in considerazione i requisiti funzionali e non funzionali del sistema, al fine di garantire un'esperienza utente ottimale e soddisfare le esigenze degli utenti finali. È stato necessario utilizzare le soluzioni di calcolo, storage e virtualizzazione fornite da Microsoft Azure, che hanno offerto un'infrastruttura affidabile e scalabile per il sistema distribuito.

2. Analisi e specifica dei requisiti

Lo sviluppo del software è stato un compito di vasta portata che ha mirato alla creazione di un prodotto software funzionante. Tale prodotto è stato costruito in base alle specifiche del cliente. Una corretta consegna del prodotto richiede una raccolta accurata dei requisiti, un'efficiente analisi di questi e, infine, una documentazione chiara come prerequisito fondamentale. L'intero processo è comunemente noto come "analisi dei requisiti" nel ciclo di vita dello sviluppo software.

Nel campo dell'ingegneria del software, l'analisi dei requisiti viene frequentemente utilizzata. Al suo interno, vengono elencate in dettaglio tutte le funzionalità che il nuovo prodotto o la modifica di un prodotto devono offrire, ovvero quelle che il software deve soddisfare al termine dello sviluppo. Durante la fase di raccolta, partecipano diverse figure, tra cui il cliente, gli specialisti di analisi di mercato e i membri del team di sviluppo, noti come stakeholder. Gli stakeholder sono persone, gruppi o aziende che detengono interesse nei confronti dell'azienda. Le parti interessate possono influenzare o essere influenzate dalle azioni, dagli obiettivi e dalle politiche dell'organizzazione. Alcuni esempi di stakeholder chiave includono i clienti, i creditori, i manager, i dipendenti, il governo (e le sue agenzie), i proprietari, gli azionisti, i fornitori, i sindacati e la comunità da cui l'azienda attinge le sue risorse.

Successivamente, viene effettuata un'attenta valutazione dei requisiti raccolti imposti dal committente, i quali verranno esaminati dalle figure coinvolte nello sviluppo che ne valutano la fattibilità. Questa è una fase delicata che potrebbe richiedere interazioni multiple tra il cliente e lo sviluppatore nel caso in cui si verifichino incomprensioni. Infine, viene redatto un documento in cui vengono definiti tutti i requisiti che il prodotto dovrà soddisfare.

Per quanto concerne il progetto presentato, è stato adottato l'approccio seguente per condurre un'analisi dei requisiti esaustiva. Il primo passo cruciale è stato l'identificazione e la definizione degli stakeholder, ovvero le parti interessate coinvolte nel progetto. Successivamente, è stata condotta un'analisi dettagliata per definire i requisiti funzionali, cioè le specifiche delle funzionalità e dei servizi che il sistema avrebbe dovuto fornire.

Inoltre, sono stati definiti anche i requisiti non funzionali, che comprendono gli aspetti di prestazione, sicurezza, usabilità e scalabilità del sistema. Questi requisiti mirano a garantire che il sistema soddisfi gli standard di qualità e le aspettative degli utenti in termini di prestazioni e affidabilità.

L'obiettivo di questa fase di analisi dei requisiti è stato quello di stabilire una base solida per la progettazione e l'implementazione del sistema, consentendo di soddisfare le aspettative degli stakeholder e fornire una soluzione efficace e di qualità.

2.1 Stakeholders

Dalla descrizione del progetto, non è stato possibile individuare in modo specifico l'utente finale coinvolto. Di conseguenza, è stato considerato un profilo di utente generico che potrebbe non possedere conoscenze specialistiche nel dominio o essere un esperto nel settore. Tale profilo rappresenta l'utente che accede al servizio per beneficiare di tutte le funzionalità e degli strumenti forniti dal sistema sviluppato, al fine di condurre analisi sui tweet disponibili. L'obiettivo principale è offrire un'esperienza utente accessibile e intuitiva, che consenta a un'ampia gamma di utenti di trarre vantaggio dai dati e condurre analisi significative senza richiedere una competenza approfondita nel dominio specifico.



2.2 Requisiti funzionali

L'analisi dei requisiti rappresenta un processo fondamentale volto a identificare e definire in modo accurato le aspettative degli stakeholder riguardo ai prodotti di un progetto specifico. In genere, ci si trova ad affrontare la sfida di dover realizzare un prodotto entro limiti di tempo ristretti, mantenendo costi minimi e garantendo

un alto livello di qualità. A partire dalla descrizione del progetto in questione, emergono alcune funzionalità essenziali che l'applicazione deve necessariamente presentare:

- Visualizzazione su mappa di grandi quantità di tweet geolocalizzati.
- Possibilità per l'utente di impostare query per il filtraggio dei dati.
- Definizione di una dashboard predefinita con widget e query preimpostate utili per l'utilizzo della piattaforma.
- Utilizzo della dashboard interattiva Kibana.
- Importazione dei dati in batch da file e memorizzazione su un database NoSQL come Elasticsearch.
- Utilizzo delle soluzioni di calcolo, storage e virtualizzazione offerte da Microsoft Azure.

Dai requisiti precedentemente delineati emerge l'importanza di filtrare e archiviare i tweet provenienti da fonti esterne al fine di consentirne la successiva visualizzazione. È di cruciale rilevanza garantire che i dati vengano raccolti in maniera statica, permanente e sicura, al fine di preservarne l'integrità e la disponibilità nel tempo. Inoltre, al fine di creare statistiche e ottenere informazioni significative, è fondamentale che tutti i dati memorizzati nel sistema vengano analizzati in modo coerente e strutturato.

I tweet rappresentano una risorsa preziosa per gli utenti generici che desiderano monitorare le attività sulla piattaforma e ottenere una visione d'insieme delle tendenze e dei contenuti condivisi. Al contempo, le query rivolte a Elasticsearch, il database utilizzato, assumono un ruolo centrale nell'elaborazione dei dati e nella creazione di viste personalizzate per gli utenti. Attraverso le funzionalità offerte da Elasticsearch, è possibile effettuare ricerche complesse e ottenere risultati pertinenti in base a criteri specifici, quali tag, lingua o parole chiave contenute nei tweet.

L'adozione di un sistema distribuito per la visualizzazione e l'analisi dei tweet permette agli utenti di accedere a un'ampia varietà di dati e di personalizzare la

loro esperienza secondo le proprie esigenze. La combinazione di un'architettura distribuita, l'utilizzo di strumenti come Kibana per la visualizzazione interattiva dei dati e l'impiego di soluzioni di calcolo, storage e virtualizzazione offerte da Microsoft Azure contribuiscono a garantire la scalabilità, l'affidabilità e la performance del sistema.

In conclusione, la progettazione e l'implementazione di un sistema distribuito per la visualizzazione su mappa di grandi quantità di tweet geolocalizzati, con la possibilità di impostare query per il filtraggio dei dati, rappresenta un'opportunità significativa per l'analisi e l'interpretazione delle attività sui social media.

2.3 Requisiti non funzionali

I requisiti non funzionali rappresentano le caratteristiche del software che non sono esplicitamente richieste dal cliente, ma che hanno un impatto significativo sul lavoro degli sviluppatori e sull'efficienza complessiva del sistema. Essi descrivono come il sistema è in grado di eseguire determinati compiti e quali criteri di prestazione devono essere soddisfatti. Nel contesto di questo progetto, sono stati individuati i seguenti requisiti non funzionali che sono stati presi in considerazione durante lo sviluppo dell'applicazione:

- **Usabilità:** l'obiettivo è stato garantire che gli utenti potessero utilizzare facilmente il software. Sono state adottate best practice di design user-friendly per creare un'interfaccia intuitiva e accessibile. L'obiettivo finale è stato offrire agli utenti un'esperienza piacevole ed efficiente nell'analisi dei dati.
- **Prestazioni:** requisito chiave per il successo del sistema, in quanto influiscono direttamente sull'efficienza e sull'usabilità complessiva. Grazie all'utilizzo di una macchina virtuale su Azure e all'ottimizzazione del software, è stato possibile garantire prestazioni elevate e una piacevole esperienza d'uso per gli utenti del sistema implementato.

- Affidabilità: requisito fondamentale per il software di visualizzazione dei tweet geolocalizzati, poiché garantisce il corretto funzionamento del sistema nel tempo, anche in presenza di eventi imprevisti o dati non conformi. Un software affidabile instaura la fiducia degli utenti e contribuisce a fornire un'esperienza utente soddisfacente e priva di interruzioni indesiderate.
- Robustezza: requisito essenziale per garantire un'esperienza utente affidabile e coerente. Nel caso specifico del progetto, la capacità di gestire correttamente l'inserimento di tweet senza il campo 'coordinate' valorizzato è un aspetto critico da considerare per garantire il corretto funzionamento e l'affidabilità del sistema.

Per quanto attiene all'utilizzo delle risorse fornite da Azure, si è adottato un approccio oculato in termini di costo, cercando di bilanciare adeguatamente le prestazioni con i costi associati. Al fine di ottimizzare l'allocazione delle risorse e garantire un uso efficiente delle stesse, è stata adottata una strategia di limitazione del numero di risorse impiegate.

3. Progettazione

La progettazione rappresenta una fase di cruciale importanza all'interno del ciclo di vita del software nell'ambito dell'ingegneria del software. Durante tale fase, che può altresì essere denominata progetto o disegno, si determina come i requisiti specificati nell'analisi verranno soddisfatti e si delinea con maggiore dettaglio la struttura che il sistema software dovrà assumere.

Nel contesto dello sviluppo del progetto, sono state adottate diverse tecnologie al fine di garantire l'efficace realizzazione del sistema software. Segue una breve ma esauriente descrizione di alcune di queste tecnologie.

3.1 Formato del file e dei dati

Prima di introdurre le tecnologie utilizzate, è opportuno descrivere i dati e i file che li contengono messi a disposizione dal dataset che l'applicazione deve gestire, in quanto è stato necessario apportare delle modifiche per l'utilizzo di questi. I file contenti i tweet presentano un'estensione .json, che indica il formato dei dati contenuti nel file come JSON (JavaScript Object Notation). Tuttavia, al loro interno i dati non sono rappresentati in maniera valida, per questo motivo prima di introdurre delle modifiche ai campi tweet, sono stati validati i file, in modo da risultare in maniera effettiva dei file .json. Si passa adesso all'analisi dei dati, nella *Figure 1* è illustrato un estratto del file di un generico dato, dove sono presenti la proprietà che indica la localizzazione dei tweet, ovvero "coordinates". È possibile notare che il campo di interesse non contiene informazioni; di conseguenza, è stato indispensabile modificare il file.

```
"notifications": null,  
"translator_type": "none"  
},  
"geo": null,  
"coordinates": null,  
"place": null,  
"contributors": null,  
"is_quote_status": false
```

Figure 1

Le modifiche apportate hanno fatto modo che il campo “coordinates” di ogni tweet risultasse come illustrato nella *Figure 2*.

```
"coordinates": {  
    "latitude": 40.529827076792415,  
    "longitude": -92.28349415448648  
},
```

Figure 2

3.2 Lo stack ELK

Lo stack ELK è un insieme di tecnologie open-source ampiamente utilizzate per la gestione dei log e l'analisi dei dati. L'acronimo ELK sta per Elasticsearch, Logstash e Kibana, che sono i componenti principali del sistema. Insieme, Elasticsearch, Logstash e Kibana offrono un potente sistema per l'analisi dei log e l'elaborazione dei dati in tempo reale. Possono essere utilizzati per la gestione dei log, il monitoraggio delle prestazioni, l'analisi di sicurezza, l'analisi dei dati di business e molte altre applicazioni in cui è necessario archiviare, indicizzare, analizzare e visualizzare grandi quantità di dati.

All'interno del progetto sono stati utilizzati solo due dei software presenti nello stack, ovvero:

- **Elasticsearch:** Elasticsearch è un motore di ricerca e analisi distribuito basato su Lucene. È progettato per archiviare, indicizzare e fornire un'interrogazione efficiente su grandi quantità di dati strutturati e non strutturati. Elasticsearch offre una scalabilità orizzontale elevata, una veloce velocità di interrogazione e supporta ricerche complesse, il filtraggio dei dati e l'aggregazione.
- **Kibana:** Kibana rappresenta una soluzione potente e intuitiva all'interno dell'ecosistema ELK, consentendo agli utenti di creare dashboard personalizzate e visualizzazioni interattive per esplorare e analizzare i dati presenti in Elasticsearch. Grazie alle sue funzionalità avanzate, Kibana offre la possibilità di filtrare i tweet in base a diversi criteri come parole chiave,

lingua o tag, consentendo agli utenti di ottenere una visione mirata e dettagliata dei dati. Inoltre, Kibana facilita la navigazione e l'interazione con i tweet visualizzati attraverso la sua interfaccia user-friendly, agevolando così il processo di analisi e l'estrazione di informazioni significative dai dati.

3.3 Microsoft Azure

Microsoft Azure è una piattaforma di cloud computing fornita da Microsoft. È progettata per offrire una vasta gamma di servizi per lo sviluppo, il deployment e la gestione di applicazioni e servizi tramite l'infrastruttura di cloud computing di Microsoft. Azure fornisce un ambiente scalabile e affidabile per ospitare applicazioni web, archiviare e analizzare dati, implementare soluzioni IoT, creare intelligenza artificiale e molto altro. Azure offre oltre 200 servizi, tra cui calcolo, archiviazione, rete, database, analisi, intelligenza artificiale, sicurezza e molto altro ancora.

Tra i principali servizi offerti da Azure, nel progetto se sono stati utilizzati due:

- Azure Virtual Machines (VM): è un servizio cloud che consente di creare e gestire macchine virtuali in un ambiente virtuale. Azure Virtual Machine offre flessibilità e scalabilità nella configurazione delle risorse di calcolo, della memoria e dello spazio di archiviazione, consentendo di adattare le risorse alle esigenze specifiche del progetto.
- Azure Storage: è un servizio di archiviazione di oggetti scalabile e affidabile. Fornisce opzioni di archiviazione duratura e sicura, consentendo di archiviare grandi quantità di dati in modo affidabile e di accedervi in modo efficiente.

3.4 Applicativi Python

Lo sviluppo di applicativi Python, di cui si parlerà dettagliatamente nei paragrafi successivi, è stato necessario per svolgere tutte quelle operazioni intermedie come:

- Download dei dati da Azure;

- Valorizzazione dei dati;
- Creazione di un indice;
- Connessione a Elasticsearch e caricamento dei dati.

4. Architettura

L'architettura del progetto si compone di due componenti interconnesse: un contenitore che ospita i file contenenti i tweet e una macchina virtuale che ospita Elasticsearch e Kibana. Entrambe le componenti sono state create utilizzando i servizi di Azure precedentemente citati.

Sfruttando questi servizi, è stato possibile creare un'architettura robusta e scalabile per il progetto. Azure ha fornito gli strumenti e le risorse necessarie per creare, configurare e gestire le componenti del sistema in modo sicuro e affidabile, consentendo di sfruttare le funzionalità avanzate di Elasticsearch e Kibana e di archiviare i dati in modo efficiente e accessibile.

5. Configurazione

Per rendere possibile l'esecuzione dei vari servizi, è stata necessaria una fase di configurazione delle tecnologie coinvolte. Durante questa fase, è stato fondamentale considerare tutti gli aspetti che possono influire sulla corretta esecuzione dei servizi. In particolare, è stato importante considerare gli aspetti di comunicazione e di sicurezza al fine di garantire una comunicazione sicura tra le diverse parti del sistema.

Assicurare la sicurezza dei dati e prevenire eventuali manipolazioni da parte di utenti malevoli è stata una priorità per l'utente finale. È necessario garantire che i dati utilizzati per le analisi siano affidabili e non siano stati alterati. Inoltre, è essenziale che il sistema soddisfi i requisiti funzionali precedentemente citati.

5.1 Configurazione Azure Storage

La configurazione per creare un contenitore su Azure Storage richiede una serie di passaggi essenziali.

Scelta la sottoscrizione e il numero di risorse, si sono scelte delle prestazioni standard e archiviazione con ridondanza geografica. Si è preferito mantenere le impostazioni predefinite per le fasi di configurazione avanzate, di rete, di prestazione dei dati, di crittografia e dei tag. Infine, è stato creato un contenitore con livello di accesso privato.

Questo processo di configurazione delle tecnologie di Azure, compresa la creazione del contenitore su Azure Storage, è una fase necessaria per garantire il corretto funzionamento del progetto e soddisfare i requisiti funzionali precedentemente definiti.

5.2 Configurazione Azure Virtual Machines (VM)

La configurazione di una macchina virtuale su Azure comporta diversi passaggi fondamentali per creare un ambiente virtuale efficiente e sicuro. Di seguito è riportata una descrizione dettagliata di questa procedura.

La prima fase consiste nell'andare a creare una risorsa nella sezione “Macchine virtuali” per iniziare il processo di creazione di una nuova macchina virtuale. Dopo aver indicato la sottoscrizione e il gruppo di risorse, si è specificato il sistema operativo desiderato, in questo caso Windows, selezionato il gruppo di risorse e scelta l’immagine di base, ovvero Windows 10 Pro versione 22H2- x64 Gen2. La decisione di selezionare questa specifica immagine è stata motivata dalla necessità di garantire la piena compatibilità con gli strumenti utilizzati. Considerando le applicazioni o i carichi di lavoro da eseguire sulla macchina virtuale, l’immagine scelta è stata valutata come la soluzione più adatta per garantire una corretta esecuzione delle operazioni necessarie. Per lo stesso motivo e per motivi economici supplementari, per rientrare nel budget, si è scelta la dimensione Standard_D2s_v3, caratterizzata da: 2 CPU virtuali, 8 GiB di RAM, 4 dischi dati, un numero massimo di 3200 operazioni di I/O al secondo, 16 GiB di archiviazione temporanea. Infine, sono state scelte le seguenti porte in ingresso: RDP (3389), SSH (22), HTTP (80).

Rispetto alle seguenti fasi di configurazione dei dischi, di rete, di gestione, di montaggio, dei tag e qualità avanzate, è stato preferito lasciare le opzioni di default. Per l’accesso alla macchina virtuale è stato impiegato il protocollo RDP (Remote Desktop Protocol) insieme all’applicazione Windows Remote Desktop.

5.3 Installazione e configurazione di Elasticsearch e Kibana

L’implementazione dello stack ELK comprende due fasi principali: l’installazione e la configurazione dei servizi Elasticsearch e Kibana. Durante la configurazione, sono stati utilizzati i file di configurazione elasticsearch.yml e kibana.yml. All’interno di questi file, sono state effettuate diverse impostazioni di rilevanza, tra cui:

1. Creazione di una rete condivisa: È stata definita una rete che permette la comunicazione tra i vari servizi all'interno dello stack ELK. Questo è fondamentale per garantire la corretta interazione tra Elasticsearch e Kibana.
2. Sistema di autenticazione: È stato implementato un sistema di autenticazione per garantire un accesso sicuro ai servizi. Questo sistema richiede l'autenticazione dell'utente prima di consentire l'accesso alle funzionalità di Elasticsearch e Kibana.

Inoltre, sono state fornite le seguenti porte predefinite per l'accesso ai servizi dello stack ELK:

- Porta 5601: Questa porta è dedicata all'accesso a Kibana, che rappresenta il punto di accesso principale per la visualizzazione e l'interazione con la dashboard.
- Porta 9200: Questa porta è riservata a Elasticsearch HTTP, che consente la comunicazione diretta con Elasticsearch attraverso le REST API. È attraverso questa porta che è possibile effettuare richieste e interagire con Elasticsearch per l'indicizzazione, l'interrogazione e la gestione dei dati.

Queste porte consentono agli utenti di accedere e utilizzare in modo appropriato i servizi Elasticsearch e Kibana all'interno dello stack ELK.

6. Implementazione

Dopo aver completato la configurazione dell'ambiente di esecuzione dei servizi, la fase successiva ha comportato l'implementazione pratica dei servizi e delle logiche necessarie per garantire il corretto funzionamento del sistema.

6.1 Validazione file JSON

Per garantire una corretta gestione dei file contenenti oggetti JSON per riga, è stato deciso di effettuare una trasformazione del file originale che conteneva un array di oggetti JSON. Tale decisione è stata presa in quanto, a causa di una validazione non adeguata del contenuto del file, non era possibile eseguirne il download in locale al momento opportuno; a differenza del caricamento, che non ha dato nessun problema. Pertanto, al fine di garantire la corretta validazione e la successiva disponibilità del file, si è optato per la trasformazione dell'array di oggetti JSON. Di seguito, nella *Figure 3*, viene presentato il codice implementato per la modifica del file al fine di garantirne la corretta validazione:

```
1  import json
2
3  def valorizzaFile(input_file_path, output_file_path):
4      array_json = []
5
6  with open(input_file_path, 'r') as input_file:
7      for line in input_file:
8          line = line.strip()
9      if line:
10         try:
11             obj = json.loads(line)
12             array_json.append(obj)
13         except json.JSONDecodeError:
14             print(f"Errore nella decodifica della riga JSON: {line}")
15
16     array_json_str = json.dumps(array_json, indent=4)
17
18 with open(output_file_path, 'w') as output_file:
19     output_file.write(array_json_str)
20
21 print("Array JSON creato con successo.")
22
23 input_file_path = "/Users/carmenlicciardi/Desktop/P_SDCC/pre-processing/tweetRis.json"
24 output_file_path = "/Users/carmenlicciardi/Desktop/drive-download-20230504T134252Z-001/tweetTraformati500.json"
25 valorizzaFile(input_file_path, output_file_path)
```

Figure 3

Dopo aver completato con successo la procedura di validazione del file, quest'ultimo è stato caricato direttamente nel contenitore Azure tramite il sito web utilizzato per la gestione dei servizi.

6.2 Collegamento allo Storage di Azure

Il collegamento al servizio di archiviazione di Azure è stato stabilito mediante l'utilizzo del modulo "Azure.storage.blob" della libreria Python di Azure SDK. Tale modulo offre una serie di funzionalità per interagire con i servizi di archiviazione di tipo Blob di Azure. In particolare, è stata utilizzata la classe BlobServiceClient, che costituisce il client per l'interazione con i servizi di archiviazione di tipo Blob di Azure. L'impiego di questa classe ha consentito di accedere e manipolare in modo efficiente i contenitori e i blob presenti nell'archiviazione di Azure durante l'esecuzione del programma. Di seguito, nella *Figure 4*, viene presentato il codice implementato.

```
✓ from azure.storage.blob import BlobServiceClient
  import json

  STORAGEACCOUNTURL = "https://tweetarchive.blob.core.windows.net/"
  STORAGEACCOUNTKEY = "storageaccountkey"
  FILE_DIRECTORY = r"\Users\azureuser\PycharmProjects\sdcc-Project\fileJSON"
  FILE_NAME = "tweet500.json"
  LOCALFILENAME = FILE_DIRECTORY + "\\" + FILE_NAME
  CONTAINERNAME = "carmen-tweet"
  BLOBNAME = "tweet500.json"

  ✓ try:
    blob_service_client_instance = BlobServiceClient(account_url=STORAGEACCOUNTURL, credential=STORAGEACCOUNTKEY)
    blob_client_instance = blob_service_client_instance.get_blob_client(CONTAINERNAME, BLOBNAME, snapshot=None)

    # Download del file in memoria
    blob_data = blob_client_instance.download_blob()
    data = blob_data.readall()

    # Decodifica i dati in formato JSON
    decoded_data = data.decode("utf-8")

    # Carica i dati come oggetto JSON
    json_data = json.loads(decoded_data)

    # Salvataggio dei dati in formato JSON su un nuovo file locale
    ✓ with open(LOCALFILENAME, "w") as file:
      json.dump(json_data, file)
      print("Dati scaricati e salvati su file locale:", LOCALFILENAME)

  ✓ except Exception as e:
    | print("Si è verificato un errore durante il download dei dati:", str(e))
```

Figure 4

6.3 Normalizzazione coordinate

Analizzando i file, è stato rilevato che la maggior parte dei tweet presentava il campo "coordinates" impostato come null. Per garantire l'accuratezza e la completezza dei dati, è stata effettuata una modifica al campo "coordinates" mediante l'implementazione del seguente codice. Questo codice è stato progettato per assegnare casualmente una coordinata geografica valida a ciascun tweet che si riferisce al territorio degli Stati Uniti d'America.

L'implementazione del codice ha consentito di garantire che tutti i tweet relativi agli Stati Uniti d'America abbiano una coordinata geografica associata, migliorando così la precisione e l'utilità delle informazioni geografiche contenute nel dataset. Di seguito, nella *Figure 5*, viene presentato il codice implementato.

```
1 ~ import json
2   import random
3
4 ~ def get_lat() -> float:
5     v1_lat = 24.396308
6     v2_lat = 49.384358
7     lat = random.uniform(v1_lat, v2_lat)
8     return lat
9
10 ~ def get_lon() -> float:
11    v1_lon = -125.000000
12    v2_lon = -66.934570
13    lon = random.uniform(v1_lon, v2_lon)
14    return lon
15
16 ~ def create_geo_point(lat: float, lon: float) :
17    geo_point = {
18        "lat": lat,
19        "lon": lon
20    }
21    return geo_point
22
23 input_file_path = r"C:\Users\azureuser\PycharmProjects\sdcc-Project\fileJSON\tweet500.json"
24 output_file_path = r"C:\Users\azureuser\PycharmProjects\sdcc-Project\fileJSON\tweetValorizzati5NuoveCoordinate.json"
25
26 # Apertura del file input
27 ~ with open(input_file_path) as json_file:
28     array_json = json.load(json_file)
29
30 # Modifica del campo "coordinates" per ogni oggetto JSON nell'array
31 ~ for obj in array_json:
32     # Verifica se il campo "coordinates" esiste nell'oggetto e se è un dizionario
33 ~     if obj.get('coordinates') is not None:
34         print("Coordinates are populated")
35 ~     else:
36         lat= get_lat()
37         lon= get_lon()
38         obj["coordinates"] = create_geo_point(lat,lon)
39
40
41 ~ with open(output_file_path, "w") as tweetValorizzati:
42     json.dump(array_json, tweetValorizzati, indent=4)
43
44 print("File creato e modificato con successo.")
```

Figure 5

6.4 Connessione ad Elasticsearch

La connessione al servizio Elasticsearch è stata stabilita attraverso l'utilizzo del modulo "elasticsearch" e la classe "Elasticsearch". All'interno del metodo, è stato implementato del codice aggiuntivo per verificare la disponibilità della macchina su cui Elasticsearch è in esecuzione.

Questa implementazione è stata realizzata al fine di garantire una gestione adeguata delle connessioni e fornire un feedback tempestivo sull'accessibilità del servizio. Utilizzando il codice aggiuntivo, è stato possibile effettuare una verifica preliminare per determinare se la macchina remota è raggiungibile o meno prima di avviare ulteriori operazioni. Ciò contribuisce a migliorare l'affidabilità e la resilienza dell'applicazione nel caso in cui la macchina remota non sia accessibile o risponda con errori. Di seguito, nella *Figure 6*, viene presentato il codice implementato.

```
5  def conn_es():
6
7      # Crea una connessione a Elasticsearch
8      es = Elasticsearch(
9          hosts=['http://localhost:9200'],
10         basic_auth=('elastic', 'password')
11     )
12
13     # Verifica la connessione
14     if es.ping():
15         print("Connessione a Elasticsearch riuscita.")
16     else:
17         print("Connessione a Elasticsearch non riuscita.")
18
19     return es
```

Figure 6

6.5 Definizione indice

La definizione di un indice in Elasticsearch svolge un ruolo fondamentale nell'organizzazione e nella strutturazione efficiente dei dati. Analogamente a una tabella in un database relazionale, un indice in Elasticsearch costituisce un contenitore logico per i documenti correlati. La sua creazione e specifica risultano

essenziali per diverse ragioni. In primo luogo, l'indice stabilisce la struttura dei dati che saranno archiviati in Elasticsearch. Definisce i campi e i relativi tipi di dati associati, consentendo a Elasticsearch di comprendere come interpretare e indicizzare correttamente i documenti. In secondo luogo, l'indice agevola l'indicizzazione e la ricerca dei dati. Attraverso l'utilizzo dell'indice, Elasticsearch crea un indice invertito che permette un recupero rapido dei documenti in base a specifici criteri di ricerca. L'assenza di un indice adeguato comporterebbe un'inefficienza e una scarsa performance nell'esecuzione delle ricerche. Inoltre, l'indice consente di configurare analisi dei dati specifiche per i campi dei documenti. Questo implica la suddivisione dei testi in termini e token, l'applicazione di filtri per la normalizzazione dei dati, l'implementazione di sinonimi e altre operazioni analoghe. Tali analisi contribuiscono a migliorare la precisione e la rilevanza delle ricerche effettuate sui dati. Infine, l'indice permette di gestire le autorizzazioni e le politiche di accesso ai dati. Attraverso la definizione di regole di sicurezza a livello di indice, è possibile controllare i privilegi di lettura, scrittura e modifica dei documenti all'interno dell'indice.

In poche parole, la definizione di un indice in Elasticsearch rappresenta un elemento cruciale per organizzare, indicizzare e rendere accessibili in modo efficiente i dati, garantendo ricerche veloci e precise. Nel contesto attuale, il mapping definito per il campo "coordinates" come tipo "geo_point" consente al front-end (Kibana) di visualizzare in modo accurato i tweet geolocalizzati. Grazie a questa configurazione, i dati delle coordinate geografiche presenti nei tweet possono essere correttamente interpretati e rappresentati nell'interfaccia di Kibana, consentendo agli utenti di effettuare analisi e visualizzazioni basate sulla posizione geografica dei tweet. Inoltre, nel codice viene effettuato un controllo sull'esito dell'operazione. Se la creazione dell'indice è stata confermata con successo, viene stampato un messaggio di conferma. In caso contrario, viene stampato un messaggio di errore insieme alla risposta restituita da Elasticsearch. Di seguito, nella *Figure 7*, viene presentato il codice implementato.

```

4  def create_index(index_name):
5
6      # Definisci il mapping per i campi del tweet
7      setting = {
8          "settings": {
9              "index.mapping.total_fields.limit": 3000,
10         },
11         "mappings": {
12             "properties": {
13                 "coordinates": {
14                     "type": "geo_point"
15                 },
16                 "created_at": {
17                     "type": "date",
18                     "format": "EEE MMM dd HH:mm:ss Z yyyy"
19                 }
20             }
21         }
22     }
23
24     # Crea l'indice con il mapping specificato
25     response = es.indices.create(index=index_name, body=setting)
26
27     # Controlla se l'operazione è andata a buon fine
28     if response["acknowledged"]:
29         print("Indice creato con successo!")
30     else:
31         print("Errore durante la creazione dell'indice:", response)

```

Figure 7

6.6 Caricamento dati

Per il caricamento dei tweet, è stata eseguita un'analisi del file che contiene l'array di oggetti JSON. Successivamente, ogni singolo oggetto JSON è stato inviato individualmente per essere indicizzato. Di seguito, nella *Figure 8*, viene presentato il codice implementato.

```
64 ~ with open(file_path, 'r') as file:  
65 |     # Carica il contenuto del file come stringa JSON  
66 |     json_string = file.read()  
67 |     print("Contenuto caricato")  
68 |  
69 |     # Converti la stringa JSON in un oggetto Python  
70 |     json_data = json.loads(json_string)  
71 |     print("Stringa convertita")  
72 |  
73 |     # Itera sugli oggetti JSON uno per uno  
74 |     for oggetto in json_data:  
75 |         response = es.index(index=index_name, document=oggetto)  
76 |  
77 |
```

Figure 8

7. Visualizzazione e analisi dei tweet

Per la visualizzazione dei tweet e l'esecuzione delle query, è stata adottata l'utilizzo di Kibana. La dashboard realizzata offre un'interfaccia intuitiva e completa per la visualizzazione e l'analisi dei tweet geolocalizzati. Attraverso la combinazione di grafici, mappe interattive e filtri, gli utenti possono esplorare i dati in modo efficace e ottenere una comprensione approfondita delle tendenze e dei modelli presenti nei tweet. La dashboard fornisce una panoramica immediata delle informazioni chiave, consentendo agli utenti di identificare rapidamente le aree geografiche più attive, i trending topic e le lingue predominanti. Inoltre, grazie alla possibilità di impostare query personalizzate, gli utenti possono filtrare i tweet in base a specifici criteri come hashtag o intervallo di tempo, consentendo un'analisi ancora più precisa e mirata. La dashboard rappresenta uno strumento indispensabile per gli utenti che desiderano ottenere insights significativi dai tweet geolocalizzati e prendere decisioni basate sui dati.

La dashboard implementata è composta da diversi componenti che lavorano sinergicamente per fornire un'esperienza utente completa e intuitiva: grafici interattivi, mappa interattiva, filtri, contatori e metriche.

7.1 Grafici interattivi

All'interno della dashboard sono presenti diversi grafici interattivi che offrono una visualizzazione dettagliata dei dati relativi ai tweet geolocalizzati. I grafici inclusi sono i seguenti:

1. Grafico a linee degli utenti più apprezzati (*Figure 9*): Questo rappresenta la media dei like ricevuti dagli utenti nel tempo. Permette di identificare gli utenti che generano maggiori interazioni e coinvolgimento da parte degli altri utenti.

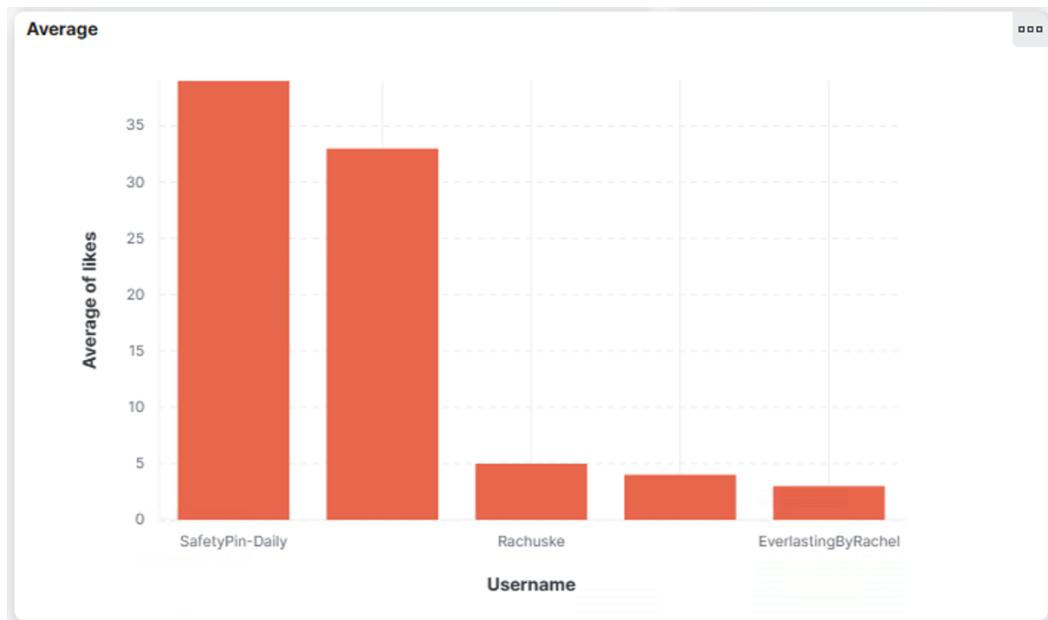


Figure 9

2. Grafico a linee degli hashtag più utilizzati (*Figure 10*): Questo visualizza l'andamento degli hashtag più utilizzati nei tweet geolocalizzati. Permette di individuare le tendenze e le tematiche più rilevanti e popolari all'interno del dataset.

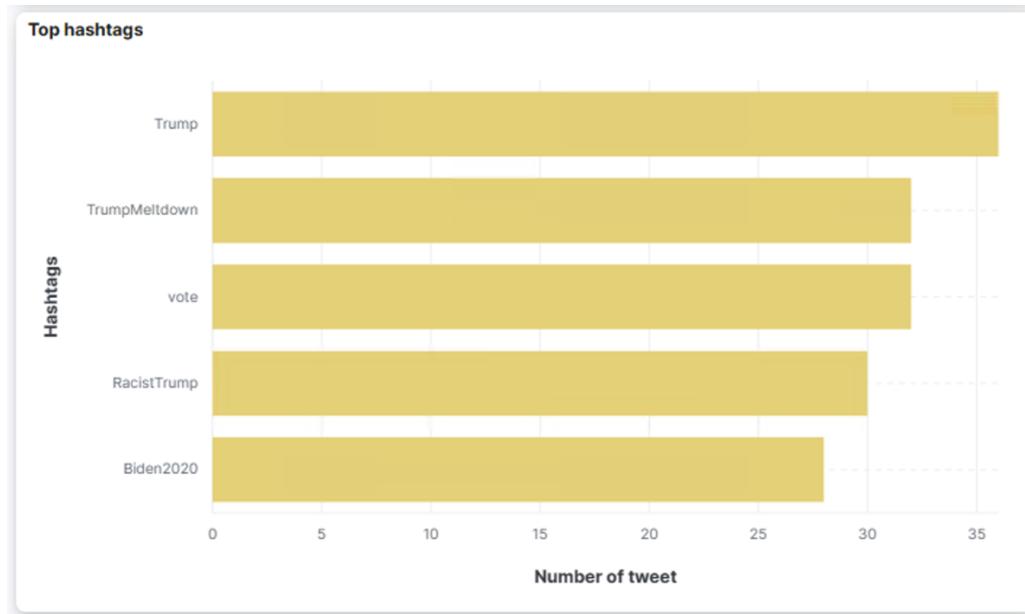


Figure 10

3. Grafico a torta dei media più citati: questo mostra la distribuzione percentuale dei media (ad esempio, giornali, siti web, blog) che vengono citati maggiormente nei tweet. Permette di comprendere quali fonti di

informazione sono più frequentemente menzionate dagli utenti.

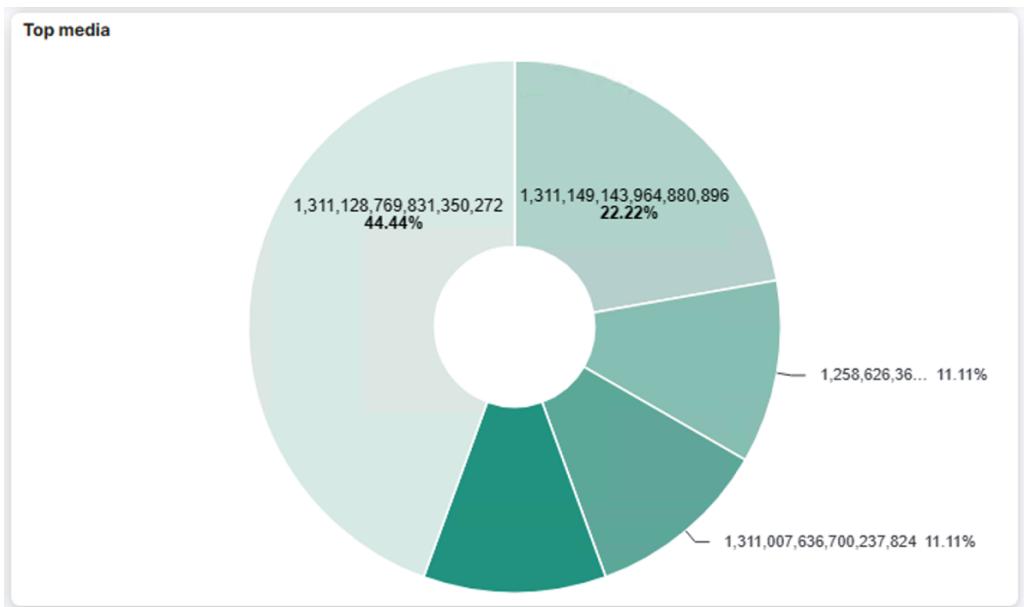


Figure 11

4. Grafico del numero di tweet per luogo (*Figure 12*): rappresenta la distribuzione geografica dei tweet in base al numero di post inviati in ogni luogo specifico. Fornisce una panoramica visiva dei luoghi in cui si concentrano maggiormente le attività di tweet.

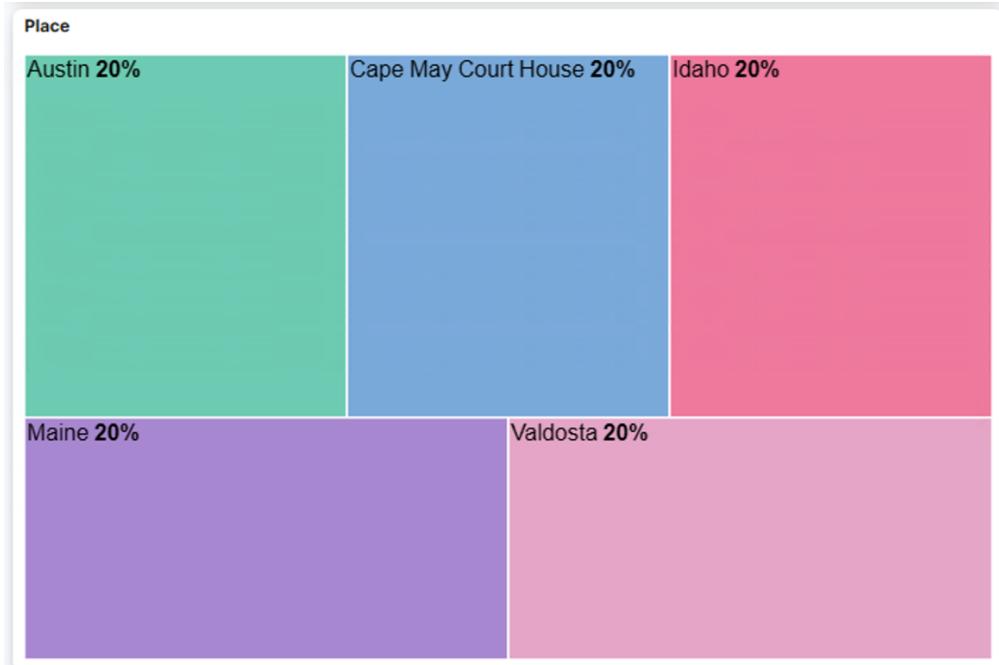


Figure 12

5. Grafico delle lingue più utilizzate (*Figure 13*): questo mostra le lingue utilizzate nei tweet. Permette di identificare le lingue predominanti

all'interno del dataset e comprendere la diversità linguistica dei tweet geolocalizzati.

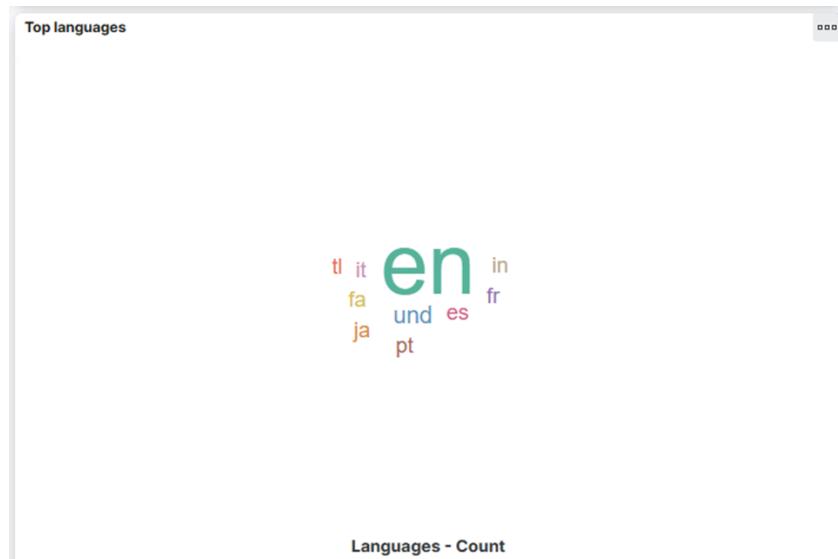


Figure 13

- Mappa interattiva (*Figure 14*): è un elemento fondamentale della dashboard e visualizza la posizione geografica dei tweet. Gli utenti possono zoomare, trascinare e interagire con la mappa per esplorare le diverse aree geografiche e ottenere una visualizzazione chiara della distribuzione spaziale dei tweet. In particolare, è stata implementata una funzionalità interattiva, la quale è possibile visualizzare nella *Figure 15*, che consente agli utenti di ottenere informazioni dettagliate sui singoli tweet semplicemente posizionando il cursore su di essi. Questa interazione permette di visualizzare: data di creazione del tweet, username dell'utente, testo completo del tweet e hashtags utilizzati.

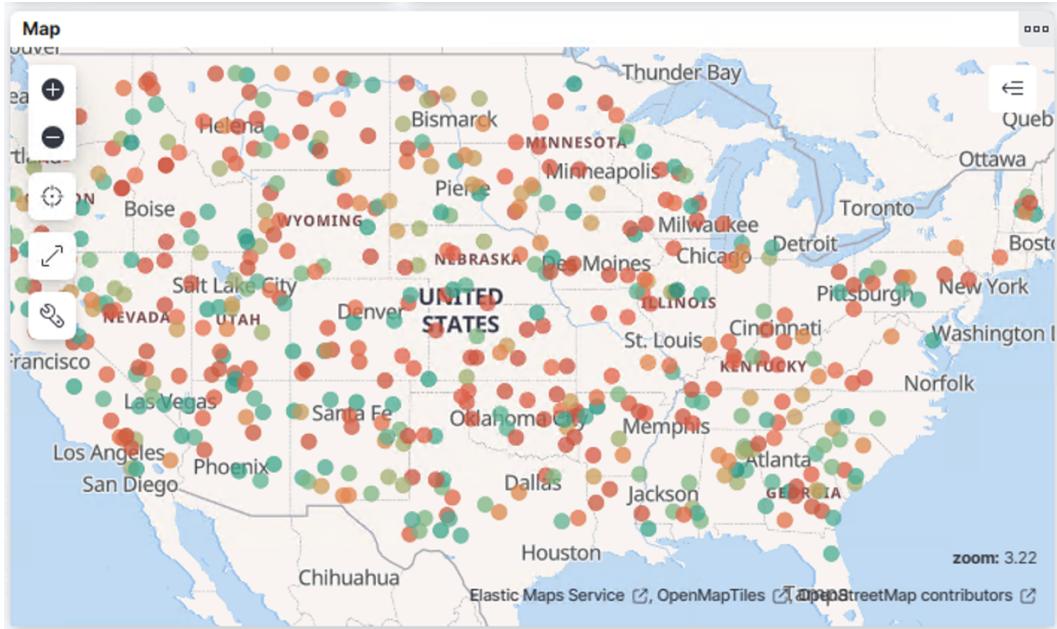


Figure 14

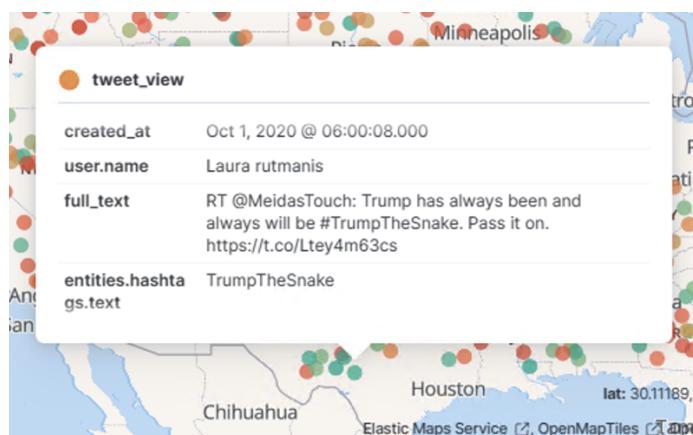


Figure 15

7.2 Filtri

I filtri che consentono agli utenti di selezionare e filtrare i dati dei tweet in base a criteri specifici. Questi filtri offrono un modo flessibile per personalizzare la visualizzazione dei tweet e ottenere risultati più rilevanti per le analisi desiderate. Gli utenti possono utilizzare un filtro per selezionare i tweet in base agli hashtags utilizzati (*Figure 16*), consentendo di focalizzarsi su specifici argomenti o temi di interesse. Inoltre, è possibile filtrare i tweet in base alla lingua utilizzata (*Figure 17*), consentendo agli utenti di concentrarsi su tweet scritti in una lingua specifica. Un filtro consente anche di selezionare i tweet che includono specifiche menzioni

(Figure 18), permettendo agli utenti di esplorare le interazioni tra gli utenti e identificare i tweet che coinvolgono persone o account specifici. Inoltre, è possibile filtrare i tweet in base alla loro sensibilità (Figure 19), consentendo agli utenti di scegliere se visualizzare solo tweet considerati non sensibili o includere anche quelli che potrebbero contenere contenuti sensibili. Un altro filtro disponibile include la selezione del periodo temporale dei tweet (Figure 20), consentendo agli utenti di specificare un intervallo di tempo specifico per analizzare i dati. Inoltre, è possibile filtrare i tweet in base alla presenza di un determinato media citato (Figure 21), consentendo agli utenti di esplorare i tweet che includono un collegamento a un media specifico, come immagini o video.

Infine, un filtro permette anche di selezionare i tweet basati sulla posizione specifica (Figure 22), consentendo agli utenti di esplorare i tweet relativi a un'area geografica specifica o a un luogo specifico di interesse.

L'utilizzo di questi filtri offre agli utenti un maggiore controllo sulla visualizzazione e l'analisi dei dati dei tweet, permettendo loro di concentrarsi sui tweet rilevanti e ottenere risultati più precisi e significativi per le loro esigenze di analisi.



Figure 16



Figure 17

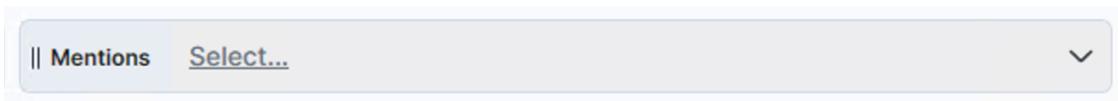


Figure 18

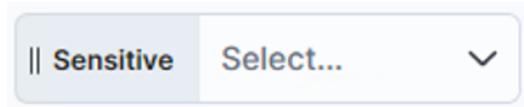


Figure 19

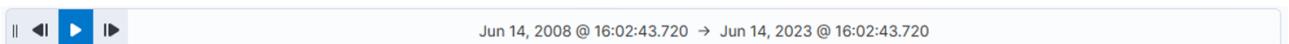


Figure 20

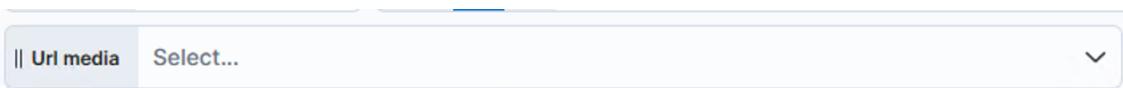


Figure 21



Figure 22

7.3 Contatori e metriche

Sono state utilizzate le metriche e i contatori, osservabili nella *Figure 23* e nella *Figure 24*, al fine di rappresentare in maniera accurata il numero di tweet e il numero di retweet. Inoltre, sono state sfruttate le metriche per consentire la selezione del range dei like dei tweet e il conteggio dei follower degli utenti. Tale approccio ha permesso di ottenere una visualizzazione chiara e dettagliata delle informazioni desiderate, consentendo un'analisi precisa e approfondita dei dati.

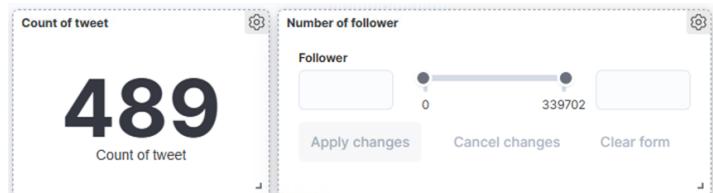


Figure 23



Figure 24

8. Conclusioni

In conclusione, attraverso l'analisi dei requisiti funzionali e non funzionali, è stato possibile progettare e implementare una dashboard interattiva che offre agli utenti la possibilità di esplorare i tweet in modo efficiente e intuitivo. La dashboard presenta una varietà di grafici interattivi, consentendo agli utenti di visualizzare le tendenze degli utenti più popolari, gli hashtag più utilizzati, i media più citati, i luoghi di tweet più frequenti e le lingue più utilizzate.

L'utilizzo dei filtri consente agli utenti di selezionare e filtrare i dati dei tweet in base a criteri specifici, migliorando la precisione e la pertinenza dei risultati delle analisi. I filtri consentono di concentrarsi su argomenti, lingue, menzioni, sensibilità dei tweet, periodi temporali, citazioni di media e luoghi specifici. Ciò offre agli utenti un'esperienza di visualizzazione dei dati più personalizzata e mirata alle loro esigenze.

La piattaforma sviluppata si basa sulla potenza dei servizi di calcolo, storage e virtualizzazione forniti da Microsoft Azure, garantendo una solida infrastruttura per la gestione e l'elaborazione dei dati.

La demo funzionante dell'applicazione ha dimostrato la sua capacità di gestire grandi moli di dati di tweet geolocalizzati, consentendo agli utenti di esplorare e analizzare i dati in modo efficace. L'implementazione di un'interfaccia utente intuitiva e l'uso di strumenti potenti come Kibana hanno contribuito a semplificare il processo decisionale degli utenti finali e a fornire una visualizzazione chiara e significativa delle informazioni.

In altre parole, il progetto ha raggiunto l'obiettivo di creare un sistema distribuito per la visualizzazione dei tweet su mappa, fornendo un'interfaccia interattiva e potente per l'analisi dei dati. L'implementazione di Kibana ed Elasticsearch ha dimostrato di essere una combinazione vincente per la gestione e la visualizzazione dei dati in modo efficace.