Specifica Architetturale

Informazioni sul Documento	
Versione	1.0.0
Approvatori	
Redattori	Francesco Protopapa Greta Cavedon Matteo Basso
Verificatori	Michele Gatto
Uso	Interno
Distribuzione	Prof. Vardanega Tullio Prof. Cardin Riccardo Gruppo *DreamTeam*

e-mail: dreamteam.unipd@gmail.com

Registro delle Modifiche

Versione	Data	Nominativo	Ruolo	Descrizione
v0.1.0	2022-05-07	Francesco Protopapa	Progettista	Verifica complessiva di coesione e consistenza (Verificatore: *Michele Gatto*)
v0.0.4	2022-05-05	Matteo Basso	Progettista	Stesura §2.3; (Verificatore: *Michele Gatto*)
v0.0.3	2022-05-05	Francesco Protopapa	Progettista	Stesura §2.2; (Verificatore: *Michele Gatto*)
v0.0.2	2022-05-04	Francesco Protopapa	Progettista	Stesura §2.1; (Verificatore: *Michele Gatto*)
v0.0.1	2022-04-30	Francesco Protopapa	Amministratore	Creazione scheletro documento e stesura §1; (Verificatore: *Michele Gatto*)

Indice

Elenco delle figure

1 Introduzione

1.1 Scopo del Documento

Lo scopo del presente documento è quello di descrivere in maniera coesa, coerente ed esaustiva le caratteristiche architetturali del prodotto *Sweeat* sviluppato dal gruppo *DreamTeam*.

1.2 Scopo del Prodotto

L'obiettivo di Sweeat e dell'azienda Zero12 è la creazione di un sistema software costituito da una Webapp. Lo scopo del prodotto è di fornire all'utente una guida dei locali gastronomici sfruttando i numerosi contenuti digitali creati dagli utenti sulle principali piattaforme social (Instagram e TikTok). In questo modo, è possibile realizzare una classifica basata sulle impressioni e reazioni di chiunque usufruisca dei servizi dei locali, non solo da professionisti ed esperti del settore.

1.3 Glossario

Per evitare ambiguità relative alle terminologie utilizzate è stato creato un documento denominato "*Glossario*". Questo documento comprende tutti i termini tecnici scelti dai membri del gruppo e utilizzati nei vari documenti con le relative definizioni. Tutti i termini inclusi in questo glossario, vengono segnalati all'interno del documento con l'apice ^{G} accanto alla parola.

1.4 Riferimenti

1.4.1 Normativi

1.4.2 Informativi

- Regolamento del progetto didattico - Materiale didattico del corso di Ingegneria del Software: https://www.math.unipd.it/~tullio/IS-1/2021/Dispense/PD2.pdf.
- Model-View Patterns - Materiale didattico del corso di Ingegneria del Software: https://www.math.unipd.it/~rcardin/sweb/2022/L02.pdf

2 Architettura del Prodotto

2.1 Architettura generale

2.1.1 Schema

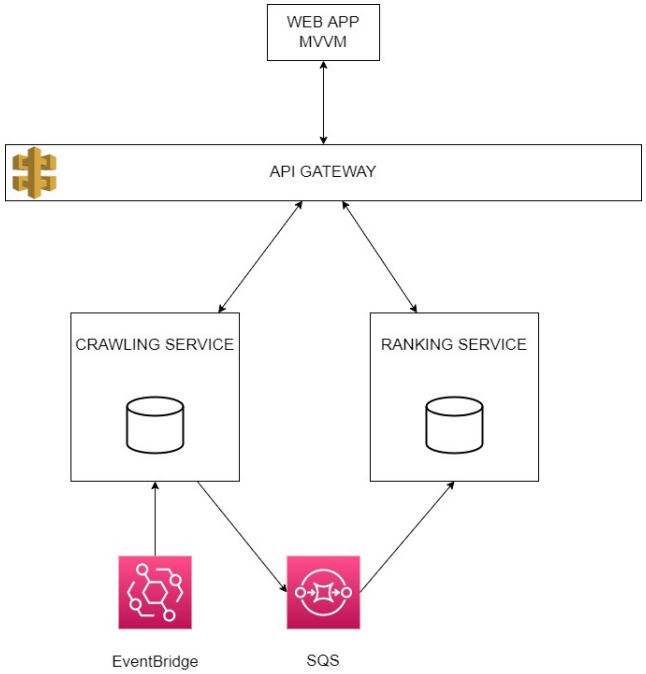

Figura 1: Architettura generale

2.1.2 Descrizione

Come richiesto dal capitolato si è deciso di utilizzare un'architettura a microservizi, i quali comunicano con il frontend tramite API gateway. In particolare sono stati individuati i seguenti microservizi:

- *Crawling Service*: questo microservizio si occupa di tutto ciò che riguarda il crawling dei dati da instagram. Il processo di crawling viene innescato da un servizio di AWS chiamato EventBridge che si occupa dello scheduling del crawling. Ogni volte che viene trovato dal crawler un post relativo ad un ristorante, questo viene inviato ad una coda di tipo SQS dalla quale andrà a leggere il servizio di ranking. Infine il Crawling Service espone una API al frontend per permettere di suggerire profili instagram da aggiungere alla lista di quelli osservati dal crawler.
- *Ranking Service*: questo microservizio invece si occupa dell'analisi dei contenuti estratti dal crawler e della realizzazione di una classifica di ristoranti. Il processo di analisi di un post viene fatto partire dalla ricezione di un messaggio sulla coda SQS, una volta letto il messaggio esso viene rimosso dalla coda ed analizzato. Infine il Ranking Service espone molteplici API al frontend in grado di fornire tutte le informazioni necessarie per poter visualizzare la classifica, i dettagli di un locale e la gestione dei preferiti.

Il frontend infine adotta un pattern MVVM e comunica col backend esclusivamente tramite API gateway. Grazie al servizio di AWS chiamato Cognito, tutta la parte relativa all'autenticazione è gestita lato frontend.

2.2 Architettura del Crawling Service

2.2.1 Descrizione

Il microservizio denominato *Crawling Service* si occupa principalmente di due funzionalità:

- Crawling dei dati: questo processo avviene tramite l'utilizzo della libreria Instagrapi. Ogni volta che viene chiamato il metodo start_crawling della classe FacadeCrawling, il sistema sceglie dal proprio database il profilo instagram che non viene guardato da più tempo e ne effettua il crawling dei dati. Per ogni media prelevato dal profilo, viene analizzata la sua location e nel caso in cui essa sia quella di un ristorante, il media viene salvato nel database ed inviato alla coda SQS in modo che possa essere analizzato dal servizio *Ranking Service*;
- Suggerimento dei profili: questa funzionalità viene esposta tramite una API rest al frontend al fine di fornire all'utente la possibilità di suggerire un profilo instagram sul quale successivamente il sistema andrà ad effettuare il crawling dei dati. Prima di essere aggiunto al database, viene controllato che il profilo non sia già presente, esista e sia pubblico. In base a ciascun esito verrà restituito il corrispettivo ritorno.

2.2.2 Diagrammi delle classi

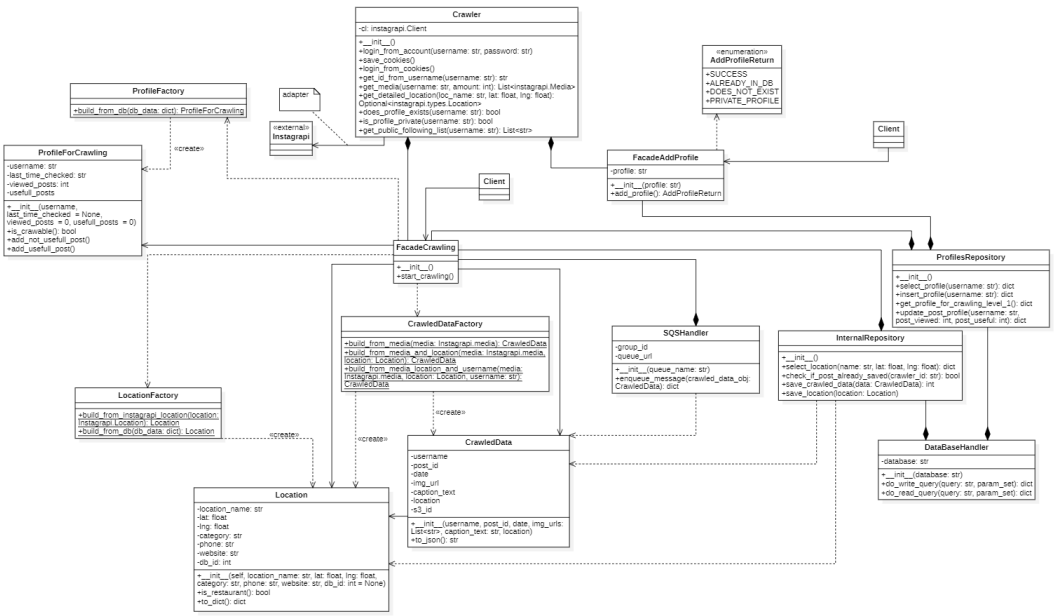

Figura 2: Crawling Service - Diagramma delle classi

2.2.3 Diagrammi di sequenza

In questa sezione vengono presentati i diagrammi di sequenza che modellano le operazioni principali del Crawling Service:

- il suggerimento di un profilo instagram da aggiungere alla lista dei profili su cui viene effettuato il crawling dei dati, nel caso in cui il profilo non sia già presente e sia pubblico;
- il processo di crawling dei dati;
- la formattazione di un singolo media ottenuto tramite crawling

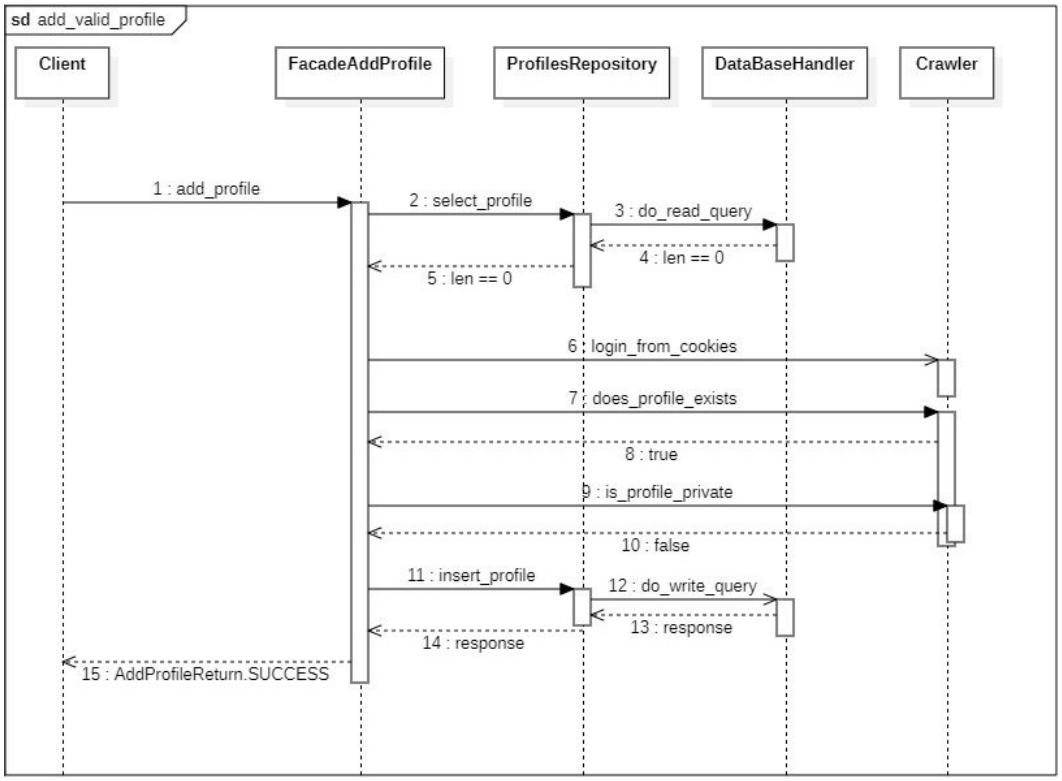

Figura 3: Crawling Service - Diagramma di sequenza - 1

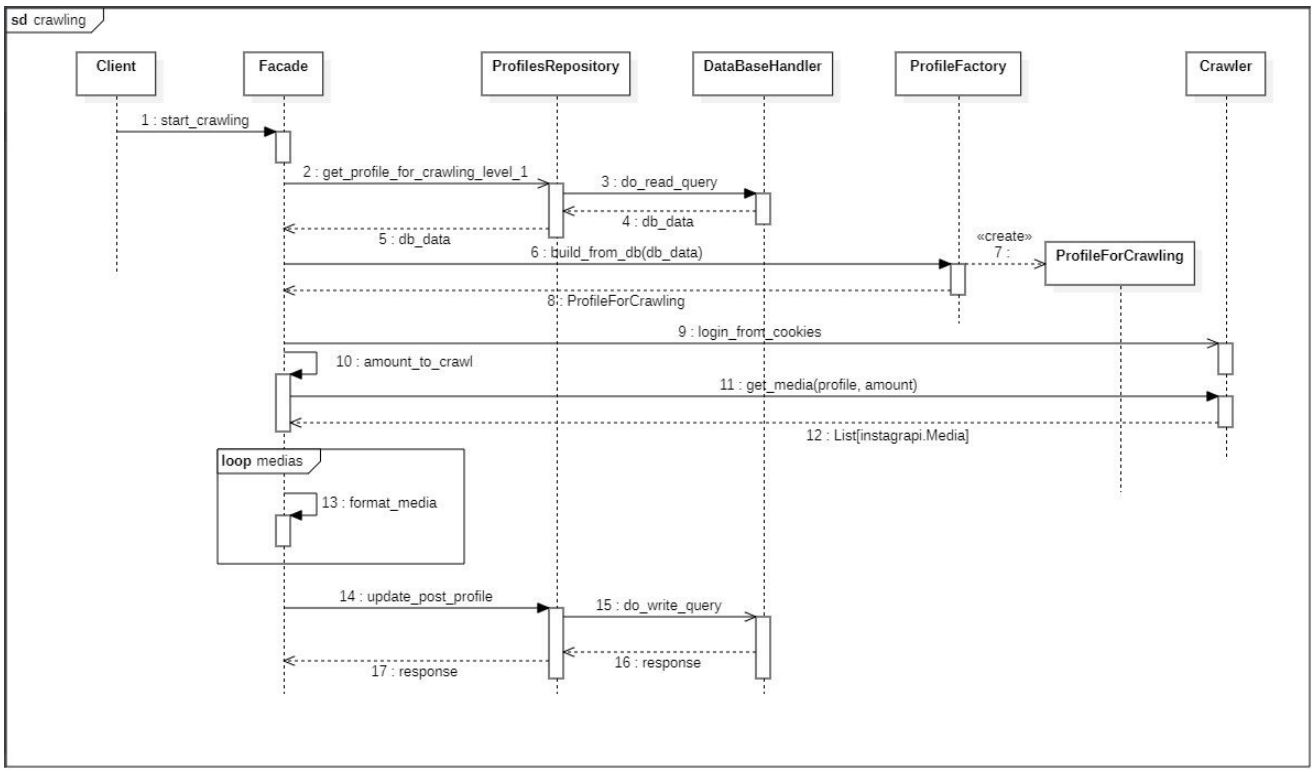

Figura 4: Crawling Service - Diagramma di sequenza - 2

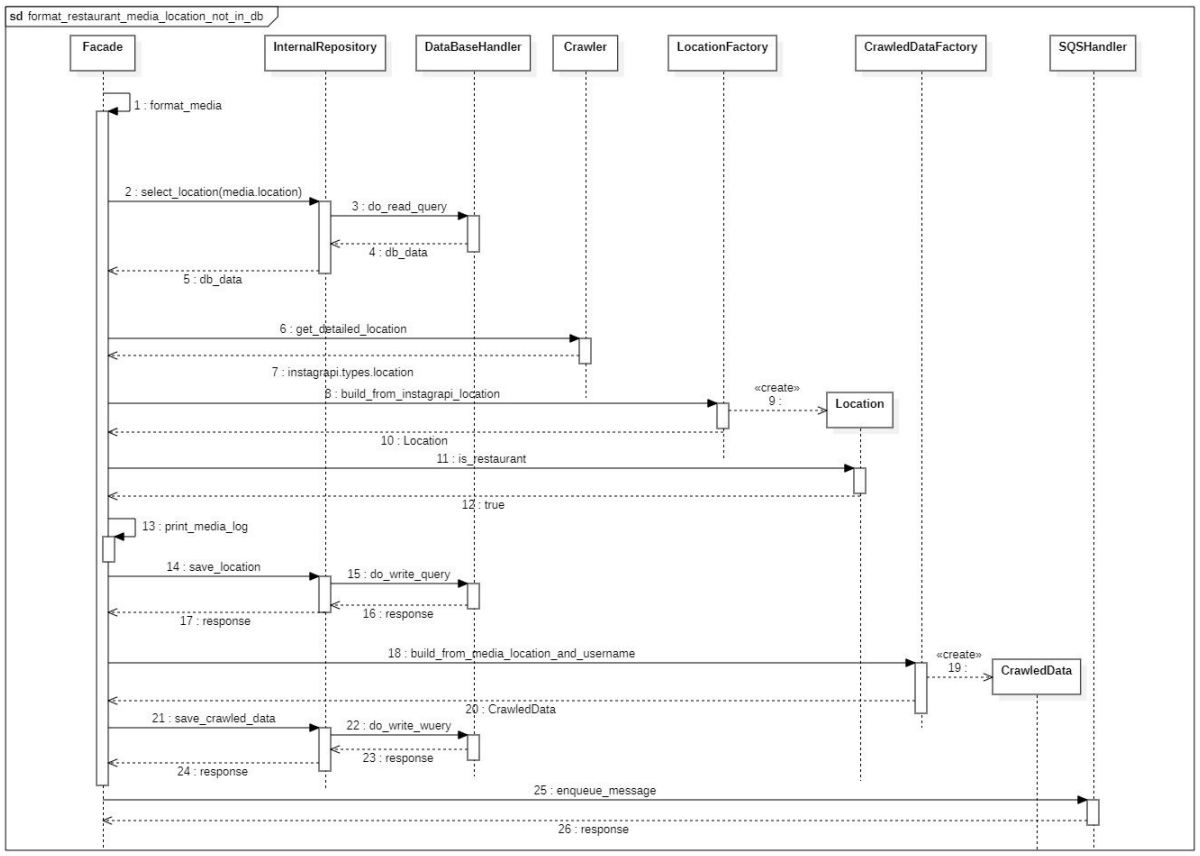

Figura 5: Crawling Service - Diagramma di sequenza - 3

2.2.4 Struttura messaggio SQS


```
{
  "messageId": "19dd0b57-b21e-4ac1-bd88-01bbb068cb78",
  "receiptHandle": "MessageReceiptHandle",
  "body": {
    "username": "lorenzolinguini",
    "post_id": "2806725869999084478_241913061",
    "date": "2022-04-01 10:10:18",
    "img_url": [
      44,
      123
    ],
    "caption_text": "“Ask The Angels”👼\n#PattiSmith 🇺🇸🎙\n:\nRiassunto langarolo
      (potenziato) in sfoglia d’agnolotti al sugo d’arrosto🥟🏠🤍🟦\nPoi Russa; Tonnato,
      Cruda & ogni sorta di beatitudine territoriale - confezionata al cesello - tra i
      capisaldi de @lapiolaalba 🥩🍷🌰\nPerché la tradizione, realizzata a queste altezze
      ascetiche, non riserva mai ‘pesci d’Aprile’🙏🤍🤖🟦\nBuon Pranzo & Buon WeekEnd
      Splendori♥",
    "location": {
      "location_name": "La Piola Alba",
      "lat": 44.7005,
      "lng": 8.036,
      "category": "Piedmont Restaurant",
      "phone": "0173442800",
      "website": "http://www.lapiola-alba.it/",
      "db_id": 8
    }
  },
  "attributes": {
    "ApproximateReceiveCount": "1",
    "SentTimestamp": "1523232000000",
    "SenderId": "123456789012",
    "ApproximateFirstReceiveTimestamp": "1523232000001"
  },
  "messageAttributes": {},
  "md5OfBody": "{{{md5_of_body}}}",
  "eventSource": "aws:sqs",
  "eventSourceARN": "arn:aws:sqs:us-east-1:123456789012:MyQueue",
  "awsRegion": "us-east-1"
}
```


Figura 6: Crawling Service - Esempio di un messaggio SQS

2.2.5 Design pattern notevoli utilizzati

Per La realizzazione del Crawling Service sono stati utilizzati i seguenti design pattern:

- **Facade:** Utilizzato per la realizzazione delle classi FacadeCrawling e FacadeAddProfile, in modo da fornire ai client un'interfaccia semplice ad un sottosistema molto complesso e disaccoppiando la logica di implementazione del sistema dal client.
- **Adapter:** Utilizzato dalla classe Crawler pre disaccoppiare il resto del sistema dai metodi di instagrapi, rendendo disponibili solo quelli necessari tramite un'interfaccia nota al sistema.
- **Static Factory:** Utilizzato per fornire dei metodi statici in grado di creare oggetti di tipo CrawledData, Location, ProfileForCrawling a partire da altri tipi di oggetti.

2.2.6 Schema del database

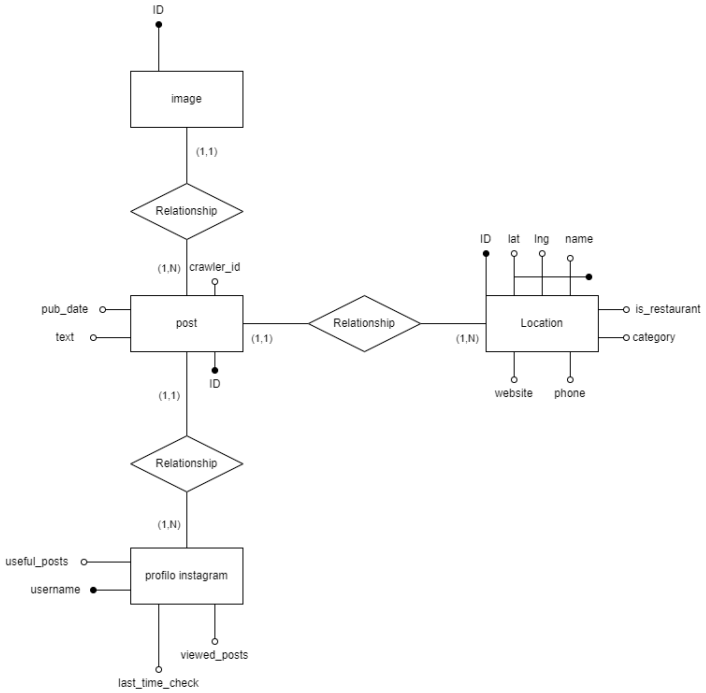

Figura 7: Crawling Service - Schema ER del database

2.3 Architettura del Ranking Service

2.3.1 Descrizione

Il microservizio denominato *Ranking Service* svolge le seguenti funzionalità:

- Gestione della classifica: ogni volta che un nuovo messaggio (struttura?) viene aggiunto alla coda SQS, viene chiamati il metodo `refresh_ranking` che si occupa di recuperare e fare la parsificazione e deserializzazione dei messaggi nella coda (lavorando a batch di massimo 10 messaggi). Dopo di che i messaggi vengono analizzati tramite i servizi Comprehend e Rekognition di AWS e vengono generati i punteggi per le foto, per il testo e per le emoji. Viene quindi aggiornato il database, aggiungendo il ristorante e il media (se non presenti) e aggiornati i punteggi. Vengono inoltre esposti due API endpoint:
  - `getRanking`: restituisce una parte della classifica generale;
  - `getLabelAndPost`: restituisce i media (e le relative labels) di un particolare ristorante.
- Funzionalità di ricerca: viene esposto un API endpoint `searchByName` che restituisce tutte le informazioni presenti nel database relative ad un ristorante con un nome specifico (media compresi);
- Gestione dei preferiti: viene esposto un API endpoint `favorites` che abilità alla gestione della lista di ristoranti preferiti per ogni utente, fornendo le funzionalità di aggiunta, rimozione e visualizzazione per la lista dei preferiti.

2.3.2 Diagrammi delle classi

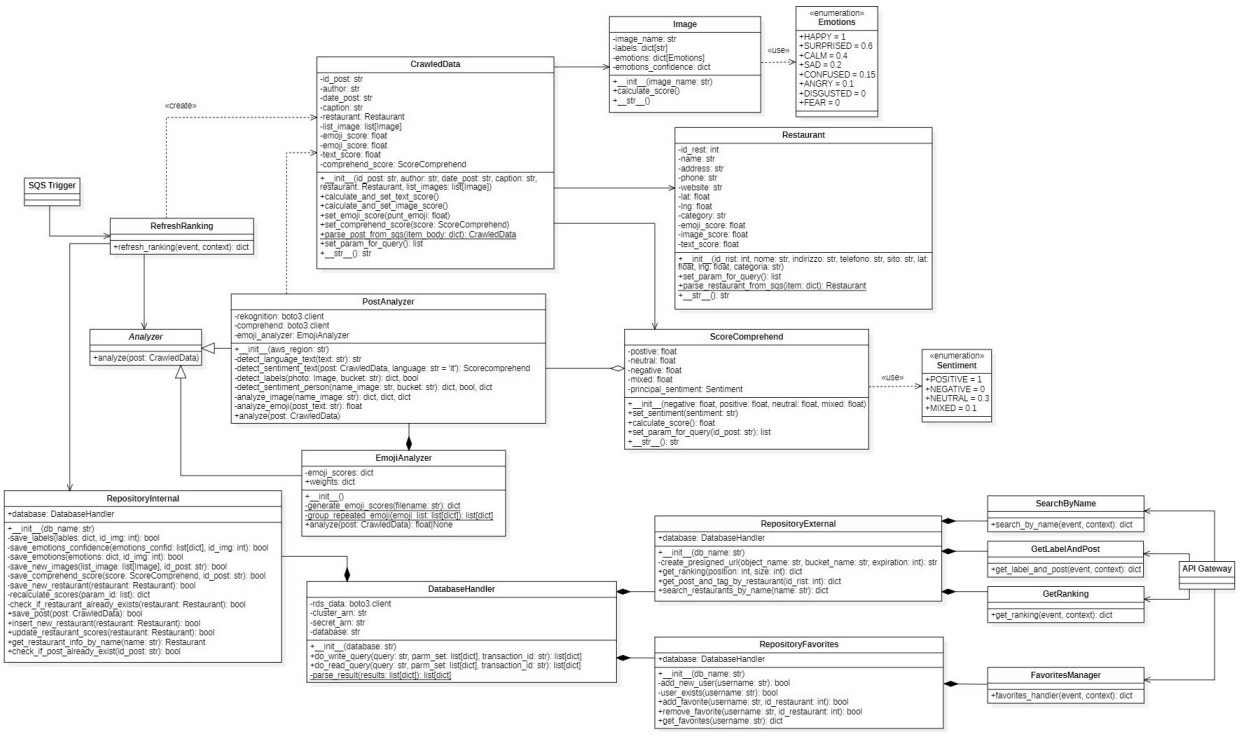

Figura 8: Ranking Service - Diagramma delle classi

2.3.3 Diagrammi di sequenza

In questa sezione vengono presentati i diagrammi di sequenza che modellano le operazioni principali del Ranking Service:

- Il processo di analisi di un media, assumendo che il media e il ristorante non siano già presenti nel database e che siano presenti persone delle immagini;

- Il processo di salvataggio del media analizzato e del ristorante, sempre assumendo che il media e il ristorante non siano già presenti nel database, e l'aggiornamento dei punteggi.

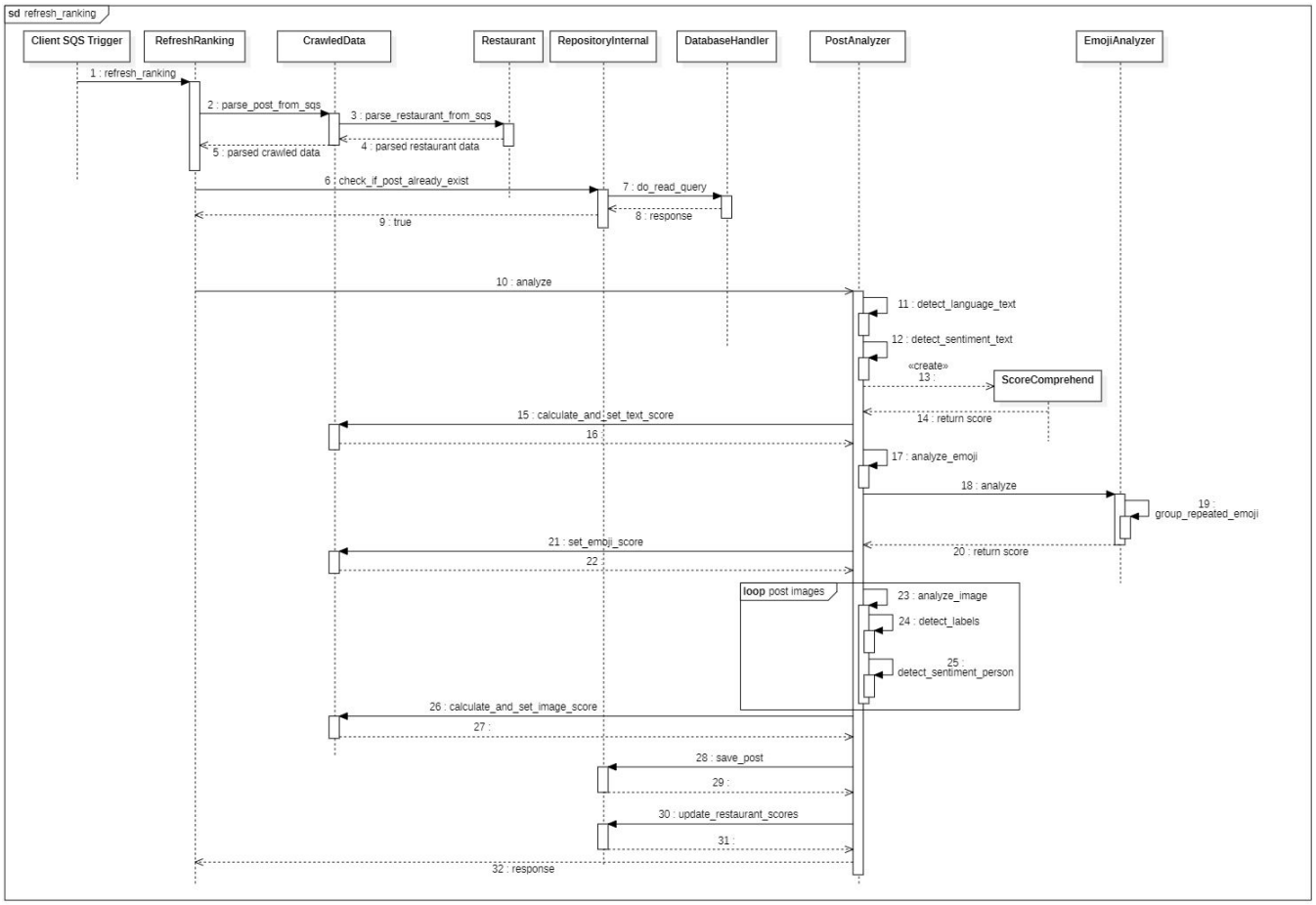

Figura 9: Ranking Service - Diagramma di sequenza - 1

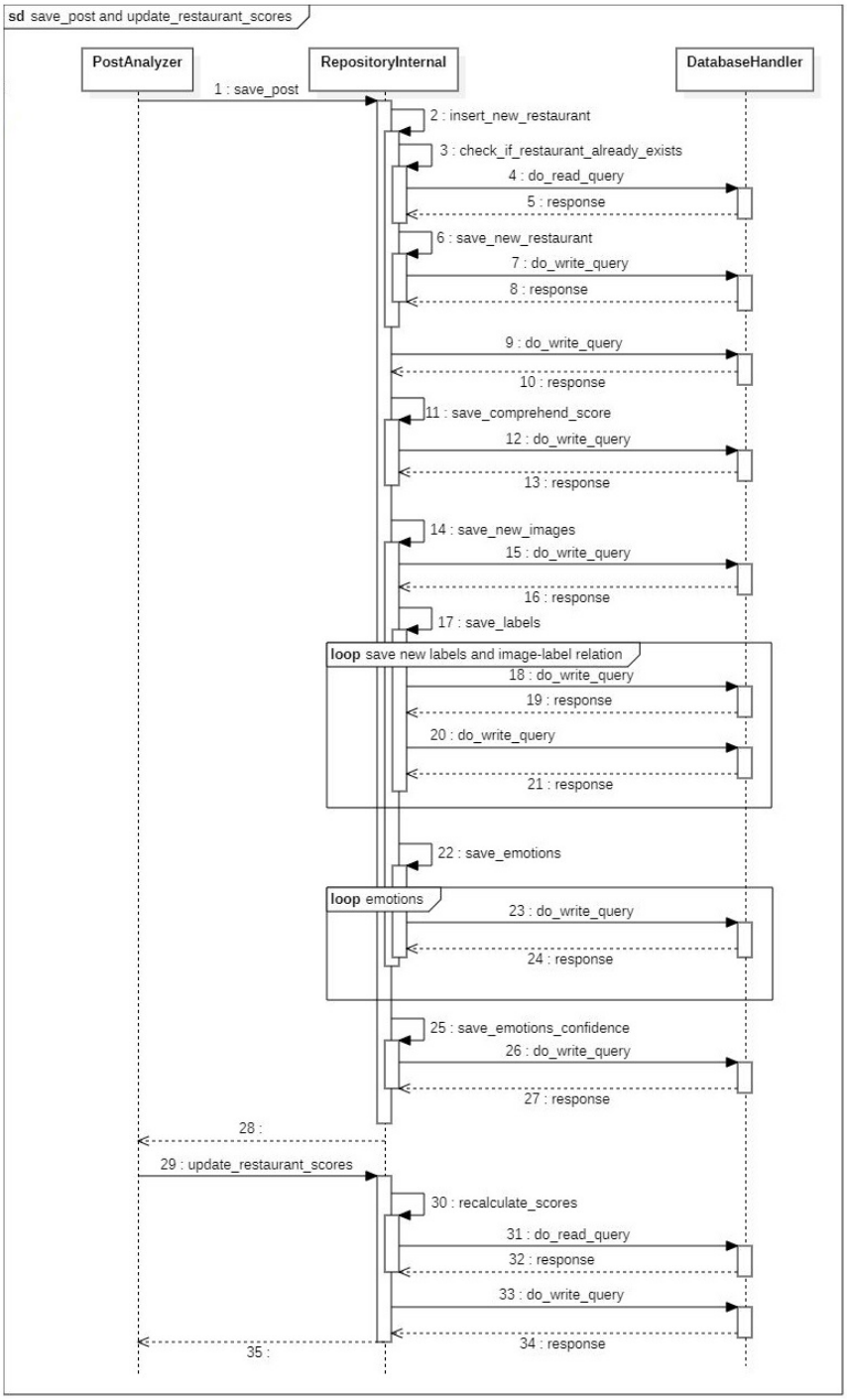

Figura 10: Ranking Service - Diagramma di sequenza - 2

2.3.4 Note sul processo di analisi

Dopo alcune prove ed osservazioni sono state fatte alcune decisioni relative al processo di analisi:

- Dopo l'analisi della immagini, verranno salvate nel database solo le label relative al cibo (quindi con padre "Food") e con confidenza di almeno il 90
- Solo se nelle immagini viene trovata la label "Person" verrà effettuata l'analisi dei sentimenti nell'immagine tramite Rekognition, anche in questo caso verranno salvati solo i sentimenti predominanti con una confidenza di almeno il 90

2.3.5 Design pattern notevoli utilizzati

Per La realizzazione del Ranking Service sono stati utilizzati i seguenti design pattern:

- **Strategy:** Utlizzato per la realizzazione di PostAnalyzer ed EmojiAnalyzer, è stato scelto principalmente per la sua versatilità.
  Questo pattern infatti ci permette, nel caso in cui in futuro ci sia bisongo di un algortimo più avanzato per l'analisi dei post, con il minimo sfrozo e modifica del codice, di implementare la nuova classe che erediterà anch'essa dalla classe base Analyzer.
- **Static Factory:** Utilizzato per la creazione di oggetti dalle varie fonti accessibili dal microservizio, come la coda(SQS), e la creazione di liste contenti dati pronti per l'utilizzo nei metodi riguardanti il database.

2.3.6 Schema del database

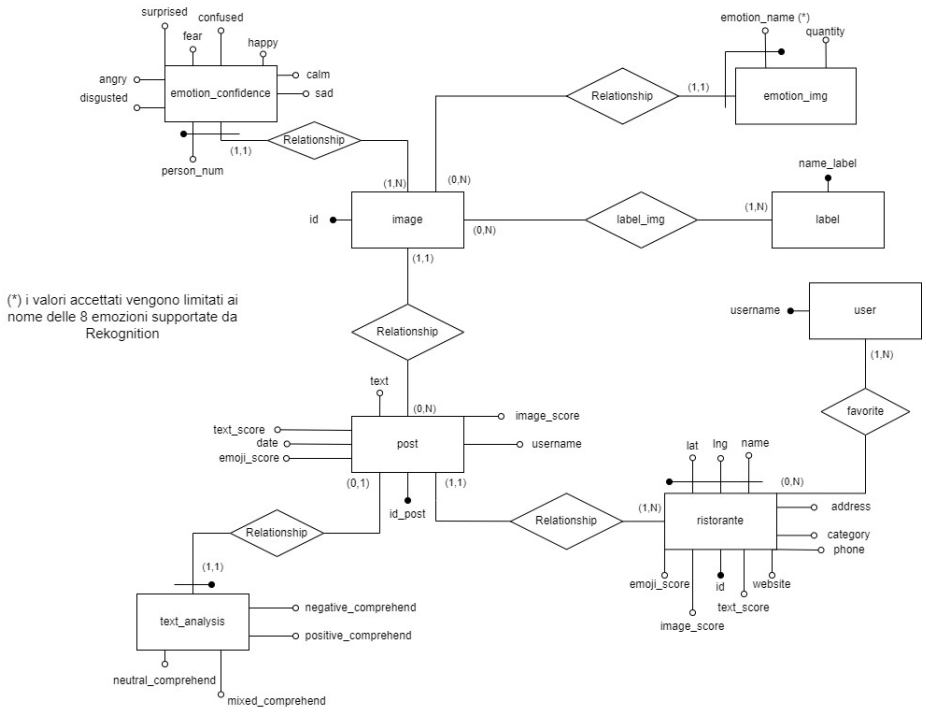

Figura 11: Ranking Service - Schema ER del database

2.4 Architettura del FrontEnd

3 Requisiti Soddisfatti

Fonte	Requisiti	Fonte	Requisiti
R1FW1	*Soddisfatto*	R2FW8.2	*Non Soddisfatto*
R1FW2	*Soddisfatto*	R2FW8.3	*Non Soddisfatto*
R1FW3	*Soddisfatto*	R2FW8.4	*Non Soddisfatto*
R1FE1	*Soddisfatto*	R1FW8.5	*Non Soddisfatto*
R1FE2	*Soddisfatto*	R3FW9	*Non Soddisfatto*
R1FE3	*Soddisfatto*	R3FW9.1	*Non Soddisfatto*
R1FE4	*Soddisfatto*	R3FW9.2	*Non Soddisfatto*
R1FE5	*Soddisfatto*	R1FW10	*Soddisfatto*
R1FE6	*Soddisfatto*	R2FE14	?????????
R1FE7	*Soddisfatto*	R1FW11	*Soddisfatto*
R1FW4	*Soddisfatto*	R2FW11.1	*Soddisfatto*
R1FW4.1	*Non Soddisfatto*	R2FW11.2	*Soddisfatto*
R1FE8	*Non Soddisfatto*	R2FW11.3	*Soddisfatto*
R1FW4.2	*Non Soddisfatto*	R2FW11.4	*Non Soddisfatto*
R1FE9	*Non Soddisfatto*	R1FW11.1.1	*Soddisfatto*
R3FW4.3	*Soddisfatto*	R2FW11.1.2	*Soddisfatto*
R3FE15	*Soddisfatto*	R2FW11.1.3	*Soddisfatto*
R1FW5	*Soddisfatto*	R3FW11.1.4	*Non Soddisfatto*
R1FW5	*Soddisfatto*	R2FW11.1.5	*Non Soddisfatto*
R2FE10	*Non Soddisfatto*	R2FW11.1.6	*Soddisfatto*
R2FE11	*Non Soddisfatto*	R1FW11.2.1	*Soddisfatto*
R2FE12	*Non Soddisfatto*	R1FW11.2.2	*Soddisfatto*
R1FW7	*Soddisfatto*	R1FW11.2.3	*Soddisfatto*
R1FW7.1	*Soddisfatto*	R1FW11.2.4	*Soddisfatto*
R1FW7.2	*Soddisfatto*	R2FW11.3.1	*Soddisfatto*
R2FW7.3	*Soddisfatto*	R2FW11.3.2	*Soddisfatto*
R2FW7.4	*Soddisfatto*	R2FW11.3.3	*Soddisfatto*
R1FW8	*Non Soddisfatto*	R2FW12	*Non Soddisfatto*
R2FE13	*Non Soddisfatto*	R2FW13	*Non Soddisfatto*
R1FW8.1	*Non Soddisfatto*	R3F1	*Non Soddisfatto*

Tabella 2: Tabella soddisfacimento requisiti