UNIVERSITÀ DEGLI STUDI DI BERGAMO

EMANUELE PERICO 1046601

CHRISTIAN CELESTINO 1046204

LAMBDA FUNCTION IMPLEMENTATE

Per il nostro lavoro abbiamo implementato 4 Lambda Function relative a:

- Ricerca dei Talk tramite autore: (get_talks_by_speaker);
- Ricerca dei Talk tramite titolo del video: (get_talks_by_title);
- Ricerca dei Talk tramite data di pubblicazione: (get_talks_by_date);

Utilizzabili direttamente dall'utente finale.

- Riferimento ai video consigliati (get_wn_by_idx).

Difficilmente utilizzabile direttamente dagli utenti poiché necessita la conoscenza della stringa di id relativa al video di interesse. Per cui è utilizzabile per estensioni alle funzionalità dell'applicazione.

Gli handler delle prime tre Lambda Function sono strutturati come a lato:

Ovviamente all'interno della funzione talk.find bisognerà inserire i campi sui quali si è interessati effettuare la ricerca (autore, data e titolo).

```
connect_to_db().then(() => {
    console.log('=> get_all talks');
    talk.find({main_speaker: body.speaker})
        .skip((body.doc_per_page * body.page) - body.doc_per_page)
        .limit(body.doc_per_page)
        .then(talks => {
                callback(null, {
                    statusCode: 200,
                    body: JSON.stringify(talks)
                })
            }
        )
```

Per quanto riguarda la funzione del watch_next implementiamo l'handler come mostrato:

In questo caso abbiamo dovuto:

- modificare il file «Talk.js» aggiungendo allo schema il campo _id (di tipo String).
- inserire la funzione «select» per rendere più pulita la visualizzazione dei video consigliati.

```
connect_to_db().then(() => {
    console.log('=> get_all talks');
    talk.find({_id: body.identifier})
                .select('_id main_speaker title watch_next')
        .skip((body.doc_per_page * body.page) - body.doc_per_page)
        .limit(body.doc_per_page)
        .then(talks => {
                callback(null, {
                    statusCode: 200,
                    body: JSON.stringify(talks)
                })
            }
        )
```

ESPERIENZA UTENTE

Data quindi la seguente richiesta da parte dell'utente:


```
{
    "speaker": "Bill Gates",
    "doc_per_page": 10,
    "page": 1
}
```


Utilizzando gli Endpoint API delle Lambda Functions possiamo vedere, tramite un software esterno come «Postman», la visualizzazione da parte dell'utente dei dati ricevuti:


```
"main_speaker": "Bill Gates",
"title": "How we must respond to the coronavirus pandemic",
"details": "Philanthropist and Microsoft cofounder Bill Gates offers insights into the COVID-19 pandemic, discussing why testing and self-isolation are essential, which medical advancements show promise and what it will take
    for the world to endure this crisis. (This virtual conversation is part of the TED Connects series, hosted by head of TED Chris Anderson and current affairs curator Whitney Pennington Rodgers. Recorded March 24, 2020)",
"posted": "Posted Mar 2020",
"url": "https://www.ted.com/talks/bill_gates_how_we_must_respond_to_the_coronavirus_pandemic",
"tags": [
    "TED",
    "talks",
    "health care",
    "disease",
    "virus",
    "medicine",
    "global issues",
    "economics",
    "pandemic",
    "TED Connects",
    "coronavirus"
],
"watch_next": [
    {
        "wn_id": "f5ae6143fa79ba55deb5e9e720b4c681",
        "url": "https://www.ted.com/talks/seth_berkley_the_quest_for_the_coronavirus_vaccine"
    },
```


Utilizzando le altre Lambda Function (get_talks_by_title, get_talks_by_date) possiamo ottenere gli analoghi risultati della ricerca voluta.

Per quanto riguarda la Lambda Function relativa al watch_next, tramite la seguente ricerca per id:


```
{
    "identifier": "727789c61e70d66a8aafef489dcf1acd",
    "doc_per_page": 10,
    "page": 1
}
```


Si ottiene il seguente risultato:


```
"_id": "727789c61e70d66a8aafef489dcf1acd",
"main_speaker": "Patrick Forth",
"title": "How can companies continue to thrive in times of change?",
"watch_next": [
    {
        "wn_id": "1ae3398f6896a66fe76c5984f001e4fe",
        "url": "https://www.ted.com/talks/philip_evans_how_data_will_transform_business"
    },
    {
        "wn_id": "4f19b70b19a3b6535984650e2135f633",
        "url": "https://www.ted.com/talks/martin_reeves_how_to_build_a_business_that_lasts_100_years"
    },
    {
        "wn_id": "9f7b1654e792011b7e1c6f4288520226",
        "url": "https://www.ted.com/session/new?context=ted.www%2Fwatch-later"
    },
    {
        "wn_id": "ddf5528eb60809c05390a9f0b47005b7",
        "url": "https://www.ted.com/talks/rocio_lorenzo_how_diversity_makes_teams_more_innovative"
    }
```


Come detto questa funzione non è facilmente utilizzabile direttamente dall'utente finale, poiché dovrebbe inserire interamente il campo id senza errori.

Risulta invece molto utile se eseguita automaticamente al termine della visualizzazione di un video.

CRITICITÀ ED EVOLUZIONI

L'utente spesso potrebbe non ricordare il titolo del video per intero o non sapere nome o cognome dello speaker. Questo potrebbe portare la Lambda Function a non restituire nessun risultato, e ciò accade anche nel caso in cui non vengano inserite correttamente le maiuscole.

Per ovviare a questa problematica è possibile evolvere le nostre Lambda Function tramite l'inserimento di un operatore delle query di MongoDB all'interno della funzione; si tratta dell'operatore $regex che cerca pattern di stringhe all'interno dei campi String. Inoltre, con l'opzione "i" è possibile eliminare la CaseSensibility della funzione.

```
connect_to_db().then(() => {
    console.log('=> get_all talks');
    talk.find({main_speaker: {'$regex': body.speaker, '$options': 'i'} })
        .skip((body.doc_per_page * body.page) - body.doc_per_page)
        .limit(body.doc_per_page)
        .then(talks => {
                callback(null, {
                    statusCode: 200,
                    body: JSON.stringify(talks)
                })
            }
        )
```

Un'altra criticità è mostrata dal momento in cui all'interno del DB risiedono link errati. In questo caso il problema dovrebbe esser gestito a monte durante la creazione dei DataLake, in cui si dovrebbero pulire i dati da errori come questi.

LINK ALLE API UTILIZZABILI

- get_wn_by_idx
  https://ig9al0z92g.execute-api.us-east-1.amazonaws.com/default/Get_Watch_Next_by_Idx
- get_talks_by_date
  https://chjvqk4xaf.execute-api.us-east-1.amazonaws.com/default/Get_Talks_by_Date
- get_talks_by_speaker
  https://t8l74admw1.execute-api.us-east-1.amazonaws.com/default/Get_Talks_by_Speaker
- get_talks_by_title
  https://exi2gdi3di.execute-api.us-east-1.amazonaws.com/default/Get_Talks_by_Title