

TEDx TOK

Progetto parte 3:
Lambda

Mazzoleni Gabriele – 1079514
Masinari Gabriele - 1079692

Funzione MyTedx: Get_Watch_Next_By_Idx

Utilizzo

Questa Lambda Function consente di visualizzare tutti i video correlati, dato un ID iniziale di un video, visualizzando i seguenti parametri:

- Related_video_ids, ossia l'ID del video correlato.
- Related_video_title, ossia il titolo del video correlato.
- Related_presentedBy, ossia il presentatore del video.

Esperienza utente

Questo permette al nostro utente di avere una chiara comprensione della lista dei video correlati e di scegliere al meglio il video desiderato.

Codice della Lambda Function e API

```
connect_to_db().then(() => {
  console.log('=> get_all talks');
  talk.find({ "_id": body.id }).select('Related_videos -_id')
    .skip((body.doc_per_page * body.page) - body.doc_per_page)
    .limit(body.doc_per_page)
    .then(talks => {
      if (!talks || talks.length === 0) {
        throw new Error('Talk not found');
      }

      let relatedVideos = talks[0].Related_videos;

      // Remove _id from each related video
      relatedVideos = relatedVideos.map(video => {
        const { _id, ...rest } = video.toObject();
        return rest;
      });

      callback(null, {
        statusCode: 200,
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify(relatedVideos)
      })
    })
})
```

Questa Lambda function può essere testata con il seguente link API:

https://ee9l9w52bk.execute-api.us-east-1.amazonaws.com/default/Get_Watch_Next_by_Idx

Risultato della funzione: Get_Watch_Next_By_Idx

Parametro in INPUT:

```
1  {  
2  |    "id": "526880"  
3  |}
```

OUTPUT testato tramite Postman:

```
[  
  {  
    "related_video_ids": "109914",  
    "related_video_title": "Whatever happened to the hole in the ozone layer?",  
    "related_presentedBy": "Stephanie Honchell Smith"  
  },  
  {  
    "related_video_ids": "76541",  
    "related_video_title": "What's in the air you breathe?",  
    "related_presentedBy": "Amy Hrdina and Jesse Kroll"  
  },  
  {  
    "related_video_ids": "100294",  
    "related_video_title": "Why plague doctors wore beaked masks",  
    "related_presentedBy": "TED-Ed"  
  }  
]
```

Lambda Function TedTok: Get_Talk_List_By_Tags

Utilizzo

Questa Lambda Function, dato un array di tag, restituisce un JSON con tutti i talk corrispondenti a quei tag, ordinati secondo la data di pubblicazione in modo decrescente e senza duplicati.

Esperienza utente

Questo permette al nostro utente, una volta selezionati i tag di suo interesse, di visualizzarli tramite il sistema.

Sono presentati per primi i talk più recenti, al fine di dare la precedenza ai discorsi più contemporanei, un fattore particolarmente importante per alcune tematiche, come la tecnologia informatica.

Codice della funzione: Get_Talk_List_By_Tags

```
1 const mongoose = require('mongoose');
2
3 const talk_schema = new mongoose.Schema({
4   _id: String,
5   speakers: String,
6   title: String,
7   url: String,
8   duration: String,
9   publishedAt: Date,
10 }, { collection: 'TedTok_data' });
11
12 module.exports = mongoose.model('talk', talk_schema);
```

Questo è il modello utilizzato per la visualizzazione dei dati (file Talk.js), il codice del gestore della funzione (file handler.js) è mostrato alla diapositiva successiva.

La lambda function, è disponibile attraverso questo link API:

https://kz0253u01m.execute-api.us-east-1.amazonaws.com/default/Get_Talk_List_By_Tags

Codice della funzione: Get_Talk_List_By_Tags

```
1 const connect_to_db = require('./db');
2 const talk = require('./Talk');
3
4 module.exports.get_by_tag = (event, context, callback) => {
5   context.callbackWaitsForEmptyEventLoop = false;
6   console.log('Received event:', JSON.stringify(event, null, 2));
7
8   let body = {};
9   if (event.body) {
10     try {
11       body = JSON.parse(event.body);
12     } catch (error) {
13       console.error('Error parsing body:', error);
14       callback(null, {
15         statusCode: 400,
16         headers: { 'Content-Type': 'text/plain' },
17         body: 'Invalid JSON body'
18       });
19       return;
20     }
21   }
22
23   const tags = body.tags;
24   if (!tags || !Array.isArray(tags) || tags.length === 0) {
25     callback(null, {
26       statusCode: 400,
27       headers: { 'Content-Type': 'text/plain' },
28       body: 'Tags are null or they are not an array.'
29     });
30     return;
31   }
32
33   if (!body.doc_per_page) {
34     body.doc_per_page = 10;
35   }
36   if (!body.page) {
37     body.page = 1;
38   }
```

```
40   connect_to_db().then(() => {
41     console.log('-> get_all talks');
42     talk.find({ tags: { '$in': tags } })
43       .sort({ publishedAt: -1 })
44       .then(talks => {
45         console.log('Talks fetched before removing duplicates:', talks.length);
46
47         // Rimuovi i duplicati
48         const uniqueTalks = talks.reduce((acc, talk) => {
49           if (!acc.some(t => t._id === talk._id)) {
50             acc.push(talk);
51           }
52           return acc;
53         }, []);
54
55         console.log('Unique talks:', uniqueTalks.length);
56
57         // Applica la paginazione
58         const paginatedTalks = uniqueTalks.slice(
59           (body.doc_per_page * (body.page - 1)),
60           (body.doc_per_page * body.page)
61         );
62
63         callback(null, {
64           statusCode: 200,
65           body: JSON.stringify(paginatedTalks)
66         });
67       })
68       .catch(err => {
69         console.error('Error fetching talks:', err);
70         callback(null, {
71           statusCode: err.statusCode || 500,
72           headers: { 'Content-Type': 'text/plain' },
73           body: 'Could not fetch the talks.'
74         });
75       });
76   }).catch(err => {
77     console.error('Error connecting to database:', err);
78     callback(null, {
79       statusCode: 500,
80       headers: { 'Content-Type': 'text/plain' },
81       body: 'Could not connect to the database.'
82     });
83   });
84 }
```

Risultato della funzione: Get_Talk_List_By_Tags

Parametro in INPUT:

```
1  {  
2  ... "tags":["wildlife"]  
3  }
```

OUTPUT testato tramite Postman:

```
{  
  "_id": "624879",  
  "slug": "rebecca_mcmackin_let_your_garden_grow_wild",  
  "speakers": "Rebecca McMackin",  
  "title": "Let your garden grow wild",  
  "url": "https://www.ted.com/talks/rebecca_mcmackin_let_your_garden_grow_wild",  
  "description": "Many gardeners work hard to maintain clean, tidy environments ... which is the exact opposite of what wildlife wants.",  
  "duration": "742",  
  "publishedAt": "2024-03-27T14:46:24.000Z",  
  "tags": [  
    "climate change",  
    "environment",  
    "animals",  
    "nature",  
    "biodiversity",  
    "trees",  
    "ecology",  
    "plants",  
    "gardening",  
    "wildlife"  
  ],  
  "Related_videos": [  
    {  
      "related_video_ids": "125757",  
      "related_video_title": "How poop turns into forests",  
      "related_presentedBy": "Ludmila Rattis"  
    },  
    ...  
  ]  
}
```


Criticità tecniche

- Non vengono effettuati controlli sugli input, ossia che l'utente può inserire ciò che vuole senza vincoli.
- Potrebbero verificarsi tempi di computazione elevati riguardo all'ordinamento e alla rimozione dei duplicati



Possibili evoluzioni

- Dare maggiore dettaglio agli errori mostrando all'utente l'errore, se esistente.
- Assicurarsi che tutti gli input siano vincolati



TEDx TOK

[Trello board](#)

[GitHub](#)