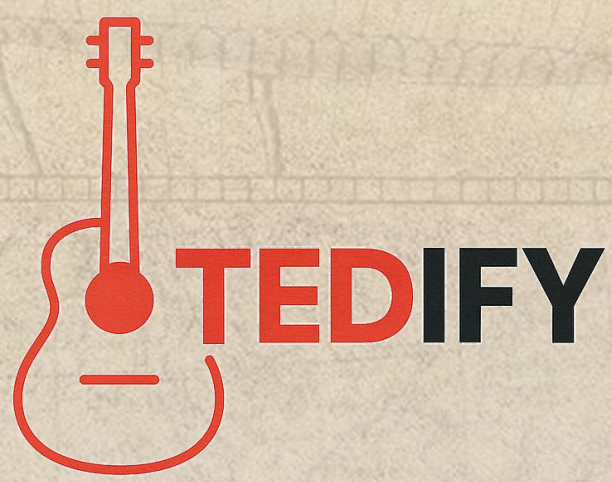
# TEDify

## «Musica che ispira, idee che suonano»




DAMIANO CARRARA (MATR. 1067871) – HOMEWORK 3

RELEASE PLAN

REPOSITORY

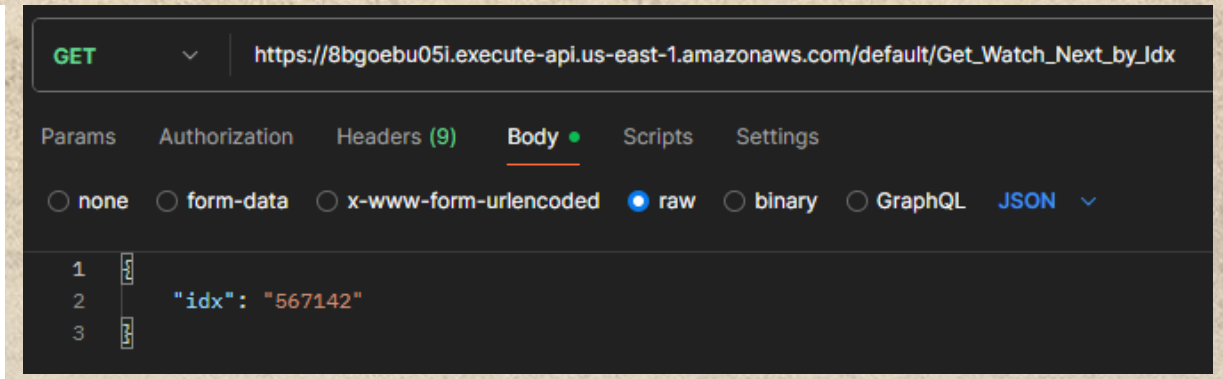
# IMPLEMENTAZIONE GET_WATCH_NEXT

- Esperienza utente: la funzione, ricevuto in input l'id di un video, restituisce la lista dei video ad esso correlati. Questo facilita la navigazione all'interno dei talk.
- Nel caso siano presenti malfunzionamenti (id in input errato, riferimenti inconsistenti…) la funzione restituisce un messaggio di errore appropriato.
- Esposta tramite API Gateway al seguente link: https://8bgoebu05i.execute-api.us-east-1.amazonaws.com/default/Get_Watch_Next_by_Idx

# GET_WATCH_NEXT_BY_IDX

```
await connect_to_db();
console.log('=> Connected to MongoDB');

const talks = await talk.find({ _id: idx })
  .skip((doc_per_page * page) - doc_per_page)
  .limit(doc_per_page);

if (!talks || talks.length === 0) {
  return callback(null, {
    statusCode: 404,
    body: 'No talk found with the given idx.'
  });
}

const watch_next_ids = (talks[0].Watch_next_id || []).map(String);

const watchnext = await talk.find({ _id: { $in: watch_next_ids } });

if (!watchnext || watchnext.length === 0) {
  return callback(null, {
    statusCode: 200,
    body: JSON.stringify({
      message: "Watch next non presenti nel dataset.",
      suggestions: []
    })
  });
}
```

Codice completo su GitHub




Esempio di richiesta con Postman

# ESEMPIO DI DOCUMENTO SU MONGODB

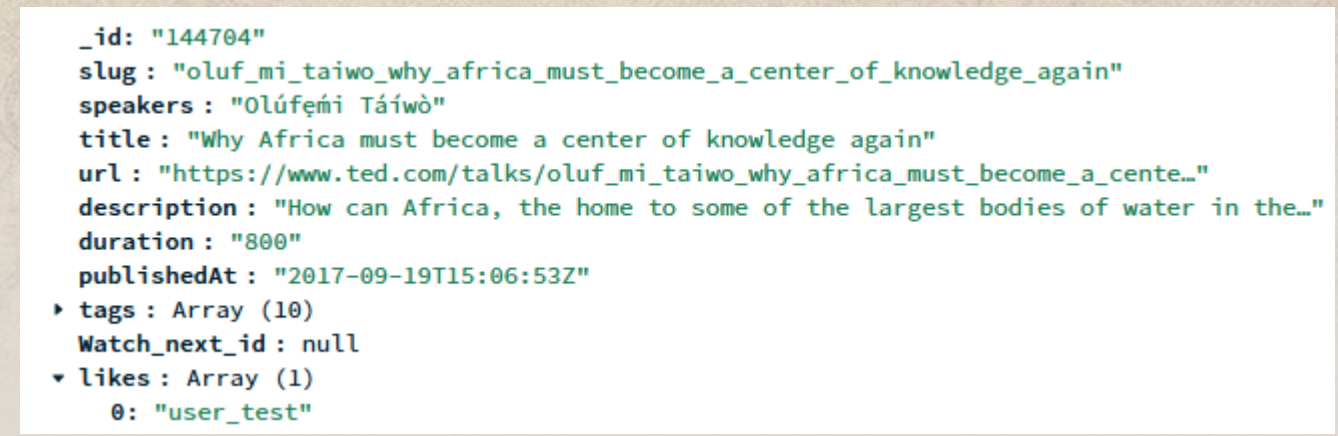

```
[
    {
        "_id": "144704",
        "slug": "oluf_mi_taiwo_why_africa_must_become_a_center_of_knowledge_again",
        "speakers": "Olúfẹ́mi Táíwò",
        "title": "Why Africa must become a center of knowledge again",
        "url": "https://www.ted.com/talks/oluf_mi_taiwo_why_africa_must_become_a_center_of_knowledge_again",
        "description": "How can Africa, the home to some of the largest bodies of water in the world, be said to have a water crisis? It doesn't, says Olúfẹ́mi Táíwò -- it has a
            knowledge crisis. Táíwò suggests that lack of knowledge on important topics like water and food is what stands between Africa's current state and a future of prosperity. In
            powerful talk, he calls for Africa to make the production of knowledge within the continent rewarding and reclaim its position as a locus of learning on behalf of humanity."
        "duration": "800",
        "publishedAt": "2017-09-19T15:06:53Z",
        "tags": [
            "Africa",
            "history",
            "philosophy",
            "library",
            "leadership",
            "water",
            "society",
            "agriculture",
            "resources",
            "rivers"
        ],
        "Watch_next_id": null,
        "likes": [
            "user_test"
        ]
    }
]
```


Risultato ottenuto in Postman

# IMPLEMENTAZIONE LIKE_UNLIKE

- Esperienza utente: l'utente può mettere «mi piace» ad un talk, o rimuoverlo nel caso abbia precedentemente messo «mi piace» al video in questione.
- Ricevuti in input l'id dell'utente e del video, la funzione aggiunge all'array «likes» (che viene generato se non è già presente) del documento lo user_id, o lo rimuove nel caso esso sia già presente.

Esempio di documento su MongoDB



# LIKE_UNLIKE

```
4   module.exports.like_unlike = async (event, context, callback) => {
5     context.callbackWaitsForEmptyEventLoop = false;
6
7     try {
8       console.log('Received event:', JSON.stringify(event, null, 2));
9
10      let body = event.body ? JSON.parse(event.body) : {};
11      const { videoId, userId } = body;
12
13      if (!videoId || !userId) {
14        return callback(null, {
15          statusCode: 400,
16          headers: { 'Content-Type': 'application/json' },
17          body: JSON.stringify({ message: 'Missing videoId or userId' })
18        });
19      }
20
21      await connect_to_db();
22      console.log('=> Connected to DB');
23
24      const talk = await Talk.findOne({ _id: videoId });
25
26      if (!talk) {
27        return callback(null, {
28          statusCode: 404,
29          headers: { 'Content-Type': 'application/json' },
30          body: JSON.stringify({ message: 'Video not found' })
31        });
32      }
```

```
34      let hasLiked = talk.likes && talk.likes.includes(userId);
35
36      const update = hasLiked
37        ? { $pull: { likes: userId } }
38        : { $addToSet: { likes: userId } };
39
40      await Talk.updateOne({ _id: videoId }, update);
41
42      return callback(null, {
43        statusCode: 200,
44        headers: { 'Content-Type': 'application/json' },
45        body: JSON.stringify({
46          liked: !hasLiked,
47          message: hasLiked ? 'Like removed' : 'Like added'
48        })
49      });
50
51    } catch (err) {
52      console.error('Error occurred:', err);
53      return callback(null, {
54        statusCode: 500,
55        headers: { 'Content-Type': 'text/plain' },
56        body: 'Internal server error: ' + err.message
57      });
58    }
59  };
```

Link API: https://w0vv1dpa34.execute-api.us-east-1.amazonaws.com/default/Like_Unlike

# CRITICITÀ TECNICHE

- Autenticazione: al momento le API sono pubbliche, quindi chiunque può utilizzarle e mettere mi piace ai talk senza autenticarsi.
- Scarsa quantità di talk: come sottolineato nella presentazione precedente, la collezione tedify ha solo 233 documenti, e quasi tutti presentano un array watch_next vuoto. Di conseguenza le Lambda functions sono state implementate sulla collezione tedx_data, in modo da poter mostrare degli esempi concreti.
- Al momento si corre il rischio di proporre all'utente dei video già visualizzati. Per ovviare a questo problema e proporre sempre video nuovi si dovrebbe tenere traccia nel database dei video già visti da ciascun utente.

# POSSIBILI EVOLUZIONI

- Aggiungere autenticazione con AWS Cognito, gestendo registrazione, login e token per l'accesso alle API.
- Creare in automatico, per ciascun utente, una playlist «preferiti», contenente tutti i talk a cui è stato messo like.
- Inserire una sezione commenti, che a loro volta possono ricevere like e dislike.
- Implementare una funzione che restituisca il numero di like per il video specificato.