

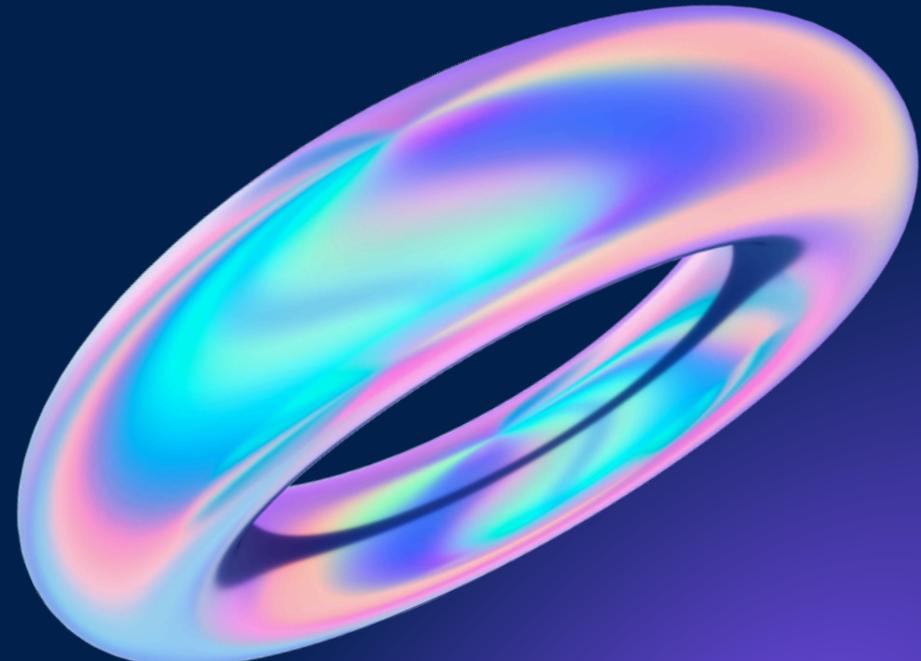
Tecnologie cloud & mobile

[Trello](#) [GitHub](#)



HealthyTEDx

Migliora la tua salute



Luisi Giuseppe 1073970
Zanchi Michael 1080756

GET WATCH NEXT BY ID

```
7 module.exports.get_talks_by_watch_next = (event, context, callback) => {
8   context.callbackWaitsForEmptyEventLoop = false;
9   console.log('Received event:', JSON.stringify(event, null, 2));
10  let body = {};
11  if (event.body) {
12    body = JSON.parse(event.body)
13  }
14  // set default
15  if(!body.idx) {
16    callback(null, {
17      statusCode: 500,
18      headers: { 'Content-Type': 'text/plain' },
19      body: 'Could not fetch the talks. ID is null.'
20    })
21  }
22
23  if (!body.doc_per_page) {
24    body.doc_per_page = 10
25  }
26  if (!body.page) {
27    body.page = 1
28  }
29
30  connect_to_db().then(() => {
31    console.log('=> get_all talks');
32    talk.find({"_id": body.idx})
33      .skip((body.doc_per_page * body.page) - body.doc_per_page)
34      .limit(body.doc_per_page)
35      .then(talks => {
36        callback(null, {
37          statusCode: 200,
38          body: JSON.stringify(talks)
39        })
40      }
41    )
42    .catch(err =>
43      callback(null, {
44        statusCode: err.statusCode || 500,
45        headers: { 'Content-Type': 'text/plain' },
46        body: 'Could not fetch the talks'
47      })
48  })
49}
```

In questo caso, la funzione Lambda, dato l'ID di un video, ritorna il "next video" da guardare. La funzione prima di tutto controlla se l'id è presente nel body; se l'id è null viene ritornato un errore "Could not fetch the talks. ID is null ", in caso contrario prosegue.

Dopo aver effettuato questa prima modifica, la Lambda function si collega con il database (in questo caso con mongoDB, nello specifico utilizziamo il db creato nella consegna precedente).

Una volta connessa viene eseguita una query per cercare l'id. Se la query viene conclusa con successo viene restituito uno "statusCode"=200, altrimenti viene restituito un errore ("statusCode"=500).

Esempio di Watch next

```
{  
  "_id": "526880",  
  "id_related": "526880",  
  "slug": "george_zaidan_how_do_gas_masks_actually_work",  
  "speakers": "George Zaidan",  
  "title": "How do gas masks actually work?",  
  "url": "https://www.ted.com/talks/george_zaidan_how_do_gas_masks_actually_work",  
  "description": "You might think of gas masks as clunky military-looking devices. But in the near future, we may need to rely on these filters as part of our everyday lives. In addition to emerging diseases, wildfire frequency has more than tripled, and climate change has increased toxic ground level ozone. So how do these masks work, and can they protect us from airborne threats? George Zaidan investigates. [Directed by Michael Kalopaidis, Zedem Media, narrated by George Zaidan, music by Manolis Manoli].",  
  "duration": "254",  
  "publishedAt": "2024-04-30T15:14:51Z",  
  "image_url": "https://talkstar-assets.s3.amazonaws.com/production/talks/talk_128547/250158f0-4687-41d3-abbe-df39232ee19a/gasmaskstextless.jpg",  
}
```

```
"id_related_video": [  
  "109914",  
  "100294",  
  "76541"  
,  
  "title_related_video": [  
    "Whatever happened to the hole in the ozone layer?",  
    "Why plague doctors wore beaked masks",  
    "What's in the air you breathe?"  
,  
]
```

```
  "tags": [  
    "environment",  
    "technology",  
    "design",  
    "education",  
    "natural disaster",  
    "chemistry",  
    "TED-Ed",  
    "animation"  
]
```

Get Talk By Feedback

Script completo

```
const connect_to_db = require('./db');

// GET BY TALK HANDLER

const Feedback = require('./Feedback');

module.exports.get_talks_by_feedback = async (event, context, callback) => {
    context.callbackWaitsForEmptyEventLoop = false;
    console.log('Received event:', JSON.stringify(event, null, 2));
    let body = {};
    if (event.body) {
        body = JSON.parse(event.body)
    }
    const { _id, feedback_text, valutazione } = body;

    // set default
    if (!_id || !feedback_text || typeof valutazione !== 'number') {
        callback(null, {
            statusCode: 500,
            headers: { 'Content-Type': 'text/plain' },
            body: 'Invalid input data.'
        })
    }
    await connect_to_db();

    const newFeedback = new Feedback({
        id,
        feedback_text,
        valutazione
    });
}
```

Per quanto riguarda il job 2 abbiamo deciso di implementare la funzionalità riguardante i feedback. Nello specifico gestisce la raccolta di feedback per i talk TEDx utilizzando una Lambda Function di AWS. La funzione Lambda viene attivata tramite una richiesta HTTP (ad esempio su Postman) e si occupa di connettersi a un database MongoDB Atlas per salvare i dati riguardanti i feedback dei video.

Se i campi “_id” o “feedback_text” sono vuoti oppure il campo “valutazione” contiene un valore che non è un numero viene generato l’errore 500.

Se i dati sono validi la funzione si connette a MongoDB Atlas e le informazioni vengono salavate in tedx_feedback. Se l’operazione si conclude con successo viene restituito un messaggio di conferma , altrimenti viene restituito un errore

ESEMPI

The screenshot shows the Postman interface with a GET request to `https://y9d2kfd9aj.execute-api.us-east-1.amazonaws.com/default/Get_Talk_By_Feedback`. The Body tab is selected, showing a raw JSON payload:

```
1 {  
2   "_id": "466415",  
3   "feedback_text": "Video molto bello",  
4   "valutazione": 4  
5 }  
6
```

The response body shows a success message: "Feedback ricevuto con successo!"

Esempio di Postman

Two MongoDB documents are shown:

```
_id: "525470"  
feedback_text : "Ottimo video!"  
valutazione : 5  
__v : 0
```



```
_id: "504947"  
feedback_text : "Video interessante"  
valutazione : 4  
__v : 0
```

Esempi dei feedback salvati su MongoDB

Three MongoDB documents are shown:

```
_id: "519458"  
feedback_text : "Consiglio a tutti di guardare questo video"  
valutazione : 5  
__v : 0
```



```
_id: "514217"  
feedback_text : "Video molto motivante"  
valutazione : 5  
__v : 0
```

CRITICITÀ

- Una prima criticità riguarda il fatto che si potrebbero incontrare errori di duplicazione se si tenta di inserire un feedback con un id già presente nel database.
- Un'altra criticità di questa implementazione è la seguente: la funzione stabilisce una nuova connessione al database ogni volta che viene invocata, il che può aumentare la latenza e il consumo di risorse.

FINE