GEOX

UNA STRADA VERSO IL FUTURO

LAMBDA NEXT

WATCH_NEXT_Handler

```
const talk = require('./Talk');
const talk_ref = require('./Talk’);
.
.
.
connect_to_db().then(() => {
        console.log('=> get_all idx of next talks');
        talk.findOne({_id:body.id})
            .then(talk_find =>

talk_ref.find({_id:talk_find.next_idx},'title
main_speaker url details').then(talked =>
                    {
                    callback(null, {
                        statusCode: 200,
                        body: JSON.stringify(talked)
                    })
                })
            )
```

WATCH_NEXT_Talk_Schema

```
const talk_schema = new Schema({
    _id: String,
    title: String,
    url: String,
    details:String,
    main_speaker: String,
    next_idx: ['talk_ref']
}, { collection: 'tedz_data' });
```

LAMBDA NEXT

Dato come parametro l'ID di un talk, la funzione lambda permette di ottenere le informazioni relative ai talk suggeriti, quali titolo, URL, relatore e dettagli.

GET: https://.../Get_Watch_Next_by_Idx
Body JSON:

```
{
        "id" : "8c1fad5ce0dab8908dee527f88697ce2"
}
```

Result JSON:


```
[
  {
    "_id": "43014f52663d1cf317a606b9c4dfe2fd",
    "main_speaker": "Alex Gendler",
    "title": "History vs. Christopher Columbus",
    "details": "Many people in the United States and ... ",
    "url": "https://www.ted.com/talks/alex_gendler_history_vs_christopher_columbus"
  }...]
```

LAMBDA GEO

GEO_AREA_Handler

```
if(!body.continente && !body.nazione && !body.citta){
        callback(null, {
            statusCode: 500,
            headers: { 'Content-Type': 'text/plain'},
            body: 'Could not fetch the talks. No parameter is defined.'})}
...
if(!body.nazione && !body.citta){
        talk.find({'geo_area.continent':body.continente},'title url main_speaker details geo_area.continent
geo_area.nation geo_area.city')
            .skip((body.doc_per_page * body.page) - body.doc_per_page)
            .limit(body.doc_per_page)
```

GEO_AREA_Talk_Schema

```
const talk_schema = new mongoose.Schema({
    title: String,
    url: String,
    details: String,
    main_speaker: String

}, { collection: 'tedz_data' });
```

LAMBDA GEO

Assegnato almeno uno dei 3 parametri disponibili, quali continente, nazione e città, la function ricerca tutti i talk relativi all'area geografica di interesse, fornendo tutti dettagli dei talk trovati.
Nella slide precedente si è riportata la parte di codice relativa alla ricerca basata su continente.
Oltre a questa, si trovano a disposizione anche quelle riguardanti la nazione e la città.

GET: https://.../default/Get_Tedx_by_Geo_Area
Body JSON:

```
{
                "continente" : "Europe"
}
```

Result JSON:

```
[
   {
      "main_speaker": "Butterscotch",
      "title": "\"Accept Who I Am\"",
      "details": "Firing off her formidable ...",
                "url": "https://www.ted.com/talks/butterscotch_accept_who_i_am",
      "geo_area": {
         "continent": "Europe",
         "nation": "Italy",
         "city": "Bergamo"
      }
   }...]
```

LAMBDA VOTE

VOTE_Handler

```
 talk.findOne({_id:body.id})
                .then(talks => {
                    var
obj={"title":talks.title,"url":talks.url,"main_speaker":t
alks.main_speaker};
                    var bd=JSON.stringify(obj)
                    var n=0
                    var sum=0
                    while(n<talks.vote_user.length){
                        sum+=talks.vote_user[n].vote
                    if(talks.vote_user[n].vote>=new
Number(body.voto_limite)){
bd=bd+JSON.stringify(talks.vote_user[n])}
                    n+=1
                }
                const media=sum/n
                bd=bd+JSON.stringify({"media
voti":media,"voti totali":n})
                callback(null, {
                    statusCode: 200,
                    body:bd
                })
```

VOTE_Talk_Schema

```
onst talk_schema = new mongoose.Schema({
    _id: String,
    title: String,
    url:String,
    main_speaker:String,
    vote_user: [{vote:Number,mail_user:String}]
}, { collection: 'tedz_data' });
```

LAMBDA VOTE

Assegnato come parametro obbligatorio l'ID, la function mostra tutti i voti relativi al talk preso in considerazione, inclusa la media delle valutazioni.
Con l'aggiunta di ulteriori parametri, è possibile effettuare una ricerca più mirata.

```
GET: https://.../Get_Vote_by_Idx
Body JSON:
{
                "id" : "8c1fad5ce0dab8908dee527f88697ce2",
                "voto_limite" : 3
}

Result JSON:
{
    "title": "History vs. Sigmund Freud",
    "url": "https://www.ted.com/talks/todd_dufresne_history_vs_sigmund_freud",
    "main_speaker": "Todd Dufresne"
vote_user:[{
    "date": "2020-02-10",
    "time": "8:59:20",
    "mail_user": "aathelstan15@ustream.tv",
    "vote": 3
}]
    "media voti": 3.2280701754385963,
    "voti totali": 57
}
```

ESPERIENZA UTENTE

WATCH_NEXT	Ricerca dei Talk correlati ad un determinato Talk, fornendo informazioni essenziali relative ad essi.
GEO_TALK	Dato un continente, una nazione o una città, fornisce le informazioni relative a tutti i talk associati all'area geografica di appartenenza.
VOTE_USER	Dato un ID, restituisce la lista dei voti associati ai talk. Ogni voto possiede informazioni aggiuntive quali data di emissione del voto, ora di emissione e autore, oltre al voto stesso.

CRITICITÀ TECNICHE

- Riguardo l'API relativa alla ricerca dei talk suggeriti, nel caso in cui le informazioni non dovessero bastare, si pone innanzi la necessità di sfruttare una seconda API relativa alla ricerca delle informazioni di ogni singolo talk.

- Nella Lambda function relativa alla ricerca basata su area geografica, potrebbe avvenire un caso di incoerenza tra dati, dove, ad esempio, una nazione non è inclusa in un continente e l'API non sa a quale parametro fare riferimento.

POSSIBILI EVOLUZIONI

- Update della API relativa ai video correlati con maggiori informazioni connesse ad essi.

- Aggiunta di ulteriori criteri di ricerca riguardo le valutazioni.
  Es. Voti di determinati utenti, Talk valutati da un utente, etc.