# TEDcoffee

## HOMEWORK 3
## *Lambda Function*


**Daniel Hernan Altamirano**
**Davide Cesani**
**Federico Nespoli**

# **TEDcoffee** – Lambda Function Implementate

- ***Lambda Function rivolte all'utente***
  - ***Get_Talks_By_Author***: dato il nome, *anche parziale*, di un autore, restituisce:
    - *Caso Nome parziale*: l'elenco dei talks sostenuti dagli autori il cui nome contiene la stringa in input.
    - *Caso Nome completo*: l'elenco dei talks sostenuti *solo* dall'autore ricercato.
  - ***Get_Talks_By_Title***: dato il titolo, *anche parziale*, di un talk, restituisce:
    - *Caso Titolo parziale*: l'elenco dei talks il cui titolo contiene la stringa in input.
    - *Caso Titolo completo*: il talk con il titolo ricercato.

- ***Lambda function di servizio***
  - ***Get_Watch_Next_By_Idx***: dato in input l'idx di un talk, restituisce gli idx e i relativi url dei Watch_Next_Talk associati.
    - *Non rivolta agli utenti ma utilizzata per la visualizzazione dei talks successivi al corrente.*
  - ***Get_Shorter***: data una durata, nel formato hh:mm:ss, restituisce l'elenco dei talks con durata non maggiore.
    - *Non rivolta agli utenti ma utilizzata per la visualizzazione dei talks più brevi.*

*Tutte le Lambda Function implementate hanno la stessa struttura di base vista durante la lezione.*
*Verranno dunque presentate sole le modifiche principali effettuate.*

**Daniel Hernan Altamirano**
**Davide Cesani**
**Federico Nespoli**

# **TEDcoffee** – Lambda Function rivolte all'utente
## ***Get_Talks_By_Author***

- **Handler**: handler_TalksByAuthor.get_by_author
- **Modifiche al gestore**:

```
talk.find({main_speaker: {'$regex': body.main_speaker, '$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)
            })
        }
    )
    .catch(err =>
        callback(null, {
            statusCode: err.statusCode || 500,
            headers: { 'Content-Type': 'text/plain' },
            body: 'Could not fetch the talks.'
        })
    );
```

  - $regex è l'equivalente MongoDB della clausola LIKE di SQL.
  - $option permette di rendere la ricerca case UNsensitive

- **Esperienza utente**: chi utilizza questa funzione desidera visualizzare i talks tenuti da uno specifico autore. Tuttavia il nome potrebbe essere difficile da scrivere. L'utente può quindi effettuare una ricerca con un nome parziale e visualizzare tutte le possibili corrispondenze.

Daniel Hernan Altamirano
Davide Cesani
Federico Nespoli

# **TEDcoffee** – Lambda Function rivolte all'utente
# ***Get_Talks_By_Title***

- **Handler**: handler_TalksByTitle.get_by_title
- **Modifiche al gestore**:

```
talk.find({title: {'$regex': body.title, '$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)
            })
        }
    )
    .catch(err =>
        callback(null, {
            statusCode: err.statusCode || 500,
            headers: { 'Content-Type': 'text/plain' },
            body: 'Could not fetch the talks.'
        })
    );
```

- $regex è l'equivalente MongoDB della clausola LIKE di SQL.
- $option permette di rendere la ricerca case UNsensitive

- **Esperienza utente**: chi utilizza questa funzione desidera visualizzare il talk con un titolo specifico. Tuttavia il titolo potrebbe essere difficile da scrivere. L'utente può quindi effettuare una ricerca con un titolo parziale e visualizzare tutte le possibili corrispondenze.

Daniel Hernan Altamirano
Davide Cesani
Federico Nespoli

# **TEDcoffee** – Lambda Function di servizio
# ***Get_Watch_Next_By_Idx***

- **Handler**: handler_WatchNextByIdx.get_watch_next_by_idx
- **Modifiche allo schema dati e al gestore**:

```
const talk_schema = new mongoose.Schema({
    _id: String,
    next_idx: Array,
    next_url: Array
}, { collection: 'TEDcoffee_data' });

module.exports = mongoose.model('WatchNexttalk', talk_schema);
```

- Schema dati ridefinito ad hoc per la funzione. In particolare, il focus si pone sull'idx del talk corrente, sugli idx dei talk ad esso successivi e i relativi url

```
talk.find({_id: body._id})
    .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)
            })
        }
    )
    .catch(err =>
        callback(null, {
            statusCode: err.statusCode || 500,
            headers: { 'Content-Type': 'text/plain' },
            body: 'Could not fetch the talks.'
        })
    );
```

- Ricerca nel DB tramite campo _id.

- **Obiettivo**: function utilizzata dal servizio per proporre all'utente i talks successivi a quello appena visualizzato.

Daniel Hernan Altamirano
Davide Cesani
Federico Nespoli

# **TEDcoffee** – Lambda Function di servizio ***Get_Shorter***

- **Handler**: handler_Shorter.get_shorter
- **Modifiche allo schema dati e al gestore**:

```
const talk_schema = new mongoose.Schema({
    _id: String,
    url: String,
    duration: String
}, { collection: 'TEDcoffee_data' });

module.exports = mongoose.model('talkShorter', talk_schema);
```

- Schema dati ridefinito ad hoc per la funzione. In particolare, il focus si pone sull'id dei talks che rispettano il vincolo di durata, sui relativi url e sulla durata stessa dei talks.

```
talk.find({duration: {$lte: body.duration}} )
    .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)
            })
        }
    )
    .catch(err =>
        callback(null, {
            statusCode: err.statusCode || 500,
            headers: { 'Content-Type': 'text/plain' },
            body: 'Could not fetch the talks.'

        })
    );
```

- Ricerca nel DB tramite campo duration. In particolare, sono elencati i talks con durata minore di quella in input

- **Obiettivo**: function utilizzata dal servizio per proporre all'utente i talks più brevi (con durata minore di una certa soglia).

Daniel Hernan Altamirano
Davide Cesani
Federico Nespoli

# **TEDcoffee** – Criticità ed Evoluzioni

- ***Criticità:***
  - Formato durata talks: la durata dei talks, per essere gestita correttamente, deve avere formato *hh:mm:ss*.
  - Presenza di record nel dataset *tedx_dataset* con campo *url* vuoto.

- ***Evoluzioni:***
  - sanificazione DB per evitare problemi legati alla mancanza di url.

Daniel Hernan Altamirano
Davide Cesani
Federico Nespoli