# TEDX NEWS

## 3° HOMEWORK

**Chiara Toffalori 1080710**
**Siria Ruggeri 1078977**

# Get_Watch_Next_by_Idx

## Schemi

- **Importazione di Mongoose**
- Lo schema **talkSchema** definisce la struttura dei documenti nella collezione tedx_data

```
const mongoose = require('mongoose');

// Definizione dello schema per la collezione tedx_data
const talkSchema = new mongoose.Schema({
    _id: String,            // ID univoco del talk
    slug: String,           // Slug del talk
    speakers: String,       // Relatori del talk
    title: String,          // Titolo del talk
    url: String,            // URL del talk
    description: String,    // Descrizione del talk
    image_url: String,      // URL dell'immagine
    duration: String,       // Durata del talk
    tags: [String],         // Array di tag associati al talk
}, { collection: 'tedx_data' });

// Creazione e esportazione del modello
module.exports = mongoose.model('Talk', talkSchema);
```

```
const mongoose = require('mongoose');

const watchNextSchema = new mongoose.Schema({
    video_id: String,                       // ID del video principale
    related_videos_id_list: [String],// Lista di ID dei video correlati
    image_url: String,
    url: String,
    tags: [String],                         // Array di tag associati al video
}, { collection: 'watch_next' });

module.exports = mongoose.model('WatchNext', watchNextSchema);
```

Abbiamo definito anche uno schema **watchNextSchema** con determinati campi per definire la struttura della collezione watch_next

# Get_Watch_Next_by_Idx

## Funzione  (handler.js)

Definizione di alcune costanti per richiamare gli schemi e il file di connessione al db

```
const mongoose = require('mongoose');
const connect_to_db = require('./db');
const Talk = require('./Talk'); // Modello per la collezione tedx_data
const WatchNext = require('./WatchNext'); // Modello per la collezione watch_next
```

```
// Mappatura dei tag con i loro valori
const tagValueMapping = {
    "society": 10,
    "politics": 9,
    "economics": 8,
    "health": 6,
    "sustainability": 8
};
```

Abbiamo definito un oggetto **tagvalueMapping** che associa ai tag di nostro interesse dei valori, in base alla rilevanza

# Funzione Lambda

- **module.exports.get_talk_by_id**: funzione principale esportata che viene chiamata da AWS Lambda.
- **context.callbackWaitsForEmptyEventLoop = false**: Impostiamo il contesto Lambda per non attendere che l'event loop sia vuoto prima di restituire la risposta.
- Se il corpo dell'evento è presente, viene analizzato come JSON. Verifica se idx è presente e ritorna un errore se non lo è.

```
module.exports.get_talk_by_id = async (event, context, callback) => {
    context.callbackWaitsForEmptyEventLoop = false;

    // Log dell'evento ricevuto
    console.log("Received event:", event);

    // Verifica se l'evento contiene `body` o se `idx` è direttamente nell'evento
    let body = event.body ? JSON.parse(event.body) : event;

    // Log del body parsato
    console.log("Parsed body:", body);

    if (!body.idx) {
        console.error("idx è nullo o non definito. Body ricevuto:", body);
        return callback(null, {
            statusCode: 500,
            headers: { 'Content-Type': 'text/plain' },
            body: 'Could not fetch the talk. idx is null.'
        });
    }
```

# Caricamento dati e risposta

```
try {
    await connect_to_db();

    // Trova il talk nella collezione tedx_data
    const foundTalk = await Talk.findById(body.idx);
    if (!foundTalk) {
        return callback(null, {
            statusCode: 404,
            headers: { 'Content-Type': 'text/plain' },
            body: 'Talk not found.'
        });
    }

    // Prepara la risposta
    const response = {
        _id: foundTalk._id,
        title: foundTalk.title,
        description: foundTalk.description,
        url: foundTalk.url,
        image_url: foundTalk.image_url,
        watch_next: []
    };
```

- **await connect_to_db():** Stabilisce una connessione con il database MongoDB.
- **Recupero del talk:** Usiamo Talk.findById(body.idx) per cercare un documento nella collezione tedx_data con l'ID fornito.

# Recupero e ordinamento

- **Recupero dei dati di WatchNext:** Troviamo i documenti correlati usando WatchNext.findOne({ video_id: body.idx }).
- **Verifica degli ID**: Se ci sono video correlati, utilizza gli ID per cercare i documenti nella collezione tedx_data.
- **Calcoliamo next_video_count**: Per ogni video correlato, calcoliamo un punteggio basato sui tag associati e sui valori definiti in tagValueMapping.

```
// Trova i video correlati usando la collezione watch_next
const watchNextData = await WatchNext.findOne({ video_id: body.idx });
if (watchNextData) {

    console.log("WatchNext data found:", watchNextData);

    // Assicurati che related_videos_id_list contenga gli ID corretti
    console.log("Related video IDs:", watchNextData.related_videos_id_list);

    if (watchNextData.related_videos_id_list.length === 0) {
        console.log("related_videos_id_list is empty.");
    } else {

        // Usa gli ID come stringhe per la query
        const relatedVideoIds = watchNextData.related_videos_id_list;
        console.log("Using string IDs for query:", relatedVideoIds);

        // Verifica se gli ID nella query sono nel formato corretto
        const relatedVideos = await Talk.find({ '_id': { $in: relatedVideoIds } });
        console.log("Related videos found:", relatedVideos);

        // Calcola next_video_count per ciascun video correlato utilizzando i tag da watch_next
        const relatedVideosWithCount = await Promise.all(relatedVideos.map(async video => {
            // Trova i tag dal record corrispondente in watch_next
            const relatedVideoTagsData = await WatchNext.findOne({ video_id: video._id });
            let relatedVideoCount = 0;

            if (relatedVideoTagsData && relatedVideoTagsData.tags) {
                relatedVideoTagsData.tags.forEach(tag => {
                    if (tagValueMapping.hasOwnProperty(tag)) {
                        relatedVideoCount += tagValueMapping[tag];
```

# Recupero e ordinamento

```
    return {
        _id: video._id,
        title: video.title,
        url: video.url,
        image_url: video.image_url,
        next_video_count: relatedVideoCount // Include il valore next_video_count
    };
}));


                // Ordina i video correlati in modo decrescente per next_video_count
                relatedVideosWithCount.sort((a, b) => b.next_video_count - a.next_video_count);

                // Aggiorna la risposta con i video correlati ordinati e con next_video_count incluso
                response.watch_next = relatedVideosWithCount.map(video => ({
                    _id: video._id,
                    title: video.title,
                    url: video.url,
                    image_url: video.image_url,
                    next_video_count: video.next_video_count // Includi next_video_count nell'array watch_next
                }));
            }
        } else {
            console.log("No WatchNext data found for video_id:", body.idx);
        }
```

- **Ordinamento dei video:** Ordiniamo i video correlati in base al next_video_count in ordine decrescente.
- **Aggiornamento della risposta:** Aggiungiamo i video correlati ordinati alla risposta finale.

# Gestione degli errori e risposta

- **Risposta di Successo**: Se tutto va a buon fine, restituiamo una risposta con stato HTTP 200 e il corpo della risposta in formato JSON.

- **Gestione degli Errori:** Se si verifica un errore, restituisce una risposta con stato HTTP 500 e un messaggio di errore.

```
        return callback(null, {
            statusCode: 200,
            body: JSON.stringify(response)
        });
    } catch (err) {
        console.error('Error:', err);
        return callback(null, {
            statusCode: 500,
            headers: { 'Content-Type': 'text/plain' },
            body: 'An error occurred while processing your request.'
        });
    }
};
```

# Chiamata API

```
{
    "idx" : "297805"
}
```


```
"_id": "297805",
"title": "A personal plea for humanity at the US-Mexico border",
"description": "In this powerful, personal talk, author and academic Juan Enriquez shares stories from inside the immigration
    crisis at the US-Mexico border, bringing this often-abstract debate back down to earth -- and showing what you can do every
    day to create a sense of belonging for immigrants. \"This isn't about kids and borders,\" he says. \"It's about us. This is
    about who we are, who we the people are, as a nation and as individuals.\"",
"url": "https://www.ted.com/talks/juan_enriquez_a_personal_plea_for_humanity_at_the_us_mexico_border",
"watch_next": [
    {
        "_id": "288142",
        "title": "The secret student resistance to Hitler ",
        "url": "https://www.ted.com/talks/iseult_gillespie_the_secret_student_resistance_to_hitler",
        "next_video_count": 19
    },
    {
        "_id": "298355",
        "title": "The psychological impact of child separation at the US-Mexico border",
        "url": "https://www.ted.com/talks/luis_h_zayas_the_psychological_impact_of_child_separation_at_the_us_mexico_border",
        "next_video_count": 19
    },
```

# USER EXPERIENCE

**POSITIVA:**

- **Personalizzazione:** Gli utenti ricevono suggerimenti personalizzati sui video correlati in base ai tag e ai valori associati, il che potrebbe aumentare la loro soddisfazione e coinvolgimento.
- **Velocità di Caricamento:** Se ottimizzato correttamente, il sistema potrebbe offrire suggerimenti rapidi e pertinenti, migliorando l'esperienza complessiva.
- **Feedback Visivo:** Gli utenti vedono informazioni dettagliate sui video suggeriti, inclusi titoli, URL e immagini, che aiuta nella navigazione e nella scelta dei contenuti.

**NEGATIVA:**

- **Errori e Ritardi:** Se ci sono problemi nel recupero dei dati o nel calcolo del punteggio, gli utenti potrebbero vedere messaggi di errore o ritardi nel caricamento delle informazioni sui video correlati.
- **Aggiornamenti dei Dati:** Se la collezione watch_next non è aggiornata, i suggerimenti potrebbero non essere pertinenti o aggiornati, riducendo l'utilità dell'esperienza.
- **Performance:** Il calcolo del punteggio e l'ordinamento dei video correlati potrebbero richiedere tempo, influenzando negativamente la reattività dell'interfaccia utente.

# POSSIBILI CRITICITÀ

**PERFORMANCE DEL DATABASE:**

- **Query complessa**: Le query MongoDB per trovare video correlati e calcolare i punteggi possono essere lente se i dati sono numerosi. L'uso di indici e ottimizzazioni delle query è cruciale.

- **Scalabilità**: Con un aumento del numero di video e tag, la performance potrebbe degradare. È importante monitorare le prestazioni e ottimizzare il database.

**ERROR HANDLING:**

- **Gestione degli errori**: Il codice gestisce gli errori generali, ma potrebbero verificarsi casi specifici non coperti, come la mancanza di dati in formato previsto o errori di rete.

- **Messaggi di Errore**: I messaggi di errore restituiti sono generali e non forniscono dettagli specifici per l'utente finale o per il debugging.

# POSSIBILI IMPLEMENTAZIONI

## AGGIORNAMENTO DEI DATI

Implementare meccanismi per aggiornare periodicamente la collezione watch_next con nuovi dati e rimuovere video non più pertinenti.

## MONITORAGGIO E LOGGING AVANZATI

Aggiungere un sistema di monitoraggio più dettagliato e logging avanzato per identificare e risolvere rapidamente problemi di performance e errori

## INTERNAZIONALIZZAZIONE E LOCALIZZAZIONE

Se l'applicazione è destinata a un pubblico globale, si potrebbe considerare la localizzazione dei contenuti e dei suggerimenti per diverse lingue e culture.

# TEDxNEWS


TRELLO BOARD


GIT HUB