

Cattaneo Luca 1079489  
Locatelli Erica 1081101

B E Y O N D

T E R R I T O R Y



HW3

# GET\_WATCH\_NEXT\_BY\_ID



Abbiamo introdotto una funzionalità “Watch Next” che suggerisce video correlati, implementando un endpoint API GET /talks/{id}/watch-next che restituisce una lista di suggerimenti arricchiti.

Abbiamo scelto un’architettura serverless per garantire efficienza, scalabilità e bassi costi, ciò ci permette di pagare solo per le risorse effettivamente consumate e di scalare automaticamente in base al traffico.

# GET\_WATCH\_NEXT\_BY\_ID

```
try {
  await connect_to_db();
  const main_talk = await Talk.findById(main_talk_id);

  if (!main_talk) {
    console.warn(`ATTENZIONE: Talk con ID ${main_talk_id} non trovato nel database.`);
    return { statusCode: 404, body: JSON.stringify({ message: 'Talk non trovato.' }) };
  }

  const watch_next_ids = Array.isArray(main_talk.WatchNext_id) ? main_talk.WatchNext_id : [];
  let suggested_talks_details = [];
  if (watch_next_ids.length > 0) {
    console.log(`DEBUG: Cerco talk suggeriti con ID: ${JSON.stringify(watch_next_ids)}`);

    // Se ancora non trova, verifica che questi ID esistano davvero nel DB.
    suggested_talks_details = await Talk.find({
      '_id': { $in: watch_next_ids }
    });
    console.log(`INFO: Trovati ${suggested_talks_details.length} suggerimenti tramite WatchNext_id.`);
  }
}
```

## Flusso logico:

1. Riceve l'ID del talk principale dall'URL tramite API Gateway.
2. Si connette a MongoDB in modo sicuro.
3. Trova il talk principale e legge la sua lista di ID suggeriti (il campo WatchNext\_id).
4. Recupera i dettagli completi di tutti i talk suggeriti con una singola query efficiente.
5. Esegue l'analisi per determinare il "perché" del suggerimento.
6. Restituisce una risposta JSON formattata all'API Gateway.

# GET\_WATCH\_NEXT\_BY\_ID: RISULTATI



GET https://glm8oppzm3.execute-api.us-east-1.amazonaws.com/v1/talks/562754/watch-next

```
{  
  "_id": "113826",  
  "title": "How to land on a comet",  
  "url": "https://www.ted.com/talks/fred_jansen_how_to_land_on_a_comet"  
},  
{  
  "_id": "114136",  
  "title": "How Mars might hold the secret to the origin of life",  
  "url": "https://www.ted.com/talks/nathalie_cabrol_how_mars_might_hold_the_secret_to_the_origin_of_life"  
},  
{  
  "_id": "114262",  
  "title": "How I fell in love with quasars, blazars and our incredible universe",  
  "url": "https://www.ted.com/talks/jedidah_isler_how_i_fell_in_love_with_quasars_blazars_and_our_incredible_universe"  
}
```

Postman

# REGISTRAZIONE E LOGIN



La funzione di **login** verifica le credenziali confrontando la password inserita con l'hash salvato e, in caso di successo, genera un token di sessione (JWT) sicuro e temporaneo, che autorizza l'utente a usare l'applicazione.

```
const newUser = new User({  
    username,  
    password: hashedPassword,  
    userType: userType  
});  
  
await newUser.save();  
  
return {  
    statusCode: 201,  
    body: JSON.stringify({ message: "Utente registrato con successo!" })  
};
```

```
await connect_to_db();  
console.log('Ricerca utente nel database...');  
  
const user = await User.findOne({ username: username.toLowerCase() });  
if (!user) {  
    console.log('Tentativo di login fallito: utente non trovato.');//  
    return {  
        statusCode: 401,  
        body: JSON.stringify({ message: "Le credenziali fornite non sono corrette." })  
    };  
}  
  
console.log('Utente trovato. Confronto delle password...');  
const isPasswordCorrect = await bcrypt.compare(password, user.password);  
  
if (!isPasswordCorrect) {  
    console.log('Tentativo di login fallito: password errata.');//  
    return {  
        statusCode: 401,  
        body: JSON.stringify({ message: "Le credenziali fornite non sono corrette." })  
    };  
}
```

La funzione di **registrazione** si occupa di creare nuovi utenti: per garantire la massima sicurezza, la password fornita viene trasformata in un codice criptato e irreversibile (hash) prima di essere salvata nel nostro database.

# REGISTRAZIONE E LOGIN



```
# 3. Lettura del file CSV da S3
print(f'Lettura del file di input: {input_csv_file}')
user_dataset = spark.read \
    .option("header", "true") \
    .option("quote", "") \
    .option("escape", "") \
    .option("delimiter", ",") \
    .csv(input_csv_file)

print("Schema del file CSV letto:")
user_dataset.printSchema()

# 4. Scrittura su MongoDB
print(f'Inizio scrittura sulla collezione '{mongo_collection_name}' in MongoDB...')
write_mongo_options = {
    "connectionName": mongo_connection_name,
    "database": mongo_database_name,
    "collection": mongo_collection_name,
    "ssl": "true",
    "ssl.domain_match": "false"
}
```



## JOB AWS GLUE

Per popolare il Data Warehouse, abbiamo creato un job di AWS Glue che gestisce il caricamento iniziale degli utenti.

Il Job esegue un processo ETL una sola volta:

1. Estrae i dati da un file log.csv presente nel bucket S3.
2. Trasforma i dati in un formato strutturato.
3. Carica in blocco gli utenti nella collezione users del database MongoDB, preparandola per l'uso da parte delle nostre funzioni di login.

# REGISTRAZIONE E LOGIN : RISULTATI



```
_id: ObjectId('68947f2ce9058c3a5722933e')
username : "Boss.fareshi"
password : [REDACTED]
userType : "studente"
```

```
_id: ObjectId('68947f2ce9058c3a5722933f')
username : "Ivan.drago"
password : '[REDACTED]'
userType : "appassionato"
```

MONGO DB  
Collection Users



# ESPERIENZA UTENTE

01

## Accesso Sicuro e Personalizzato:

L'utente entra in un ambiente su misura. Grazie al nostro sistema di login, l'app riconosce il suo ruolo (studente, divulgatore, appassionato) e adatta l'interfaccia e le funzionalità, offrendo un'esperienza pertinente e unica fin dal primo istante.

02

## Scoperta Intelligente dei Contenuti:

Dopo ogni talk, l'API watch-next suggerisce il video successivo. Questo trasforma la fruizione passiva in un percorso di scoperta guidato e consapevole.

03

## Percorsi Formativi Guidati:

L'architettura raggruppa i talk in "Percorsi Formativi" tematici, offrendo allo studente un'esperienza simile a un mini-corso. Questo non solo guida l'apprendimento ma incentiva l'utente a completare un intero argomento, aumentando il coinvolgimento e la conoscenza acquisita.

# CRITICITÀ TECNICHE

## Limiti dell'Ambiente e Adattabilità:

- 01 Il nostro ruolo IAM (LabRole) non aveva i permessi per usare servizi gestiti come Amazon Cognito o Bedrock, bloccando i nostri piani iniziali per l'autenticazione e i quiz.

## Parsing di Dati Eterogenei (AWS Glue)

- 02 Il nostro primo job di Glue leggeva ogni riga del file log.csv come una singola colonna, fallendo nel separare username, password e userType

## Incompatibilità tra Codice e Schema DB:

- 03 Le nostre funzioni Lambda fallivano con un errore Cast to ObjectId nel cercare utenti che esistevano nel database.



# POSSIBILI EVOLUZIONI

01

## Personalizzazione Avanzata con AI:

Evolvere il sistema di suggerimenti utilizzando servizi come Amazon Personalize per analizzare la cronologia di ogni utente e fornire raccomandazioni veramente individuali.

02

## Gamification e Certificazioni:

Implementare la funzionalità di quiz automatici alla fine di ogni "Percorso Formativo".

03

## Dashboard e Strumenti per i Divulgatori:

Creare una sezione dedicata per il divulgatore, dove può visualizzare statistiche dettagliate sui propri talk (visualizzazioni e commenti) e avere un form per proporre nuovi contenuti.

04

## Creazione di una Community di Apprendimento:

Introdurre funzionalità social. Gli utenti potrebbero commentare i talk, creare gruppi di studio tematici legati ai percorsi formativi e seguire i loro divulgatori preferiti.

