Project Made by:

1080572 - Arnoldi Elisa
1081373 -Merli Gabriele

TspaCEDx

Explore Some New Content about Space !

powered by: TEDx

INTRODUZIONE e IDEE

TspaceEDx

IDEA:

TSpaceDx è il nostro impegno per diffondere la cultura spaziale e creare una comunità di persone interessate all'universo.

OBBIETTIVO:

La nostra missione è ispirare e educare mentre la nostra visione è un mondo dove la conoscenza dello spazio è alla portata di tutti.

PROBLEMA

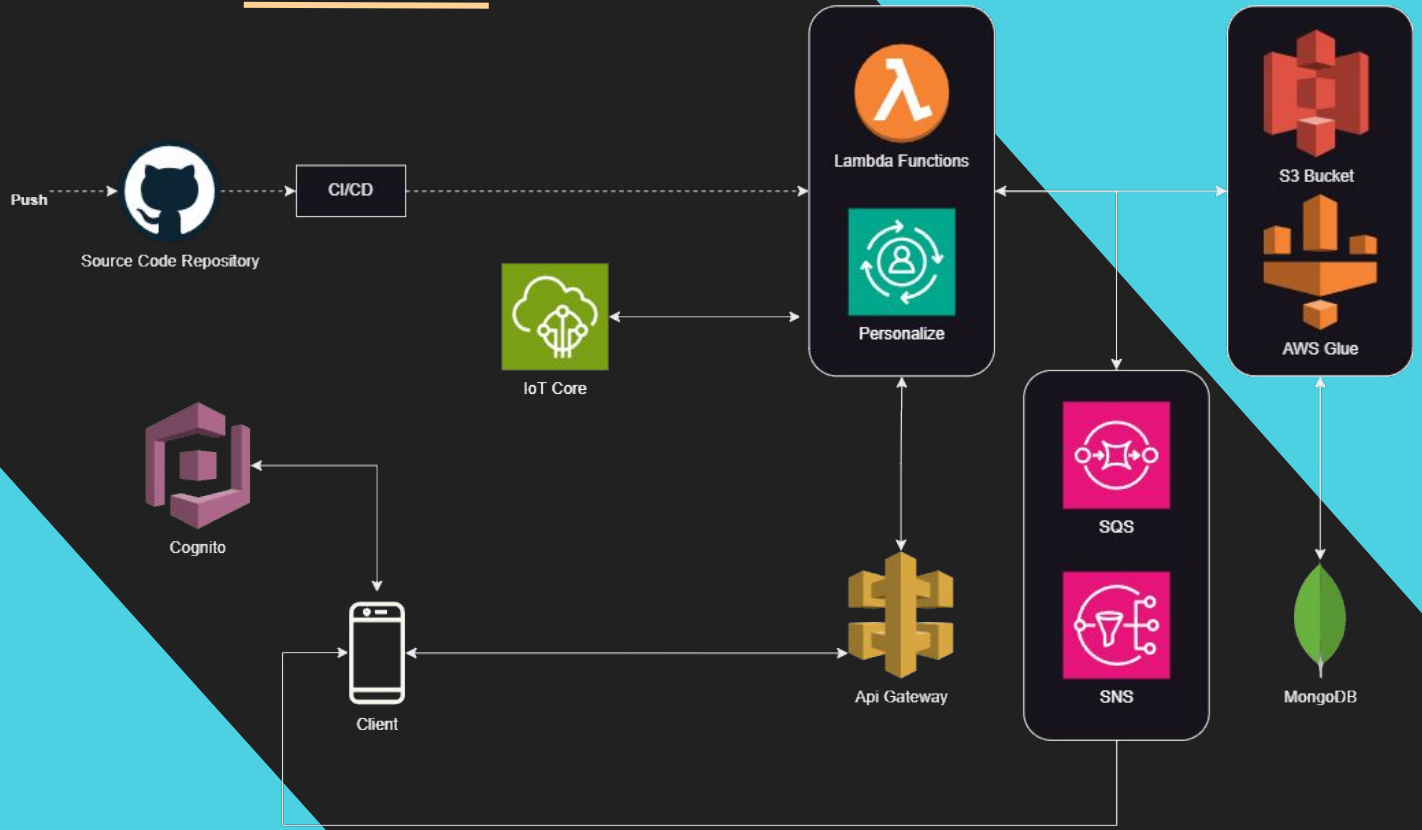
- Molti trovano difficile accedere a informazioni aggiornate e affidabili sullo spazio.

- Esistono poche piattaforme dedicate esclusivamente alla divulgazione scientifica spaziale e ancora meno che permettono la condivisione di esperienze e conoscenze tra appassionati.

SOLUZIONE

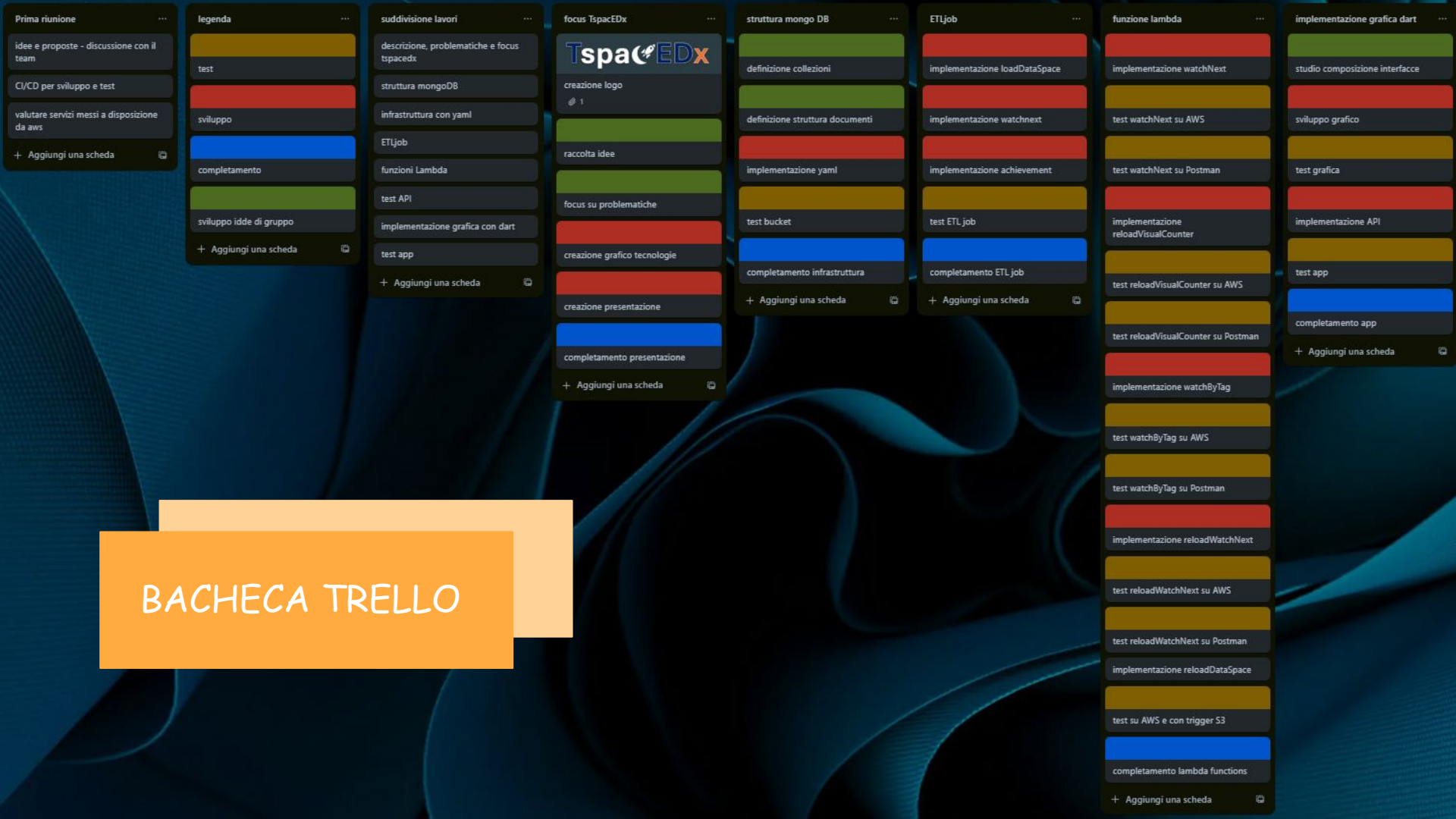
- TSpaceDx è la piattaforma ideale per chi vuole esplorare il mondo dello spazio.
- Offriamo notizie aggiornate, articoli dettagliati e forum per discussioni.
- Con TSpaceDx, gli utenti possono non solo apprendere, ma anche contribuire attivamente alla comunità.

TECNOLOGIE


Push
Source Code Repository
CI/CD
Lambda Functions
Personalize
S3 Bucket
AWS Glue
IoT Core
Cognito
Client
Api Gateway
SQS
SNS
MongoDB

CRITICITA'

- Moderare e gestire una comunità online può essere complesso, soprattutto con l'aumento degli utenti.
- Minimizzare la latenza e garantire l'alta disponibilità dei servizi è fondamentale per una buona esperienza utente.
- Gestire intelligentemente la riproduzione di video dando priorità ai più pertinenti e ricercati.



BACHECA TRELLO

TspacEDx

- Link git

  URL:https://github.com/Merluz/TspacEDx

- Link board trello

  URL:https://trello.com/b/oiasbz5H

AWS glue - REALIZZAZIONE

TspaceEDx

PySpark Job loadDataSpace

Il codice esegue legge diversi dataset CSV da un bucket S3, li filtra, li unisce e li trasforma utilizzando Spark.

I dataset contengono informazioni su conferenze TEDx, dettagli, immagini e tag. Dopo aver integrato e filtrato i dati il risultato viene scritto in una collezione MongoDB per un utilizzo successivo.

1. Inizializzazione: Configura il contesto Spark e Glue e avvia il lavoro.
2. Lettura dei dataset: Carica i file CSV da S3 in data frame Spark.
3. Unione e trasformazione: Integra i dataset unendoli su chiavi comuni e applica filtri per includere solo i dati pertinenti.
4. Scrittura dei risultati: Salva il dataset finale filtrato in una collezione MongoDB utilizzando AWS Glue.

```
# READ IMAGES DATASET
images_dataset = spark.read \
    .option("header", "true") \
    .option("quote", "\"") \
    .option("escape", "\"") \
    .csv(images_dataset_path) \
    .select(F.col("id").alias("id_ref"),
            F.col("url").alias("image_url"))

# JOIN WITH TEDX DATASET
tedx_dataset_main = tedx_dataset_main.join(images_dataset, tedx_dataset_main.id == images_dataset.id_ref, "left") \
    .drop("id_ref")

# READ TAGS DATASET AND FILTER FOR "space"
tags_dataset = spark.read.option("header", "true").csv(tags_dataset_path)
tags_dataset_filtered = tags_dataset.filter(F.col("tag") == "space")

# AGGREGATE MODEL, ADD TAGS TO TEDX_DATASET
tags_dataset_agg = tags_dataset_filtered.groupBy(F.col("id").alias("id_ref")).agg(F.collect_list("tag").alias("tags"))

# JOIN TAGS WITH TEDX DATASET
tedx_dataset_agg = tedx_dataset_main.join(tags_dataset_agg, tedx_dataset_main.id == tags_dataset_agg.id_ref, "left") \
    .drop("id_ref") \
    .select(F.col("id").alias("_id"),
            F.col("slug"),
            F.col("speakers"),
            F.col("title"),
            F.col("url"),
            F.col("description"),
            F.col("duration"),
            F.col("publishedAt"),
            F.col("image_url"),
            F.col("tags"))

# AGGREGATE ALL TAGS FOR EACH ITEM
all_tags_dataset = tags_dataset.groupBy(F.col("id").alias("id_ref")).agg(F.collect_list("tag").alias("all_tags"))

# JOIN ALL TAGS WITH TEDX DATASET
tedx_dataset_agg_final = tedx_dataset_agg.join(all_tags_dataset, tedx_dataset_agg._id == all_tags_dataset.id_ref, "left") \
    .drop("id_ref") \
    .select(F.col("_id"),
            F.col("slug"),
            F.col("speakers"),
            F.col("title"),
            F.col("url"),
            F.col("description"),
            F.col("duration"),
            F.col("publishedAt"),
            F.col("image_url"),
            F.col("all_tags").alias("tags"))
```

```python
# INIZIALIZZA IL JOB CONTEXT E IL JOB
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

# OPZIONI DI CONNESSIONE MONGODB PER LA LETTURA
read_mongo_options = {
    "connectionName": "TEDX2024",
    "database": "tspacedx",
    "collection": "tspacedx_data",
    "ssl": "true",
    "ssl.domain_match": "false"
}

# LEGGE I DATI DA MONGODB
tedx_dataset = glueContext.create_dynamic_frame.from_options(
    connection_type="mongodb",
    connection_options=read_mongo_options
).toDF()

# FUNZIONE PER TROVARE VIDEO CON TAG SIMILI
def find_similar_videos(df):
    exploded_df = df.withColumn("tag", F.explode("tags"))

    # Self-join per trovare tutti i video con almeno un tag in comune diverso da se stessi
    joined_df = exploded_df.alias("df1").join(
        exploded_df.alias("df2"),
        (F.col("df1.tag") == F.col("df2.tag")) & (F.col("df1._id") != F.col("df2._id")),
        "inner"
    ).select(
        F.col("df1._id").alias("current_id"),
        F.col("df2._id").alias("next_id")
    ).distinct()

    return joined_df

# TROVA I VIDEO SIMILI
similar_videos_df = find_similar_videos(tedx_dataset)
```

PySpark Job watch_next

Questo script in Python utilizza AWS Glue e Apache Spark per leggere dati da una collezione MongoDB, identificare video con tag simili, e determinare i video suggeriti con priorità basata sul conteggio delle visualizzazioni successive e pertinenza dei tag.

I risultati vengono poi scritti in una nuova collezione MongoDB.

PySpark Job watch_next

1. Inizializzazione del Contesto e Job
2. Lettura dei Dati da MongoDB
3. Identificazione dei Video Simili
4. Lettura dei Dati di Achievement
5. Determinazione dei Video Prioritari
6. Scrittura dei Risultati in MongoDB

```
# Unisce similar_videos_df con achievement_df per ottenere next_video_count
joined_df = similar_videos_df.join(
    achievement_df,
    similar_videos_df["next_id"] == achievement_df["_id"],
    "left"
).select(
    similar_videos_df["current_id"],
    similar_videos_df["next_id"],
    achievement_df["next_video_count"]
)

# Determina il next_id con la priorità più alta per ogni current_id basato su next_video_count
window_spec = Window.partitionBy("current_id").orderBy(F.desc("next_video_count"))
prioritized_df = joined_df.withColumn("rank", F.row_number().over(window_spec)).where(F.col("rank") == 1).drop("rank")

# Scrive prioritized_df nella collezione MongoDB watch_next_data
write_mongo_options = {
    "connectionName": "TEDX2024",
    "database": "tspacedx",
    "collection": "watch_next_data",
    "ssl": "true",
    "ssl.domain_match": "false"
}

prioritized_dynamic_frame = DynamicFrame.fromDF(prioritized_df, glueContext, "nested")

glueContext.write_dynamic_frame.from_options(
    frame=prioritized_dynamic_frame,
    connection_type="mongodb",
    connection_options=write_mongo_options
)
```

mongoDB atlas

Esempio di visualizzazione di documento nella collection

watch_next_data


```
_id: ObjectId('667ece882d537247543504c4')
current_id : "489187"
next_id : "507997"

_id: ObjectId('667ece882d537247543504c7')
current_id : "459028"
next_id : "507997"

_id: ObjectId('667ece882d537247543504d8')
current_id : "347139"
next_id : "430928"
```


tspacedx_data


```
_id: "526916"
slug : "brian_crim_the_nazis_recruited_to_win_the_cold_war"
speakers : "Brian Crim"
title : "The Nazis recruited to win the Cold War"
url : "https://www.ted.com/talks/brian_crim_the_nazis_recruited_to_win_the_co…"
description : "In May of 1945 the Third Reich was in chaos. Adolf Hitler was dead and…"
duration : "385"
publishedAt : "2024-04-16T15:03:18Z"
image_url : "https://talkstar-assets.s3.amazonaws.com/production/talks/talk_128548/…"
▾ tags : Array (10)
  0: "science"
  1: "engineering"
  2: "rocket science"
  3: "education"
  4: "history"
  5: "war"
  6: "United States"
  7: "space"
  8: "TED-Ed"
```

PySpark Job achievement

```python
# Carica i dati da MongoDB in un DynamicFrame
tedx_dynamic_frame = glueContext.create_dynamic_frame.from_options(
    connection_type="mongodb",
    connection_options=read_mongo_options
)

# Converte DynamicFrame in DataFrame
tedx_df = tedx_dynamic_frame.toDF()

# Estrae i tag e aggiungili come una nuova colonna
tedx_df = tedx_df.withColumn("tags", F.col("tags"))

# Definisce una lista di tuple tag-valore per accumulare il conteggio
tag_value_mapping = [
    ("space", 10),
    ("astronomy", 7),
    ("Planets", 7),
    ("aliens", 5),
    ("science", 5),
    ("technology", 5),
    ("future", 7)
]

# Calcola next_video_count basato sui tag
tedx_df = tedx_df.withColumn("next_video_count",
    sum([F.when(F.array_contains("tags", tag), value).otherwise(0) for tag, value in tag_value_mapping])
)

# Seleziona solo le colonne necessarie per achievement_df
achievement_df = tedx_df.select("_id", "next_video_count") \
    .withColumn("achievement", F.lit("Watched and moved to another video")) \
    .withColumn("date", F.current_timestamp())
```

Questo script legge i dati da una collezione MongoDB tspacedx_data e calcola un valore di next_video_count per ogni video in base ai tag associati.

Utilizzando una mappatura predefinita di tag e valori, il codice assegna un punteggio a ciascun video. quindi vengono scritti in una collezione MongoDB tspacedx_achievement.

I valori associati ai tag sono strutturati per essere modificati semplicemente in base alle tendenze.

mongoDB atlas

watch_next_data

Esempio di visualizzazione di documento nella collection


```
_id: "97264"
next_video_count : 15
achievement : "Watched and moved to another video"
date : 2024-06-29T14:40:53.060+00:00

_id: "483716"
next_video_count : 27
achievement : "Watched and moved to another video"
date : 2024-06-29T14:40:53.060+00:00

_id: "391653"
next_video_count : 15
achievement : "Watched and moved to another video"
date : 2024-06-29T14:40:53.060+00:00
```


next_video_count è il nome della variabile con la priorità del talk

IMPLEMENTAZIONE JOB

1. Analizza il trend dei Talk e restituisce il Talk successivo più rilevante sulla base dei tag e delle visualizzazione degli utenti.
2. Facile modifica valori di rilevanza associati ai tag..
3. La logica di assegnazione della rilevanza tende ad essere piu efficiente con la presenza sostanziosa di utenti attivi.

CRITICITA'

- Generazione dei tag, impossibilità di cercare con parole che non siano tag già presenti.
- Possibilità che i Talks più rilevanti restino tali per troppo tempo.
- Velocità di caricamento

TspacEDx

- Link git

  URL:https://github.com/Merluz/TspacEDx

- Link board trello

  URL:https://trello.com/b/oiasbz5H

LAMBDA function - REALIZZAZIONE

TspaceEDx

LAMBDA - watchNext

La funzione watchNext ha il compito, dato il l'id del video corrente, di trovare il talk successivo in base al trend e ottenere tutti i dati relativi a quel talk

Chiamata API

```
{
    "current_id": "519572"
}
```

Headers (7)

Pretty Raw Preview JSON


```
{
    "_id": "304720",
    "slug": "mike_brown_the_search_for_our_solar_system_s_ninth_planet",
    "speakers": "Mike Brown",
    "title": "The search for our solar system's ninth planet",
    "url": "https://www.ted.com/talks/mike_brown_the_search_for_our_solar_system_s_ninth_planet",
    "description": "Could the strange orbits of small, distant objects in our solar system lead us to a big discovery? Planetary astronomer M:
        lurking in the far reaches of our solar system -- and shows us how traces of its presence might already be staring us in the face.",
    "duration": "822",
    "publishedAt": "2019-11-22T16:00:40Z",
    "image_url": "https://talkstar-photos.s3.amazonaws.com/uploads/955d4cc0-2147-4c7a-991d-928306470fd2/MikeBrown_2019S-embed.jpg",
    "tags": [
        "science",
        "astronomy",
        "universe",
        "Planets",
        "space",
        "discovery",
        "solar system",
        "asteroid"
    ]
}
```


```
try {
    // Connessione al database
    await connect_to_db();

    const WatchNextData = mongoose.connection.db.collection('watch_next_data');
    const TspacedxData = mongoose.connection.db.collection('tspacedx_data');

    // Trova il documento con il next_id corrispondente al current_id fornito
    const watchNextData = await WatchNextData.findOne({ current_id: current_id });

    if (!watchNextData) {
        return {
            statusCode: 404,
            headers,
            body: JSON.stringify({ message: 'next_id non trovato' }),
        };
    }

    const next_id = watchNextData.next_id;

    // Trova il documento nella collezione tspacedx_data con l'_id corrispondente al next_id
    const tspacedxData = await TspacedxData.findOne({ _id: next_id });

    if (!tspacedxData) {
        return {
            statusCode: 404,
            headers,
            body: JSON.stringify({ message: 'Documento non trovato' }),
        };
    }

    return {
        statusCode: 200,
        headers,
        body: JSON.stringify(tspacedxData),
    };
} catch (error) {
    return {
        statusCode: 500,
        headers,
        body: JSON.stringify({ message: 'Errore del server', error: error.message }),
    };
```

LAMBDA - reloadVisualCounter

La funzione reloadVisualCounter ha il compito di incrementare la priorità di ogni tag in base alle visualizzazioni

```
async function updateNextVideoCount(_id) {
  try {
    let achievementDoc = await AchievementModel.findOne({ _id });

    if (!achievementDoc) {
      achievementDoc = new AchievementModel({ _id });
    }

    achievementDoc.next_video_count += 1;

    // Check if next_video_count is greater than 99
    if (achievementDoc.next_video_count > 99 && !achievementDoc.vip) {
      achievementDoc.vip = true;
    }

    await achievementDoc.save();
  } catch (error) {
    console.error('Error updating next_video_count:', error);
    throw error;
  }
}
```

```
exports.lambda_handler = async (event, context) => {
  try {
    console.log('Received event:', JSON.stringify(event));

    // Connessione a MongoDB prima di eseguire la logica della Lambda
    await connectToDB();

    const _id = event['_id'];
    if (!_id) {
      throw new Error('Missing _id in event');
    }

    // Aggiorna next_video_count nel documento MongoDB
    await updateNextVideoCount(_id);

    // Recupera il documento aggiornato dal database
    const updatedDoc = await AchievementModel.findOne({ _id });

    if (!updatedDoc) {
      throw new Error(`Failed to find document for _id: ${_id}`);
    }

    return {
      statusCode: 200,
      body: JSON.stringify({ next_video_count: updatedDoc.next_video_count })
    };
  } catch (error) {
    console.error('Error processing request:', error);
    return {
      statusCode: 500,
      body: JSON.stringify({ message: `Error processing request: ${error.message}` })
    };
  }
};
```

Chiamata API

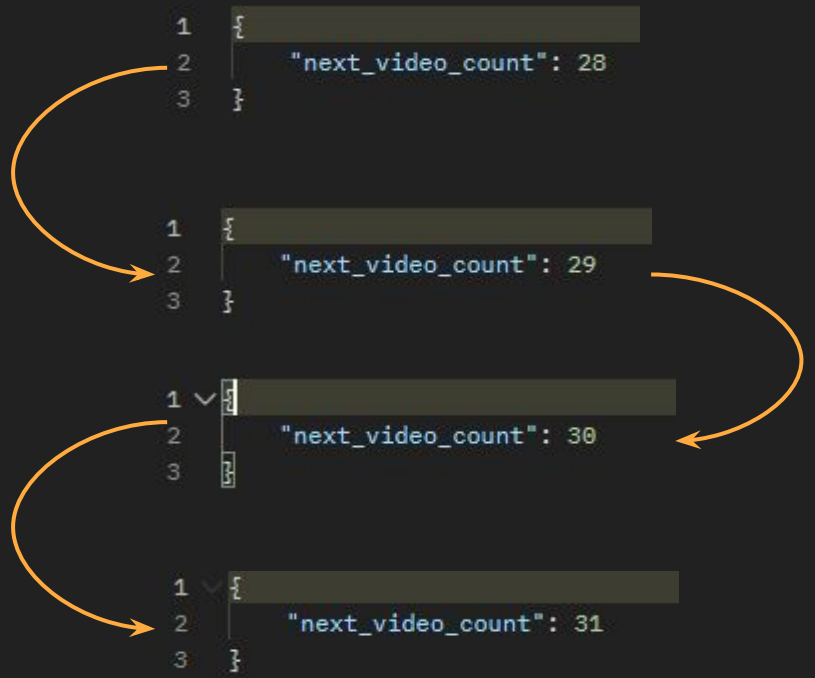
```
{
  "_id": "214565"
}
```

dy Cookies Headers (7) Test Results

Pretty Raw Preview Visualize

```
{
  "next_video_count": 27
}
```

Aumento ad ogni richiamo



next_video_count è il nome della variabile con la priorità del talk

LAMBDA - watchByTag

La funzione watchByTag ha il compito, dato un tag, di trovare i talks associati a quel tag

```
try {
    console.log('Received event:', JSON.stringify(event, null, 2));

    let body = {};
    if (event.body) {
        body = JSON.parse(event.body);
    }

    // Controllo sul tag
    if (!body.tag) {
        return {
            statusCode: 400,
            headers: { 'Content-Type': 'text/plain' },
            body: 'Tag parameter is missing.'
        };
    }

    // Impostazioni predefinite per il numero di documenti per pagina e pagina
    body.doc_per_page = body.doc_per_page || 10;
    body.page = body.page || 1;

    // Connessione al database
    await connect_to_db();

    // Query per trovare i talk con il tag specificato
    const talks = await talk.find({ tags: body.tag })
        .skip((body.doc_per_page * body.page) - body.doc_per_page)
        .limit(body.doc_per_page);

    // Ritorna i risultati
    return {
        statusCode: 200,
        headers: {
            'Content-Type': 'application/json',
            'Access-Control-Allow-Origin': '*', // Permette l'accesso da qualsiasi origine
            'Access-Control-Allow-Methods': 'OPTIONS,POST,GET', // Metodi consentiti
            'Access-Control-Allow-Headers': 'Content-Type', // Header consentiti
        },
        body: JSON.stringify(talks),
    };
```

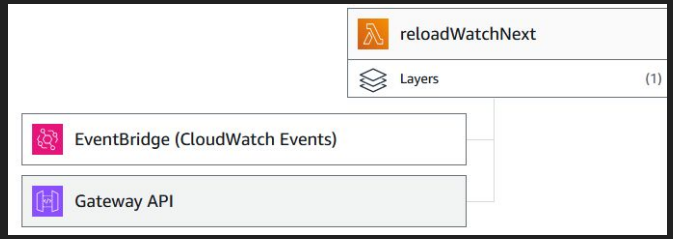
CHIAMATA API

```
{
    "tag": "war"
}
```

la risposta della API è una lista di documenti

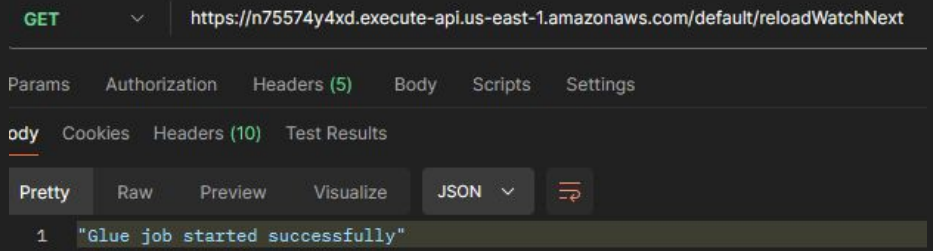

```
[
    {
        "slug": "brian_crim_the_nazis_recruited_to_win_the_cold_war",
        "speakers": "Brian Crim",
        "title": "The Nazis recruited to win the Cold War",
        "url": "https://www.ted.com/talks/brian_crim_the_nazis_recruited_to_win_the_cold_war",
        "description": "In May of 1945 the Third Reich was in chaos. Adolf Hitler was dead and German surrende
            was eager to recruit the smartest minds in Germany before the Soviets got the chance— regardless c
            digs into the clandestine campaign. [Directed by Jeff Le Bars, JetPropulsion.space, narrated by Ac
        "duration": "385",
        "publishedAt": "2024-04-16T15:03:18Z",
        "image_url": "https://talkstar-assets.s3.amazonaws.com/production/talks/talk_128548/0d141979-7ce8-433
        "tags": [
            "science",
            "engineering",
            "rocket science",
            "education",
            "history",
            "war",
            "United States",
            "space",
            "TED-Ed",
            "animation"
        ]
    },
    {
        "slug": "david_hoffman_sputnik_mania",
        "speakers": "David Hoffman",
        "title": "Sputnik mania",
        "url": "https://www.ted.com/talks/david_hoffman_sputnik_mania",
        "description": "Filmmaker David Hoffman shares footage from his feature-length documentary Sputnik Mar
            race and the arms race -- and jump-started science and math education around the world.",
        "duration": "210"
```

LAMBDA - reloadWatchNext



La funzione reloadWatchNext ha il compito di richiamare il job watch_next per ricaricare i talks successivi in base alle nuove priorità associate ai talks.

può essere avviata tramite API oppure in automatico ogni quanto di tempo.




```python
glueJobName = "watch_next"


def lambda_handler(event, context):
    try:
        response = client.start_job_run(JobName=glueJobName)
        logger.info('## STARTED GLUE JOB: ' + glueJobName)
        logger.info('## GLUE JOB RUN ID: ' + response['JobRunId'])

        #intestazioni CORS
        headers = {
            'Access-Control-Allow-Origin': '*',
            'Access-Control-Allow-Methods': 'OPTIONS,POST,GET',
            'Access-Control-Allow-Headers': 'Content-Type',
        }

        return {
            'statusCode': 200,
            'headers': headers,
            'body': json.dumps('Glue job started successfully')
        }
```

LAMBDA - reloadDataSpace

La funzione reloadDataSpace ha il compito di richiamare il job loadDataSpace per ricaricare la collezione "tspacedx_data" dopo ogni volta che viene aggiornato il bucket S3 con nuovi dati.

```python
import json
import os
import logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)

import boto3
client = boto3.client('glue')

glueJobName = "loadDataSpace"

def handler(event, context):

    response = client.start_job_run(JobName = glueJobName)
    logger.info('## STARTED GLUE JOB: ' + glueJobName)
    logger.info('## GLUE JOB RUN ID: ' + response['JobRunId'])
    return response
```

IMPLEMENTAZIONE JOB

- Tempo di esecuzione dei job glue elevato, ricarica dei video prioritari lenta

- SOLUZIONE:
  Sfruttare Amazon S3 Transfer Acceleration per accelerare il trasferimento dei video prioritari in S3. Questo servizio ottimizza automaticamente la velocità di trasferimento utilizzando una rete di distribuzione globale di Amazon CloudFront, riducendo i tempi di latenza e migliorando l'accesso ai dati

TspacEDx

- Link git

  URL:https://github.com/Merluz/TspacEDx

- Link board trello

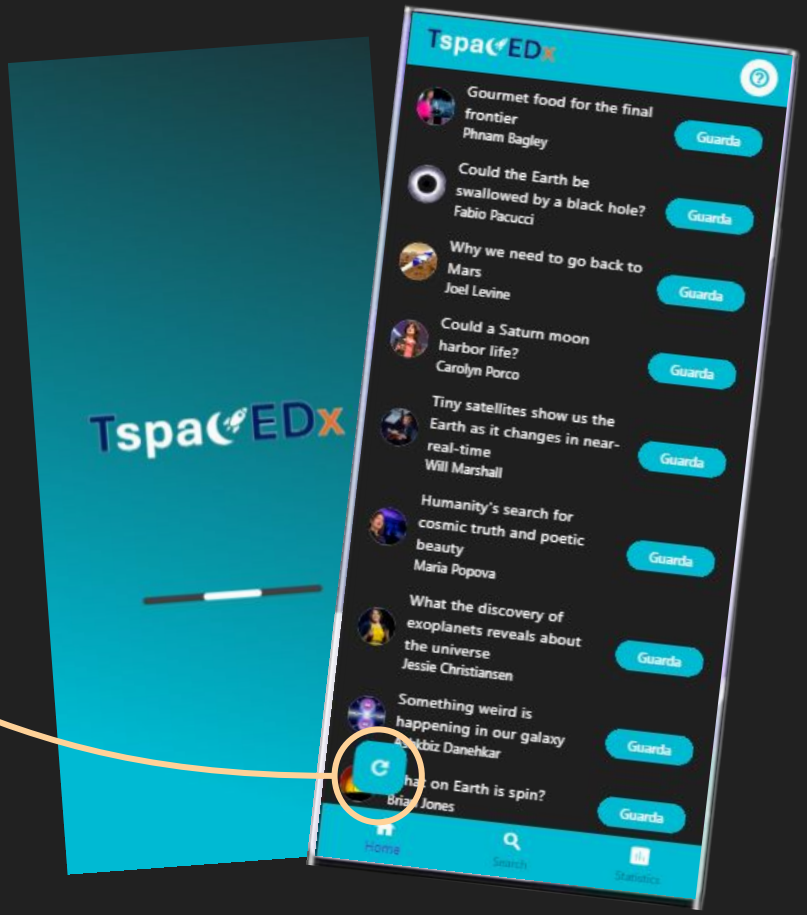
  URL:https://trello.com/b/oiasbz5H

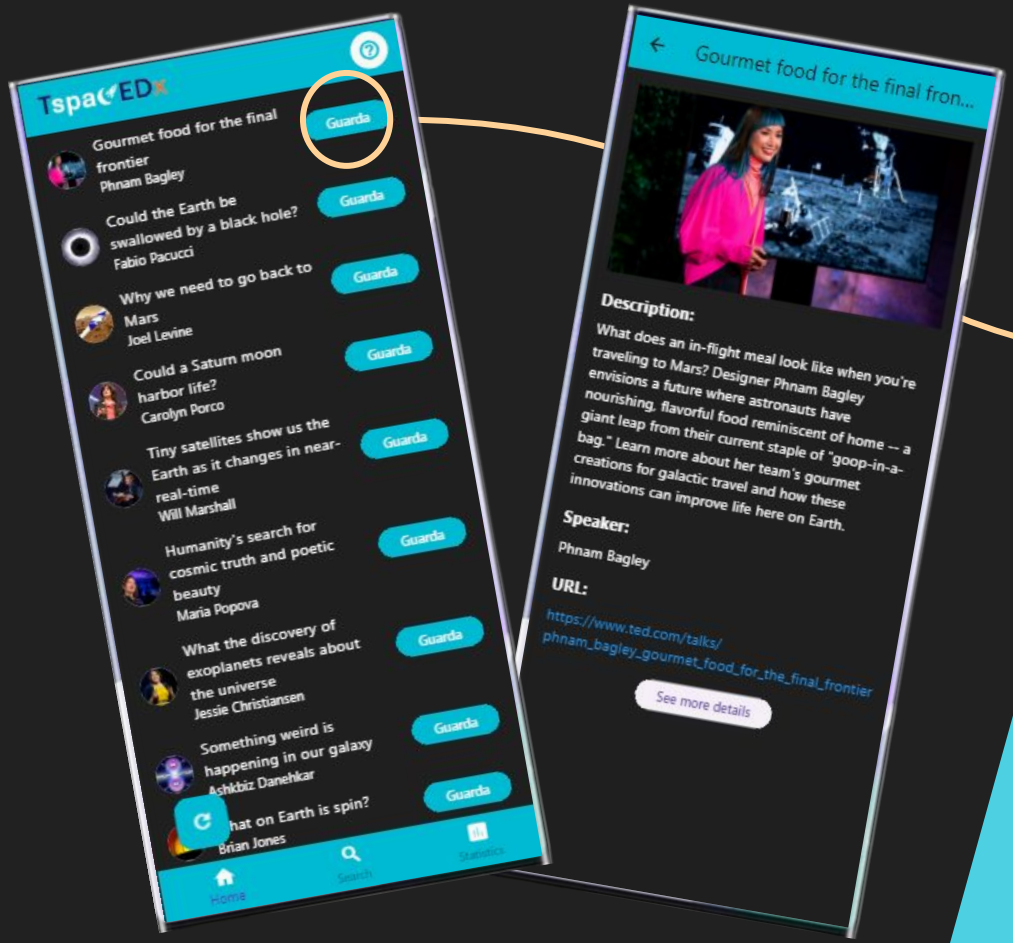
Applicazione - GRAFICA

TspaCEDx

HOMEPAGE

Appena carica l'applicazione, ci si trova nella homepage, si possono già vedere alcuni video consigliati o caricarne altri tramite il pulsante, oppure navigare nelle altre sezioni



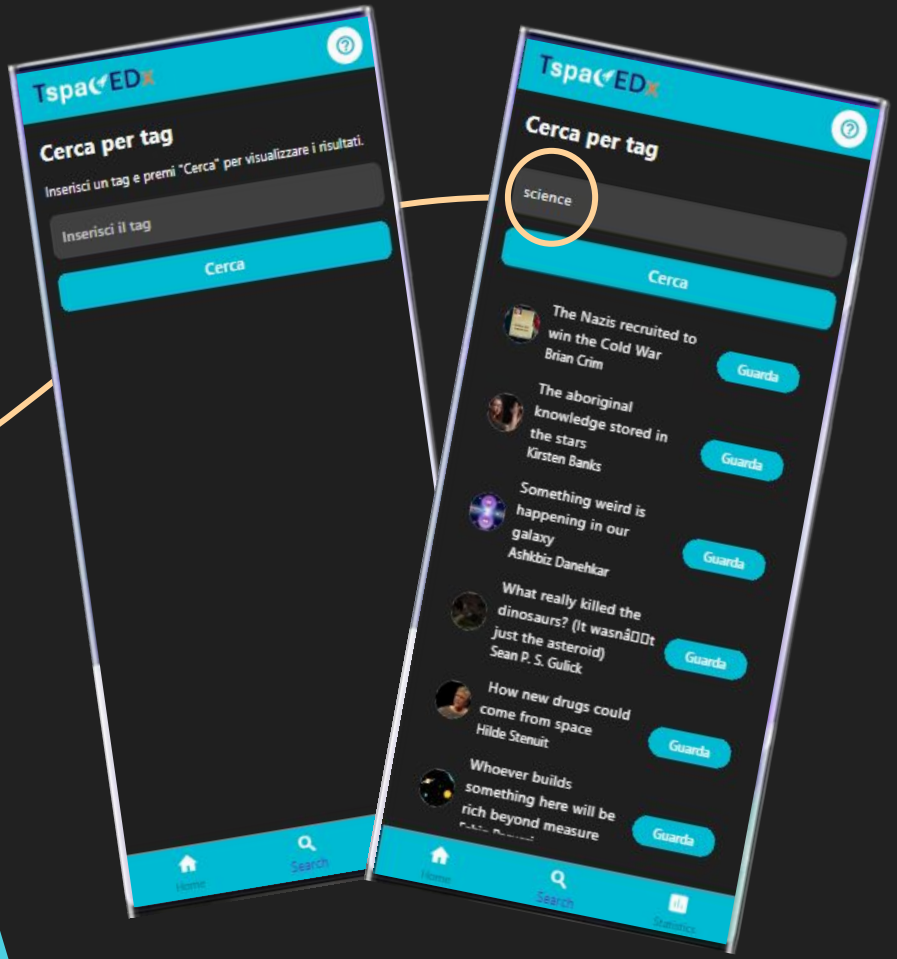


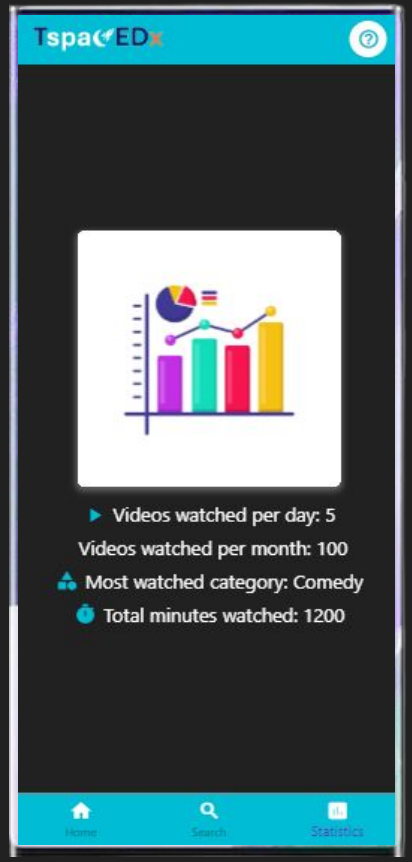
VIDEO-PAGE

Selezionando uno dei vari video, tramite il pulsante "Guarda", si apre la pagina del video con immagine di anteprima, descrizione del video, lo speaker e il link del video in questione

SEARCH-PAGE

Se i video caricati non sono quelli di proprio interesse si può cercare tramite "Tag", ad esempio cercando "science", ricercando solo video inerenti. La pagina si raggiunge con l barra di navigazione sottostante



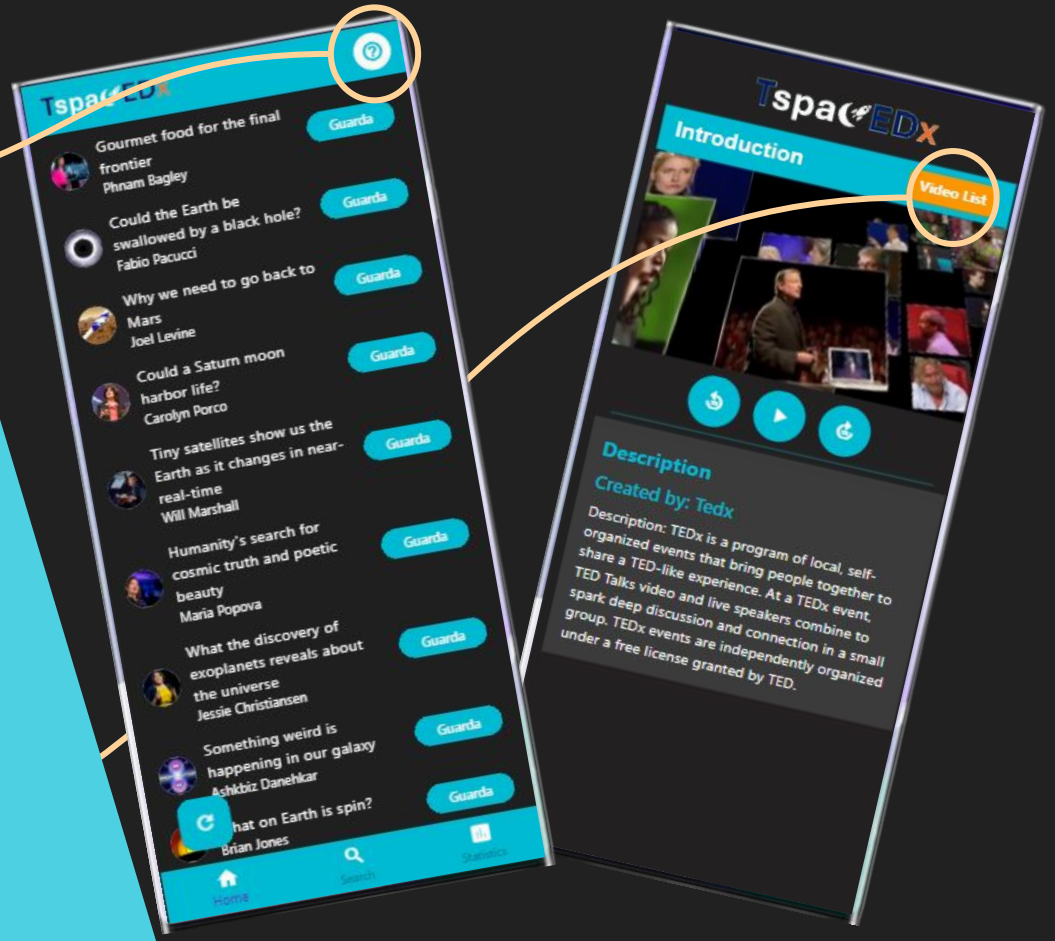


STATISTIC-PAGE

Nella barra di navigazione sottostante è possibile accedere alla pagina delle statistiche, dove puoi vedere i minuti di riproduzione e la categoria più guardata

VIDEO PLAYER

In alto, il bottone "help" che porta al video introduttivo su TEDx, con tanto di descrizione. Il video è riproducibile sull'applicazione ed è dotato di controller: pausa/riprendi e pulsanti che ti permettono di andare leggermente più avanti o indietro nel video. Per tornare indietro si usa il bottone "video list"



Explore Some New Content about Space !