Instituto Tecnológico de Costa Rica

IC4302 - Bases de Datos II

Documentación Proyecto 2

Profesor: Nereo Campos Araya

Estudiantes:

- Fiorella Zelaya Coto - 2021453615
- Isaac Araya Solano - 2018151703
- Melany Salas Fernández - 2021121147
- Moisés Solano Espinoza - 2021144322
- Pablo Arias Navarro - 2021024635

Instrucciones para ejecutar su proyecto

Componentes

Loader

Parse Artists.csv

Se define la función parseArtists para hacer la lectura y el parseo de los artistas de los archivos de artistas.

```
def parseArtists(artistDownloaded_var):
    try:
        client = MongoClient(uri)  #Mongo Client
        client = MongoClient(uri, server_api=ServerApi('1'))
        #Prueba conexión con Mongo DB
        client.admin.command('ping')
        print("Successfully connected to MongoDB")
```

Lo primero que se hace en la función es abrir la conexión de Mongo DB, se envía un ping para comprobar que la conexión es correcta.

```
db = client[str(DatabaseName)]  #Database to be use
collection = db[str(ArtistsCollection)] #Database to be use
```

Después, definimos la base de datos y la collection que se va a usar para cargar/bajar datos a Mongo Atlas.

```
#csv_reader to parse the csv file
csv_reader = csv.reader(artistDownloaded_var, delimiter=',')
header = next(csv_reader)  #skip the header
```

También, se define un csv reader para hacer la lectura y parseo del csv de artistas, además, se deffine el delimitador por el cual se separan los campos y se hace un skip de la fila del header.

```
documents  = []   #list of documents to be inserted
doc = {}          #document to be inserted in the list of documents

artistsNames = collection.distinct("artist")  #To verify if the artist is already in the database by name
max = 0
```

Posteriormente se define una lista para almacenar los documentos que serán insertados en la collection de Mongo y un documento que almacena la información del artista que se está leyendo actualmente. También existe la variable **artistsNames** para obtener los nombres de los artistas que existen actualmente en Mongo, esto se usa para hacer la verificación de los artistas que ya han sido agregados a la collection. Se define un max en caso de que se desee limitar la cantidad de artistas que se van a subir a la collection.

```
for row in csv_reader:
    if max == 100:
        break
    if not row[0] in artistsNames:
        #Parse of the csv file
        doc['artist'] = row[0]
        #parsing genres
        doc['genres'] = row[1].split(';')
        doc['songs'] = row[2]
        doc['popularity'] = row[3]
        doc['link'] = row[4]
        #Add the document to the list of documents
        documents.append(doc)
        #Add the artist name to the list of artists names in the database
        artistsNames.append(row[0])
        doc = {}  #reset the document
    else:
        print(row[0] + " is already on the collection")
    max = max + 1
```

El el ciclo para recorrer las filas del csv se verifica si se llegó al límite de artistas subidos a Mongo, si aun no se ha alcanzado, se verifica si el nombre del artista esta en la lista de artistsNames, si no esta, debe ser agregado a la lista de documentos, para esto se hace el parse y se le asigna los valores correspondientes a cada parse del documento, para genres de hace un split con el ";" para almacenar los genres como una array.

```
#Insert the list of documents into the database
collection.insert_many(documents)
client.close() #close the connection
```

Se usa la función insert_many para insertar todos los documentos a Mongo y se cierra el cliente.

```
except Exception as e:
    print("Unexpected error:", e)
    return 0
return 1
```

Finalmente, si hay un error se despliega el error en la consola.

Parse Lyrics.csv

Se define la función parseArtists para hacer la lectura y el parseo de los artistas de los archivos de artistas.

```
def parseLyrics(lyricsDownloaded_var):
    try:
        client = MongoClient(uri)  #Mongo Client
        client = MongoClient(uri, server_api=ServerApi('1'))
        #Prueba conexión con Mongo DB
        client.admin.command('ping')
        print("Successfully connected to MongoDB")
```

Lo primero que se hace en la función es abrir la conexión de Mongo DB, se envía un ping para comprobar que la conexión es correcta.

```
db = client[str('OpenLyricsSearch')]  #Database to be use
collection = db[str('lyricsCollection')] #Collection to be use
artistCollection = db[str('artistsCollection')] #Collection of artists
```

Después, definimos la base de datos y la collection que se va a usar para cargar/bajar datos a Mongo Atlas. Adicionalmente, se define la artistsCollection para obtener los artistas existentes en Mongo.

```
#csv_reader to parse the csv file
csv_reader = csv.reader(artistDownloaded_var, delimiter=',')
header = next(csv_reader)  #skip the header
```

También, se define un csv reader para hacer la lectura y parseo del csv de artistas, además, se deffine el delimitador por el cual se separan los campos y se hace un skip de la fila del header.

```
documents  = []   #list of documents to be inserted
doc = {} #document to be inserted in the database

artistDocuments = list(artistCollection.find()) #list of artists documents
songLinks = collection.distinct("songLink") #list of song names in the database
```

Posteriormente se define una lista para almacenar los documentos que serán insertados en la collection de Mongo y un documento que almacena la información del lyric que se está leyendo actualmente. También existe la variable **artistsDocuments** para obtener los que están en artistas en Mongo, esto se usa para

agregar la información del artista a cada lyric. También, se define una lista con los songLink que hay en la base de datos, para evitar agregar duplicados.Adicionalmente, se define un max en caso de que se desee limitar la cantidad de artistas que se van a subir a la collection.

```
for row in csv_reader:
    if max == 100:
        break
    #Obtain the artist document from the list of artists documents
    matching_dict = list((d for d in artistDocuments if row[0] == d['link']))
    #Case 1: The artist is not in the database
    if (matching_dict.__len__() == 0):
        print("The artist " + row[0] + " is not in the database")
        continue
```

Se define un ciclo paa ir por cada fila del csv. En este se verifica el max y también se obtiene el documento del artista que hace match con el link del autor del lyric, se verifica que este exista en la base de datos comprobando que matchingDict tenga un len superior a 0.

```
#Case 2: The song is not in the database
elif(row[2] not in songLinks):
    #Parse of the csv file
    doc['artist'] = matching_dict[0]["artist"]
    doc['genres'] = matching_dict[0]["genres"]
    doc['popularity'] = matching_dict[0]["popularity"]
    doc['songs'] = matching_dict[0]["songs"]
    doc['artistLink'] = matching_dict[0]["link"]
    doc['songName'] = row[1]
    doc['songLink'] = row[2]
    doc['lyric'] = row[3]
    doc['language'] = row[4]
    #Add the song name and the artist name to the list of song names in the database
    songLinks.append(row[2])
    #Add the document to the list of documents
    documents.append(doc)
    doc = {} #reset the document
    max = max + 1
else:
    print("The song " + row[1] + " by " + matching_dict[0]["artist"] + " is already on the collection")
```

Para verificar que la canción no exista en la base de datos se verifica el songLink, en caso de que no exista se forma el documento y luego se agrega a la lista de documentos, tambien se agrega el songLink a la lsita de links y se vacia el documento actual.

```
        #Insert the list of documents into the database
        collection.insert_many(documents)
        #Close the connection
        client.close()
        return 1
    except Exception as e:
        print("Unexpected error:", e)
        return 0
    return 1
```

Se usa la función insert_many para insertar todos los documentos a Mongo y se cierra el cliente. Finalmente, si hay un error se despliega el error en la consola.

API

App de React

Pruebas realizadas

Resultados de las pruebas unitarias

Conclusiones

1- La comunicación entre el los miembros de grupo de trabajo es fundamental para un buen desarrollo del proyecto.

2- Se debe mantener una buena organización para poder realizar el trabajo.

3- Es de gran importancia entender los conceptos básicos vistos en clase para realizar el proyecto.

4- El tener un buen control de versiones y la correcta utilización de github facilita el trabajo en equipo.

5- Se deben aplicar buenas prácticas de programación para mantener el orden.

6- Mantener la estructura definida del proyecto es esencial para evitar el desorden.

7- Se debe desarollar un código legible y entendible.

8- Se debe organizar el equipo de trabajo desde el día 1.

9- Se debe tener una estruuctura clara y ordenada del proyecto y lo que requiere.

10- Es importante la división de trabajo para poder desarrollar todos los componentes.

Recomendaciones

1- Hacer reuniones periódicas para discutir los avances del proyecto y mejorar la comunicación.

2- Mantener la organización de la tarea, siguiendo la infraestructura y recomendaciones dadas por el profesor.

3- Repasar los conceptos vistos en clase y complementar con investigación mejorar el entendimiento y aumentar la eficacia con la que se trabajará.

4- Hacer uso de github para el control de versiones y trabajo en conjunto.

5- Seguir un estándar de código.

6- Seguir aprendiendo y enriqueciendo el conocimiento después de finalizar el trabajo.

7- Investigar sobre las diferentes herramientas esenciales para desarrollar la solución e ir tomando apuntes sobre los aspectos importantes de cada uno de estas. Esto facilitará el desarrollo de la solución.

8- Tener una buena estructura del proyecto y dividir el proyecto de forma funcional para avanzar progresivamente.

9- Repartir y asignar tareas a cada integrante del equipo.

10- Definir roles en el equipo de trabajo para mantener el orden y procurar buena dinámica de trabajo.

Referencias bibliográficas donde aplique