Universidad San Carlos de Guatemala

Curso: Sistemas de bases de datos 2

Catedrático: Marlon Orellana

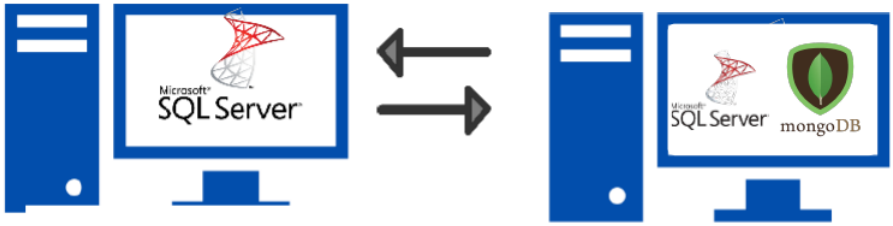
Auxiliar: Jurgen Ramírez

Sección: N

Integrantes:

Nombre	Carnet
Josué David Zea Herrera	201807159
Jorge David Espina Molina	201403632
Kenni Roberto Martínez Marroquín	201800457

Arquitectura del sistema

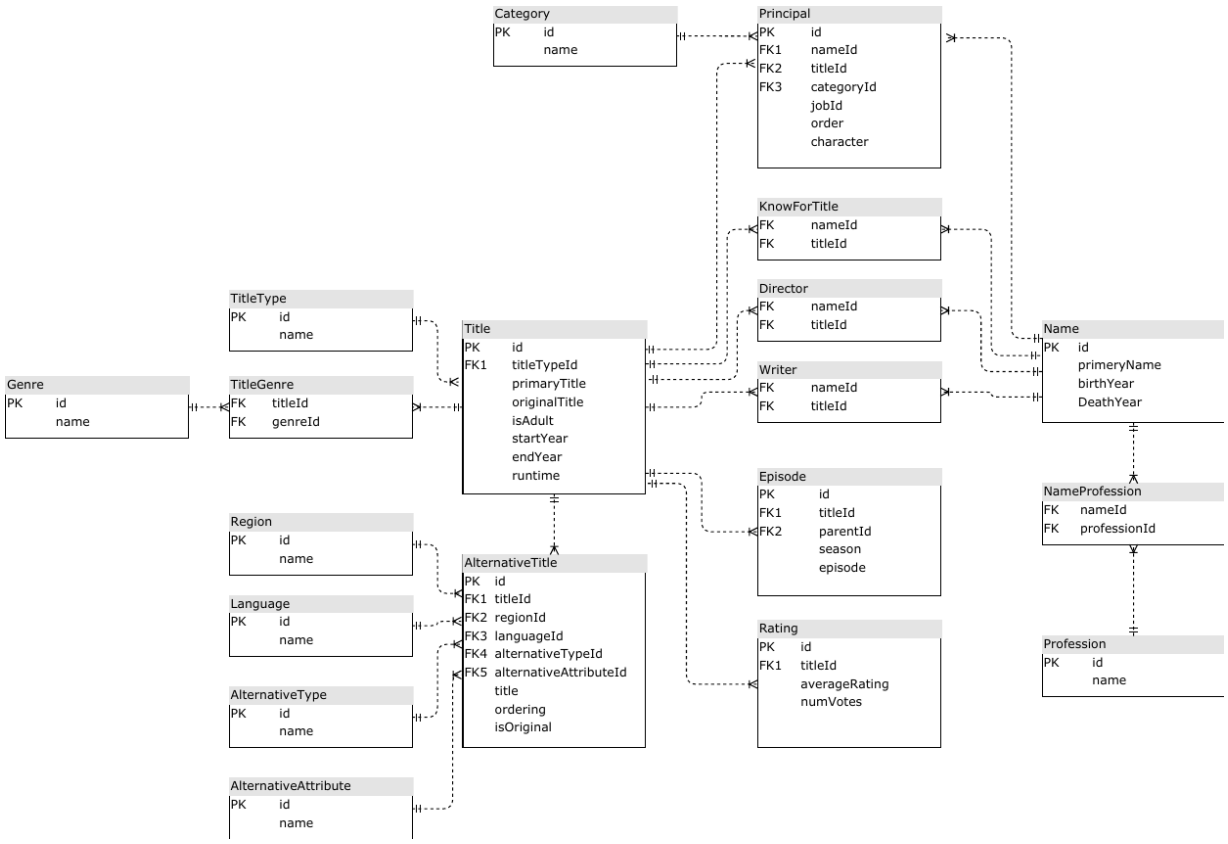


IMDB: 192.168.1.2

NETFLIX y MONGO: 192.168.1.31

IMDB

Esta parte del proyecto se desarrolló en una máquina virtual con el sistema operativo Linux 20.04 LTS, y la base de datos fue implementada en el motor Sql Server, en este se implementó el siguiente modelo relacional:



Carga de datos

Este procedimiento necesita de tablas temporales, mismas que almacenarán la información proporcionada para cargar los correspondientes registros en el modelo relacional ya implementado en la base de datos.

Para este ejemplo se realizará la carga de archivos del archivo titlebasics.tsv.

Primero creamos la tabla temporal que almacenará los registros del archivo, este debe contener la misma estructura de datos a almacenar.

```
CREATE TABLE titlebasics(
    tconst VARCHAR(45),
    titleType VARCHAR(155),
    primaryTitle VARCHAR(455),
    originalTitle VARCHAR(455),
    isAdult VARCHAR(125),
    startYear VARCHAR(45),
    endYear VARCHAR(45),
    runtimeMinutes VARCHAR(45),
    genres VARCHAR(45)
);
```

Procedemos a insertar los datos utilizando la siguiente instrucción:

```
BULK INSERT titlebasics
FROM '/home/canchemolinas/titlebasics.tsv'
WITH
(
    FIRSTROW = 2,
    FIELDTERMINATOR = '\t',
    ROWTERMINATOR = '\n'
);
```

Realizar este procedimiento para todos los archivos .tsv a cargar en el sistema.

Creación del modelo relacional e inserción de los datos

Creamos el modelo relacional a utilizar para posteriormente cargar los datos utilizando la información contenida en las tablas temporales.

Para este ejemplo usaremos la tabla titletype.

Creamos la tabla con la correspondiente estructura.

```
CREATE TABLE titletype(
    id int IDENTITY(1,1),
    name VARCHAR(75),
    PRIMARY KEY(id)
);
```

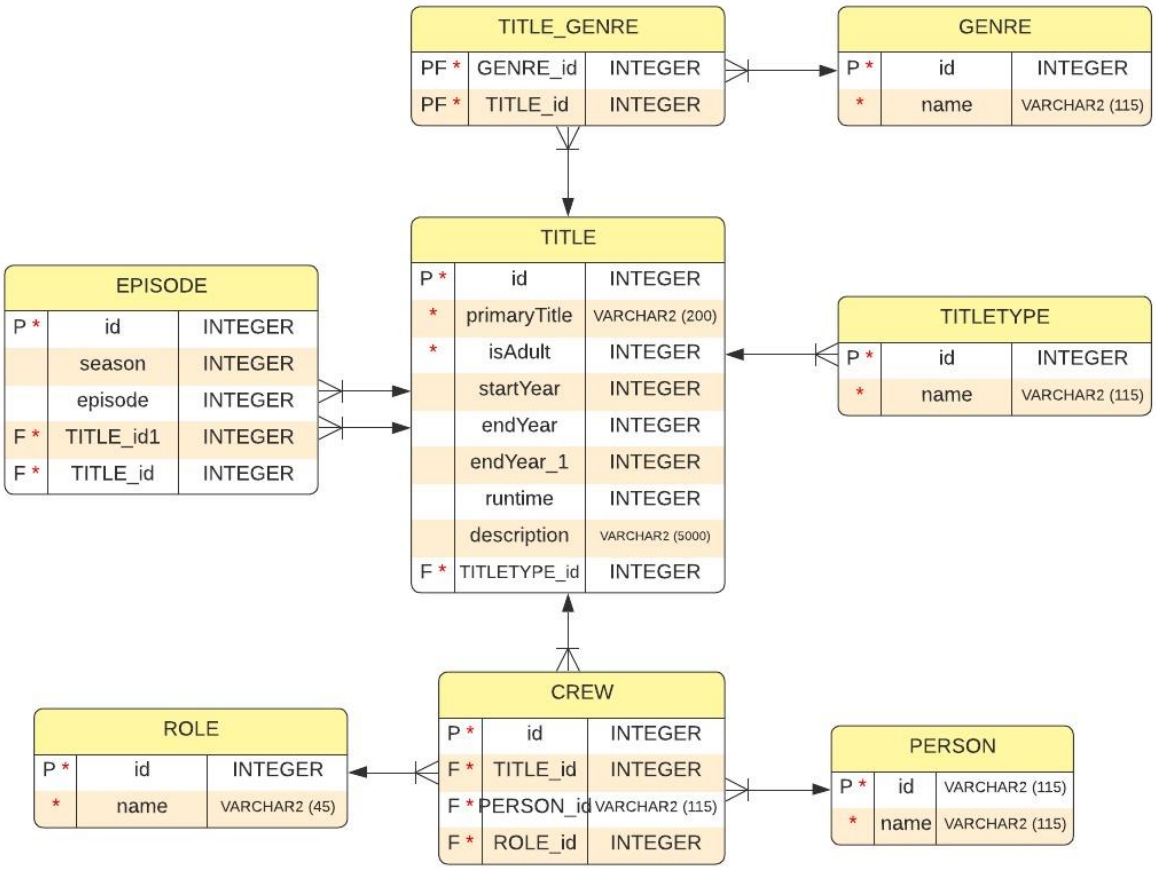
Posteriormente realizamos las correspondientes inserciones, estas se realizan incluyendo la clausula select en el insert, de esta manera se insertarán los registros obtenidos en la consulta, para esta tabla se obtiene la información de la tabla temporal titlebasics.

```
INSERT into titletype
SELECT distinct titleType FROM titlebasics;
```

De esta manera quedarán insertados los datos en la nueva tabla, realizar el mismo procedimiento para todo el modelo relacional, los archivos necesarios para estos procedimientos puede encontrarlos en el siguiente enlace.

Netflix

Esta parte del proyecto se desarrollo en una máquina virtual con el sistema operativo Linux 20.04 LTS, y la base de datos fue implementada en el motor Sql Server, en este se implementó el siguiente modelo relacional:



Este modelo surgió en base al modelo entidad relación implementado en IMDB sin embargo se vio reducido para la base de datos de Netflix con el motivo de mantener aquí únicamente la información necesaria y así poder optimizar los datos.

Descripción de las tablas:

Tabla	**Descripción**
GENRE	Aquí se almacenarán todos los posibles géneros que tendrá algún título.
TITLETYPE	Aquí almacenan los posibles tipos de un título.
TITLE	Aquí almacenamos la información relevante de un título, también se hace referencia a los posibles tipos de títulos y géneros que pueda clasificarse un título. El valor de esta llave primaria es igual a la de IMDB para poder vincular su rating.

TITLE_GENRE	En esta tabla se almacenan todos los géneros posibles en los que pueda clasificarse un título.
EPISODE	En dado caso el título sea una serie, acá se encuentra un detalle de cada episodio correspondientes a un mismo título.
PERSON	Esta tabla almacena todas las personas que pueden verse envueltas en algún título.
ROLE	Acá se especifican los posibles roles que las personas pueden ejercer. Éstos pueden ser actor, actriz, director o escritor.
CREW	Acá se definen todas las personas y rol que ejercen en un mismo título.

Este modelo se definió refiriéndose a algunas de las actuales directivas de Netflix, procedemos a cargar el modelo relacional a la base de datos.

Inserción de datos a Netflix

Para estar insertando nuevas películas en la base de datos de Netflix, se necesita un procedimiento almacenado y que la película que se vaya insertando cree un vínculo que relacione directamente la base de datos IMDB con la de Netflix, dado que ambas están en diferente servidor, se crea un Linked Server para conectar entre servidores.

Linked Service

```
use NETFLIX;

EXEC sp_addlinkedserver
    @server = N'35.193.226.141',
    @srvproduct = N'SQL Server';
```

Con la anterior instrucción quedan enlazadas las bases de datos, la dirección IP 32.193.226.141 que pertenece al servidor que aloja la base de datos de IMDB.

Procedemos a insertar la información de interés y estática para la base de datos de Netflix.

Store procedure

El llenado de las tablas TITLE, EPISODE, TITLE_GENRE y CREW se hacen mediante el uso de un procedimiento almacenado que se encarga de realizar todas las acciones correspondientes.

Iniciamos asignándole el identificador insert_title y asignándole los siguientes parámetros:

Parámetro	Tipo	Descripción
@titulo	VARCHAR	Contiene el nombre del título que se desea almacenar.
@anio	INT	Contiene el año de inicio, el año en el que se inició o estrenó el título a almacenar.
@tipo_titulo	INT	Contiene el tipo de título que es (ya sea, tvSerie, show, etc)
@desc	VARCHAR	Contiene una breve descripción del título que se va a almacenar
@flag	VARCHAR	Parámetro de salida para imprimir el resultado del procedimiento almacenado

Procedemos a crear el sp con sus parámetros.

```
CREATE PROCEDURE insert_title
    @titulo varchar(600),
    @anio int,
    @tipo_titulo int,
    @desc varchar(5000),
    @flag varchar(115) OUTPUT
```

Luego se hace uso de una variable @id_title la cual almacena el id del título en dado caso ya esté registrado en la base de datos de IMDB.

```
DECLARE @id_title VARCHAR(115);

SET @id_title = (
  SELECT TOP 1 tl.id FROM [35.193.226.141].IMDB.dbo.title tl
  WHERE UPPER(tl.primaryTitle) LIKE UPPER(@titulo) AND tl.startYear =
  @anio AND tl.titleTypeId = @tipo_titulo
);
```

Se hace la respectiva verificación si la variable es nula o no (es decir, si el título existe o no). Si no existe, se sale del procedimiento.

```
IF @id_title IS NULL
  BEGIN
    -- SIGNIFICA QUE EL TITULO NO ESTA EN EL CATALOGO DE IMDB
      SET @flag = 'El título ' + @titulo + ' no se ha encontrado.';
      RETURN
  END
```

En dado caso ya exista, se inserta en las tablas respectivas que se han mencionado antes, además se vuelve a verificar que no exista el título a agregar en la base de datos de Netflix.

```
IF (SELECT primaryTitle FROM title t WHERE t.id = @id_title) IS NULL
    BEGIN
        -- INSERTAR A LA TABLA TITLE
        INSERT INTO title(id, primaryTitle, isAdult, startYear, endYear, runtime,
        [description], titleTypeId)
        SELECT TOP 1 tl.id, tl.primaryTitle, tl.isAdult, tl.startYear, tl.endYear,
        tl.runtime, @desc, tl.titleTypeId
        FROM [35.193.226.141].IMDB.dbo.title tl WHERE tl.id = @id_title;
        -- INSERTAR A LA TABLA CREW
        INSERT INTO crew(titleId, personId, roleId)
        SELECT TOP 3 pr.titleId, pr.nameId, role.id
        FROM [35.193.226.141].IMDB.dbo.principal pr INNER JOIN
        [35.193.226.141].IMDB.dbo.category cg ON pr.categoryId = cg.id
        INNER JOIN role ON role.name = cg.name
        WHERE pr.titleId = @id_title AND (cg.name = 'actor' OR cg.name = 'actress');
```

```
        INSERT INTO crew(titleId, personId, roleId)
        SELECT TOP 1 dr.titleId, dr.nameId, role.id
        FROM [35.193.226.141].IMDB.dbo.director dr, role
        WHERE dr.titleId = @id_title AND role.name = 'director';
        INSERT INTO crew(titleId, personId, roleId)
        SELECT TOP 1 dr.titleId, dr.nameId, role.id
        FROM [35.193.226.141].IMDB.dbo.director dr, role
        WHERE dr.titleId = @id_title AND role.name = 'writer';
        -- INSERT TITLE_GENRE
        INSERT INTO title_genre(titleId, genreId)
        SELECT * FROM [35.193.226.141].IMDB.dbo.titleGenre g WHERE g.titleId =
        @id_title;
        -- INSERTAR A LA TABLA EPISODE
        IF (SELECT count(*) FROM [35.193.226.141].IMDB.dbo.episode ep WHERE
            ep.parentId = @id_title) > 0
            BEGIN
                INSERT INTO episode(titleId, parentId, season, episode)
                SELECT ep.titleId, ep.parentId, ep.season, ep.episode
                FROM [35.193.226.141].IMDB.dbo.episode ep WHERE ep.parentId =
                @id_title;
            END

        SET @flag = 'El título ' + @titulo + ' ha sido agregado.';
        RETURN
    END
```

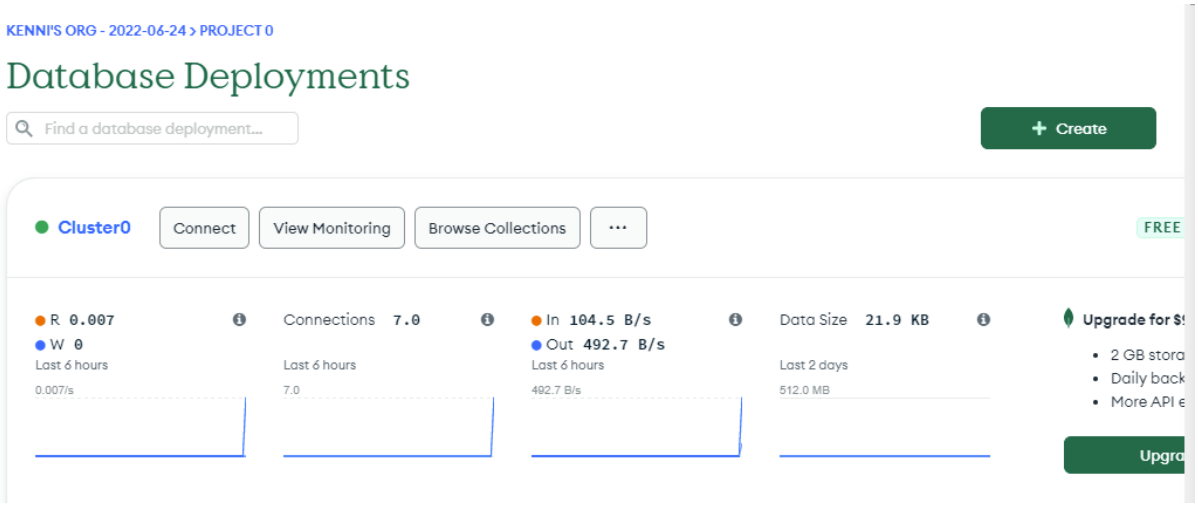
Y si ya ha sido agregado antes, no se agrega nada en la base de datos.

```
ELSE
  BEGIN
    SET @flag = 'El título ' + @titulo + ' ya ha sido agregado antes.';
    RETURN
  END
```

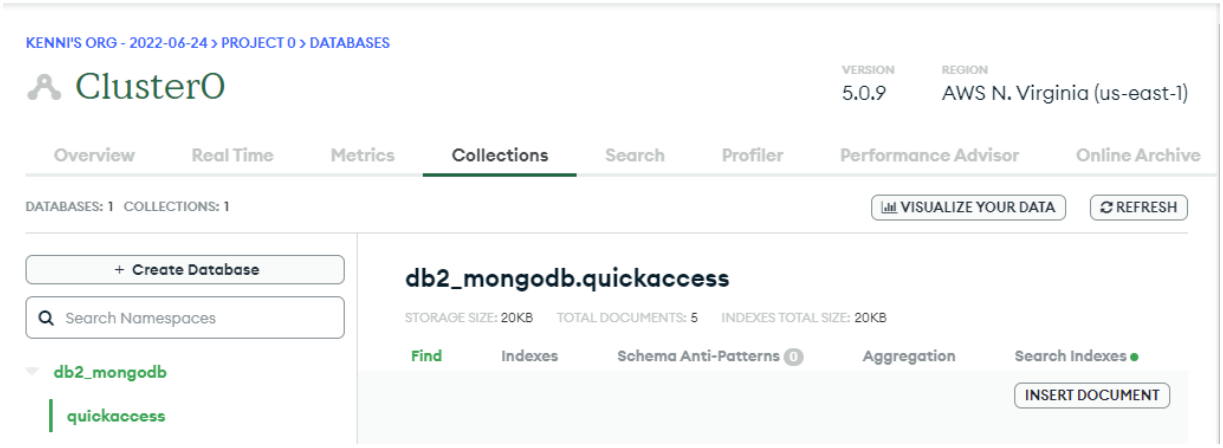
Todos estos scripts puede encontrarlos en el siguiente enlace.

MongoDB

Para implementar una base de datos de MongoDB en la nube, se hizo uso del servicio proporcionado por la misma compañía: Atlas; seguido se creó un clúster en el que se crearía la base de datos:



El clúster creado lleva el nombre de **Cluster0**, la base de datos que se utilizará para las consultas será llamada **db2_mongodb**, y la colección que se utilizará será **quickacess**



La colección puede ser visualizada desde cloud.mongodb.com, sin embargo, si se desea utilizar con MongoDB Compass, es necesario crear un **Connection String:**

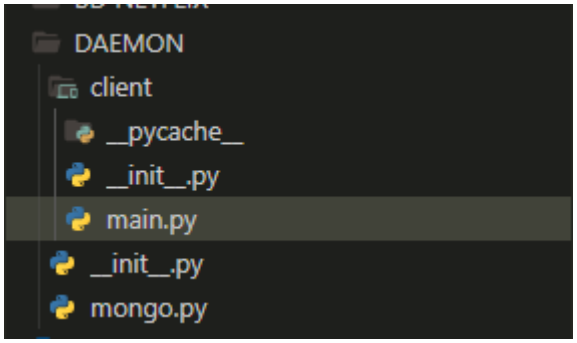
mongodb+srv://db2_p2:8XdKTBUCq4vi3ODF@cluster0.dfzgh5g.mongodb.net/test

Este String puede variar, dependiendo de la aplicación que se desea utilizar para conectarse a la base de datos.

Daemon

Se implementará un Daemon con Python, el cual estará recuperando la información de la base de datos de **NETFLIX.**

La estructura del Daemon será la siguiente:



El cliente será el que contenga las conexiones a SQL Server y a través de este también se podrán ingresar títulos al catálogo de Netflix y obtener consultas generales desde MongoDB.

mongo.py será el archivo que creará el Daemon, para esto haremos uso de la librería **Daemonize**, que puede ser obtenida por medio de pip:

pip install daemonize

En este archivo también creamos una conexión a MongoDB, haciendo uso de la librería **pymongo**:

```
from client import main        You, hace 19 horas • Demonio, cliente y conexión a MongoDB …
from pymongo import MongoClient
from time import sleep
from daemonize import Daemonize

pid = "/tmp/collect-mongo.pid"

# Conectar a mongo
client = MongoClient('mongodb+srv://db2_p2:8XdKTBUCq4vi3ODF@cluster0.dfzgh5g.mongodb.net/?retryWrites=true&w=

# Seleccionar base de datos y coleccion
db = client['db2_mongodb']
collection = db.quickaccess
```

Seleccionamos la Base de datos a utilizar, y luego accedemos a la colección deseada.

Creamos una función que recupere los datos de NETFLIX desde el cliente, y le damos el formato deseado para almacenarlo en nuestra colección:

```
def insert_data():
    for single_title in main.all_titles():
        # Obtener lista de codigos actuales en la base de datos de Netflix
        details = main.detailed_title(single_title)
        # Crear el objeto JSON
        title = {
            "_id": details[0],
            "primaryTitle": details[1],
            "isAdult": details[2],
            "startYear": details[3],
            "endYear": details[4],
            "runtime": details[5],
            "description": details[6],
            "titleType": details[7],
            "cast": main.cast_table(single_title),
            "director": main.get_creator(single_title, 'director'),
            "writer": main.get_creator(single_title, 'writer'),
            "genre": main.get_genres(single_title),
            "rating": {
                "averageRating": main.get_rating(single_title)[0],
                "numVotes": main.get_rating(single_title)[1]
            }
        }
        # Insertar objeto a Mongo
        collection.insert_one(title)
```

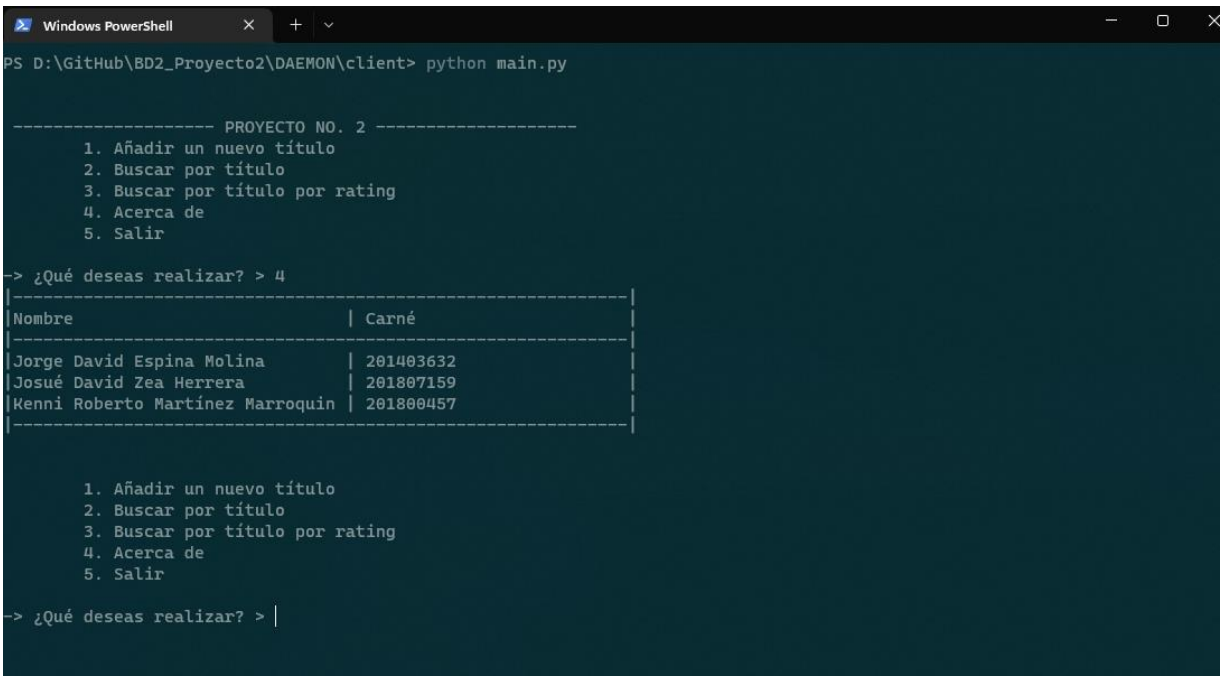
Seguido, creamos un método que será el que tome un temporizador para ejecutar la recuperación de datos cada X cantidad de segundos, finalmente, se utiliza Daemonize para iniciar el Daemon ejecutando el método anterior:

```python
def first():
    while True:
        collection.drop()
        insert_data()
        sleep(15)

daemon = Daemonize(app="run_netflix", pid=pid, action=first())
daemon.start()      You, hace 19 horas • Demonio, cliente y conexión a
```

Cliente

Se cuenta con un sencillo cliente de consola, en él se pueden insertar datos directamente a la base de datos de NETFLIX, haciendo uso del procedimiento almacenado **insert_title**, el cual busca información del título ingresado en la base de datos IMDB.



Se pueden buscar datos en la base de datos de MongoDB, especificando el título del proyecto, o retornar múltiples títulos basados en el rating que se desea buscar:

```
PS D:\GitHub\BD2_Proyecto2\DAEMON\client> python main.py


 -------------------- PROYECTO NO. 2 --------------------
        1. Añadir un nuevo título
        2. Buscar por título
        3. Buscar por título por rating
        4. Acerca de
        5. Salir

-> ¿Qué deseas realizar? > 2
Ingrese el título que desea buscar > Mulan
{'_id': 'tt0238429',
 'cast': ['Ric Herbert', 'Joanna Moore', 'Joanna Moore'],
 'description': '',
 'director': 'Geoff Beak',
 'endYear': 0,
 'genre': ['Animation', 'Family'],
 'isAdult': 0,
 'primaryTitle': 'Mulan',
 'rating': {'averageRating': 6.0, 'numVotes': 133},
 'runtime': 50,
 'startYear': 1998,
 'titleType': 'video',
 'writer': 'Geoff Beak'}


        1. Añadir un nuevo título
        2. Buscar por título
        3. Buscar por título por rating
        4. Acerca de
        5. Salir

-> ¿Qué deseas realizar? >
```

Windows PowerShell

```
        1. Añadir un nuevo título
        2. Buscar por título
        3. Buscar por título por rating
        4. Acerca de
        5. Salir

-> ¿Qué deseas realizar? > 3
Ingrese el rating que desea que posea los títulos > 6
{'_id': 'tt0000009',
 'cast': ['Blanche Bayliss', 'Blanche Bayliss', 'Chauncey Depew'],
 'description': 'Pelicula muda en blanco y negro',
 'director': 'Alexander Black',
 'endYear': 0,
 'genre': ['Romance'],
 'isAdult': 0,
 'primaryTitle': 'Miss Jerry',
 'rating': {'averageRating': 6.0, 'numVotes': 152},
 'runtime': 45,
 'startYear': 1894,
 'titleType': 'movie',
 'writer': 'Alexander Black'}


{'_id': 'tt0000023',
 'cast': [],
 'description': '',
 'director': 'Louis Lumi+¿re',
 'endYear': 0,
 'genre': ['Documentary', 'Short'],
 'isAdult': 0,
 'primaryTitle': 'The Sea',
 'rating': {'averageRating': 6.0, 'numVotes': 1094},
 'runtime': 1,
 'startYear': 1895,
 'titleType': 'short',
 'writer': 'Louis Lumi+¿re'}


{'_id': 'tt0000997',
 'cast': ['Kate Bruce', 'Gertrude Robinson', 'Edith Haldeman'],
 'description': 'Hetty is engaged to George, but after her sister dies she '
                "breaks the engagement in order to raise her sister's orphaned "
                'daughter.',
 'director': 'D.W. Griffith',
 'endYear': 0,
 'genre': ['Drama', 'Short'],
 'isAdult': 0,
```

La información obtenida de estas dos opciones de búsqueda se realiza a través de MongoDB:

```
def get_titleMongo():
    title = input("Ingrese el título que desea buscar > ")
    if collection.count_documents({"primaryTitle":title}) != 0:
        result = collection.find({"primaryTitle":title})
        for x in result:
            pprint(x)
            print("\n")
    else: print("No se ha encontrado ningún título con el criterio ingresado.")
    menu()

def get_titleByRatingMongo():
    rating = input("Ingrese el rating que desea que posea los títulos > ")
    result = collection.find()
    for x in result:
        x1 = json.dumps(x)
        xjson = json.loads(x1)
        ratings = json.loads(json.dumps(xjson["rating"]))
        if (ratings["averageRating"]==float(rating)):
            pprint(x)
            print("\n")
    menu()
```

El Daemon utiliza consultas a NETFLIX para luego insertarlas en MongoDB: estas son unas de las funciones que se utilizan:

```
def all_titles():
    cursor.execute('SELECT id FROM title;')
    titles = []
    row = cursor.fetchone()
    while row:
        titles.append(str(row[0]))
        row = cursor.fetchone()

    return titles

def cast_table(id_title):
    cursor.execute("SELECT p.name FROM crew c INNER JOIN person p ON c.personI
    cast = []
    row = cursor.fetchone()
    while row:
        cast.append(str(row[0]))
        row = cursor.fetchone()
    return cast

def detailed_title(id_title):
    cursor.execute("SELECT t.id, t.primaryTitle, t.isAdult, t.startYear, t.end
    return cursor.fetchone()

def get_genres(id_title):
    cursor.execute("SELECT g.name FROM title_genre tg INNER JOIN genre g ON tg
    genres = []
    row = cursor.fetchone()
    while row:
        genres.append(str(row[0]))
        row = cursor.fetchone()
    return genres
```