

מסמך תיעוד | Cinema Bay

עיצוב מסד הנתונים

מסד הנתונים של Cinema Bay מכיל כ- 100,000 רשומות המרכיבות 7 טבלאות המשמשות את האפליקציה:

1. FILM - טבלה שמייצגת סרט עם כל ה- attributes הקשורות מהזאה שלו:

film_id	title	year	image	summary	trailer	rating	director
---------	-------	------	-------	---------	---------	--------	----------

- film_id: מזהה ייחודי שמייצג את הסרט
- title: שם הסרט
- year: שנת ההוצאה של הסרט
- image: קישור לפוסטר של הסרט
- summary: תקציר על הסרט
- trailer: קישור לטרילר ב- YouTube
- rating: דירוג של הסרט לפי אתר IMDb
- director: שם הבמאי

אינדקסים וModelPropertyות:

	Column	Type	Primary/Foreign/Index
PRIMARY	film_id	BTREE	Primary
	title_index	BTREE	Index
	year_index	BTREE	Index
	summary_index	FULLTEXT	Index
	rating_index	BTREE	Index
	director_index	BTREE	Index

шиಕולים בקביעת הסכמה של הטבלה: המידע שהטבלה FILM מכילה נשלף מה- APIs במספר איטרציות, ולכן בהתחלה חשבנו ליצור טבלאות קטנות מ- 2-3 עמודות, כך ש- film_id יהיה מפתח ראשי בכל אחד ואוותם attributes שנשלפו ביחד. בסוף החלנו לעבוד עם טבלה גדולה שתכלול את כל ה- attributes הנ"ל ביחד, ואת מכיוון ש- film_id קובע את כלו, כך שסכמה רחבה תחסוך לנו הרבה פעולה של join בין ה- primary key ו- foreign key. כמו כן, האפליקציה שלנו לא צריכה לחפש מידע על סרט מסוים.

אפשרויות: העמודות title, year, rating של הטבלה FILM שימשו אותנו בשאלות בפקודות ORDER BY WHERE, GROUP BY ועוד. מסד הנתונים שלנו מכיל כ- 100,000 רשומות, ולכן היה חשוב לנו לזכור אינדקס לכל عمودה שהשlijפה באחת מהשאלות (השאלות מופיעות בהמשך) מתבצעת לפיה. הוספה האינדקסים שיפורה את מהירות שlijft הנתונים מה- DB שימושית. כמו כן, האפליקציה שלנו מבצעת חיפוש בעמודה summary המכילה מידע מטיפוס TEXT, ולכן גם עבור העמודה הזאת הגדרנו אינדקס מטיפוס FULLTEXT, ואת כדי שהchiposh בתקציר של הסרט יבוצע מהר ככל הניתן.

2. FILM_LOCATION - טבלה שמייצגת את אתרי הצילום של הסרטים:

film_id	location
---------	----------

- מזהה של הסרט film_id
- אתר צילום location

אינדקסים ומפתחות:

Column	Type	Primary/Foreign/Index
film_id	BTREE	Foreign, references film_id of FILM
film_id_index	BTREE	Index

שיקולים בקביעת הסכמה של הטבלה: מכיוון שלכל סרט יש הרבה אתרים צילום, וכל אתרים צילום יכול להיות משותף עם כמה סרטים, אז שמרנו את המידע על אתרים הצילום בטבלה נפרדת. בחרנו לא לשים את המידע על אתרים הצילום בטבלה FILM כדי שלא יהיה לנו כפילויות במידע בטבלה FILM.

אופטימיזציות: אין לנו שאליות שמחפשות לפי אתר הצילום لكن לא היה צורך שם באינדקס, אולם ישנו שאליות שמסנוות לפי ה- id film_id ולכן הוספנו שם אינדקס (ליעול פקודות join למשל).

3. FILM_GENRE - טבלה שמייצגת את הז'אנרים שהסרט משתייך אליו:

film_id	genre
---------	-------

- מזהה של הסרט film_id
- ז'אנר שהסרט משתייך אליו (לכל סרט יכולים להיות כמה כאלה) genre

אינדקסים ומפתחות:

Column	Type	Primary/Foreign/Index
film_id	BTREE	Foreign, references film_id of FILM
film_id_index	BTREE	Index
genre_index	BTREE	Index

שיקולים בקביעת הסכמה של הטבלה: השיקולים זהים לשיקולים של הטבלה FILM_LOCATION (לכל סרט יכולים להיות כמה ז'אנרים ולהפוך).

אופטימיזציות: הוספנו אינדקס לעמודות genre ו- id_film כדי לשפר את זמן השליפה של השאלות שמסתמכו על הערך של העמודה genre ו/או id_film.

4. FILM_PROVIDER - טבלה שמייצגת את שירותי הסטרימינג שמשדרים את הסרט:

film_id	provider
---------	----------

- film_id: מזהה של הסרט
- provider: שירות סטרימינג שמשדר את הסרט

אינדקסים ומפתחות:

Column	Type	Primary/Foreign/Index
film_id	film_id	BTREE Foreign, references film_id of FILM
film_id_index	film_id	BTREE Index

שיקולים בקביעת הסכמה של הטבלה: השיקולים זרים לשיקולים של הטבלה FILM_LOCATION (לכל סרט יכולם להיות כמה שירותי סטרימינג מתאימים ולהפוך).

אפשרויות אופטימיזציה: אין לנו שאלות שמסנוות רשות לфи שירות הסטרימינג, אבל ישן שאלות שמסנוות לפי film_id لكن הוספנו שם אינדקס.

5. FILM_AWARD - טבלה שמייצגת את הפרסים הסרט זכה בהם, כולל מספר הזכיות בכל פרס:

film_id	award	count
---------	-------	-------

- film_id: מזהה של הסרט
- award: שם הפרס
- count: מספר הפעמים שהסרט זכה בפרס

אינדקסים ומפתחות:

Column	Type	Primary/Foreign/Index
film_id	film_id	BTREE Foreign, references film_id of FILM
film_id_index	film_id	BTREE Index

שיקולים בקביעות הסכמה של הטבלה: השיקולים זרים לשיקולים של הטבלה FILM_LOCATION (סרט יכול לזכות במספר פרסים ולהפוך).

אפשרויות אופטימיזציה: הוספנו אינדקס עבור film_id כדי ליעיל שאלות שמסנוות לפי הערך של id.

.6 - טבלה שמייצגת שחקן/ית עם כל ה- attributes שנובעות מה�性ה שלו:

actor_id	actor_name	birthdate	image
----------	------------	-----------	-------

- מזהה ייחודי של השחקן/ית :actor_id
- שם השחקן/ית :actor_name
- תאריך הלידה של השחקן/ית :birthdate
- קישור לתמונה של השחקן/ית :image

אינדקסים ומפתחות:

	Column	Type	Primary/Foreign/Index
PRIMARY	actor_id	BTREE	Primary
	actor_name_index	BTREE	Index
	actor_birthdate_index	BTREE	Index

שיקולים בקביעות הסכמה של הטבלה: הטעלה הנ"ל מייצגת שחקן/ית, ולכן מוגנו שהיא תהיה נפרדת מהטבלאות של הסרטים כדי שלא יהיו לנו כפליות במידע בטלנות (הקשר בין השחקנים והסרטים יעשה ע"י join), וכי שפעולות העדכון של הטבלאות יהיו פשוטות ככל הניתן.

אפשרויות: הוספנו אינדקסים לעמודות actor_name ו- birthdate, וזאת כי יש לנו שאלות שמחפשות לפי העמודות האלה, וכך במקרה כזו האינדקסים יכולים לשפר את זמן השיפוח מהטבלה.

.7 - טבלה שמקשרת בין סרט לשחקנים הראשיים שלו: FILM_STAR

film_id	actor_id
---------	----------

- מזהה של סרט :film_id
- מזהה של שחקן/ית ראשי/ת בסרט :actor_id

אינדקסים ומפתחות:

	Column	Type	Primary/Foreign/Index
	film_id	BTREE	Foreign, references film_id of FILM
	film_id_index	BTREE	Index
	actor_id	BTREE	Foreign, references actor_id of ACTOR
	actor_id_index	BTREE	Index

שיקולים בקביעת הסכמה של הטבלה: השיקולים זרים לשיקולים של הטעלה FILM_LOCATION (לסרט יכול להיות כמה שחקנים ראשיים ולהפוך).

אפשרויות: הוספנו אינדקסים לעמודות film_id ו- actor_id כדי ליעל את השאלות שמסננות לפיו שתי העמודות האלה.

הערה: אחרי שתיארנו את הTELLOWS של מסד הנתונים ניר שטללאות נקבעו כך שלא תהיה אף הפרה של BCNF, כאשר התלוויות במקרה שלנו הן:

- בטבלה FILM

film_id → title, year, image, summary, trailer, rating, director

- בטבלה ACTOR

actor_id → actor_name, birthdate, image

- שימוש בטבלאות גדולות יותר (শম্পুল অবস্থা attributes) יביא בהכרח להפרה של BCNF, מה שיגרום לשכפולי מידע מיותרים לגמר.
- חלוקת הטבלאות שלנו לטבלאות עוד יותר קטנות תביא אף היא לשכפולי מידע, כי הרוי אם נחלק את FILM (למשל) לשתי טבלאות, אז ה- `film_id` יctrיך להופיע בשתייהן, אבל אין צורך בכך.

השאלות המרכזיות

1. החזרת השמות של כל השחקנים שנולדו בחודש מסוים, ממויינים (בסדר יורד) לפי ממוצע הדירוג של הסרטים שבהם השחקן השתתף:

```

1  SELECT actor_name
2  FROM
3      (SELECT ACTOR.actor_id, ACTOR.actor_name, AVG(FILM.rating) AS film_avg
4      FROM ACTOR, FILM, FILM_STAR
5      WHERE ACTOR.birthdate LIKE '%.1'
6      AND ACTOR.actor_id = FILM_STAR.actor_id
7      AND FILM.film_id = FILM_STAR.film_id
8      GROUP BY ACTOR.actor_id, ACTOR.actor_name
9      ORDER BY film_avg DESC ) SUB_QUERY

```

הדוגמה הנ"ל תחזיר את השמות של כל השחקנים שנולדו בינוואר, ממויינים (בסדר יורד) לפי ממוצע הדירוג של הסרטים בהם השתתפו.
הטבלאות מتوزן מס' הנתונים שהשאילתה עשתה בהם שימוש: ACTOR, FILM וכבונן הטבלה FILM_STAR שמקשרת בין הטבלאות.

2. החזרת הז'אנרים שאלהם סרט נתון משתייך ביחיד עם המיקום של הסרט באותו ז'אנר (יחסית לסרטים האחרים שימושיים לו), ממויינים (בסדר עולה) לפי הדירוגים:

```

1  SELECT FILM_GENRE.genre, COUNT(*) AS cnt
2  FROM FILM, FILM_GENRE
3  WHERE FILM.film_id = FILM_GENRE.film_id
4  AND FILM.rating >
5  (SELECT FILM.rating FROM FILM WHERE FILM.film_id = 'tt3417422')
6  AND FILM_GENRE.genre IN
7      (SELECT FILM_GENRE.genre
8      FROM FILM_GENRE
9      WHERE FILM_GENRE.film_id = 'tt3417422')
10 GROUP BY FILM_GENRE.genre
11 ORDER BY cnt;

```

הדוגמה הנ"ל תחזיר את המיקום של הסרט שהזהה שלו הוא tt3417422 בכל אחד מהז'אנרים שליהם הסרט משתייך, כאשר התוצאה ממויינת לפי המיקומים בסדר עולה.
הטבלאות מتوزן מס' הנתונים שהשאילתה עשתה בהם שימוש: FILM ו התבלה FILM_GENRE שמקשרת את הסרט לז'אנרים שאלהם הוא משתייך.

3. החזרת המזהים של הסרטים שיש להם מעל x שחקנים ראשיים משותפים עם סרט נתון, ביחיד עם מס' הסרט השחקנים המשותפים, כאשר התוצאה ממומינית בסדר עולה לפי שנת ההוצאה של הסרטים. השאלה הזו עזרה לנו כדי לנחש את הסרטים שימושתיים לאותה סדרת סרטים ולהחזיר אותם בסדר כרונולוגי:

```

1  SELECT FILM.film_id, COUNT(*) AS shared
2  FROM FILM, FILM_STAR
3  WHERE FILM.film_id <> 'tt3417422'
4  AND FILM.film_id = FILM_STAR.film_id
5  AND FILM_STAR.actor_id IN
6      (SELECT FILM_STAR.actor_id
7       FROM FILM, FILM_STAR
8       WHERE FILM.film_id = FILM_STAR.film_id
9       AND FILM.film_id = 'tt3417422')
10 GROUP BY FILM.film_id
11 HAVING shared > 10
12 ORDER BY FILM.year;
```

הדוגמה הנ"ל תחזיר את כל הסרטים שיש להם מעל ל- 10 שחקנים משותפים עם הסרט שהמזהה שלו הוא *tt3417422* בסדר כרונולוגי.

הטבלאות מتوزן מוצד הנתונים שהשאילתה עשתה בהם שימוש: FILM והטבלה FILM_STAR שמקשרת בין הסרט והשחקנים הראשיים שהוא תפקידו הסרט.

4. החזרת כל הסרטים שבתקציר שלהם מופיעה מילה מסוימת: Full-Text Search Query .

```

1  SELECT FILM.film_id
2  FROM FILM
3  WHERE Match(summary) Against('happy' IN BOOLEAN MODE);
4
```

הדוגמה הנ"ל תחזיר את כל מזהי הסרטים שבתקציר שלהם מופיעה המילה *happy*. כדי ליעיל את החיפוש בטקסט הגדרכנו FULLTEXT INDEX על העמודה summary של הטבלה FILM.

הטבלאות מتوزן מוצד הנתונים שהשאילתה עשתה בהם שימוש: הטבלה FILM שמכילה את העמודה .summary

5. החזרת שמות השחקנים שהשתתפו הסרטים שבויימו ע"י במאית נתון ואת מס' הסרטים (של אותו במאית) שבהם הם השתתפו, כאשר התוצאה ממומינית בסדר יורד של מס' הסרטים.

```

1  SELECT ACTOR.actor_name, COUNT(*) AS times
2  FROM ACTOR, FILM_STAR, FILM
3  WHERE ACTOR.actor_id = FILM_STAR.actor_id
4  AND FILM_STAR.film_id = FILM.film_id
5  AND FILM.director = 'David Yates'
6  GROUP BY ACTOR.actor_name
7  ORDER BY times DESC
```

הדוגמה הנ"ל תחזיר את שמות השחקנים שהשתתפו הסרטים שבויימו ע"י David Yates ואת מס' הסרטים (של אותו במאית) שבהם הם השתתפו, כאשר התוצאה ממומינית בסדר יורד של מס' הסרטים.

הטבלאות מتوزן מוצד הנתונים שהשאילתה עשתה בהם שימוש: הטבלאות FILM, ACTOR והטבלה FILM_STAR שמקשרת בין הסרט והשחקנים הראשיים.

6. בהנחת מזהה של סרט x , מספר שנים y ו- f , להחזיר את המזהים של כל הסרטים ותמונות הפוסטר שלהם בתנאי שיש להם לפחות z אןර משותף אחד עם הסרט x , שהמරחק בין שנות ההוצאה שלהם ושותת ההוצאה של הסרט x קטן מ- y , והמרחק בין הדירוג שלהם לזה של x קטן מ- f . השאלה זו שמשה אותנו כדי להמליץ למשתמש על סרטים בהתחסס על סרט מסוים שהוא אהב.

```

1  SELECT DISTINCT FILM.film_id, FILM.image
2  FROM FILM, FILM_GENRE,
3      (SELECT FILM.year AS f_year
4       FROM FILM
5       WHERE FILM.film_id = 'tt3417422') SUB_QUERY_1,
6      (SELECT FILM.rating AS f_rating
7       FROM FILM
8       WHERE FILM.film_id = 'tt3417422') SUB_QUERY_2
9  WHERE FILM.film_id = FILM_GENRE.film_id
10 AND FILM.film_id <> 'tt3417422'
11 AND FILM_GENRE.genre IN
12     (SELECT FILM_GENRE.genre
13      FROM FILM_GENRE
14      WHERE FILM_GENRE.film_id = 'tt3417422')
15 AND ABS(FILM.year - f_year) < 5
16 AND ABS(FILM.rating - f_rating) < 0.5;
```

הדוגמה הנ"ל תחזיר את המזהים של הסרטים ואת תמונות הפוסטר שלהם אם נחטכים עם הסרט שהמזהה שלו הוא $tt3417422$ בז'אנר אחד, ושותת ההוצאה שלהם רחוקה משותת ההוצאה שלו ב- 4 שנים לכל היותר, וה למרחק בדירוג ביניהם קטן מ- 0.5.

הטבלאות מتوزع מסד הנתונים שהשאילתה עשתה בהם שימוש: FILM ו- הטבלה FILM_GENRE שמקשרת את הסרט לז'אנרים שאליים הוא משתיעך.

7. החזרת מספר הסרטים מכל ז'אנר ששחקן מסוים השתתף בהם, כאשר התוצאה ממויינת בסדר יורד לפי מספר הסרטים:

```

1  SELECT FILM_GENRE.genre, COUNT(*) popularity
2  FROM FILM_GENRE, FILM, FILM_STAR
3  WHERE FILM_GENRE.film_id = FILM.film_id
4  AND FILM.film_id = FILM_STAR.film_id
5  AND FILM_STAR.actor_id = 'nm0264578'
6  GROUP BY FILM_GENRE.genre
7  ORDER BY popularity DESC;
```

הדוגמה הנ"ל תחזיר עבור השחקן שהמזהה שלו הוא $nm0264578$ את הז'אנרים של הסרטים שהשתתפו בהם, בלבד עם מספר הסרטים מכל ז'אנר, כאשר התוצאה ממויינת בסדר יורד לפי מספר הסרטים.

הטבלאות מتوزע מסד הנתונים שהשאילתה עשתה בהם שימוש: FILM ו- הטבלאות FILM_GENRE ו- FILM_STAR שמקשרות בין הסרט לז'אנרים שלו ולשחקנים הראשיים.

8. לכל שחקן מחזיר את המזהה, השם, תאריך הלידה, התמונה וממוצע הדירוגים של הסרטים בהם הוא השתתף, בתנאי שהיה שחקן ראשי בסרט נתון:

```
1  SELECT ACTOR.actor_id,
2        ACTOR.actor_name,
3        ACTOR.birthdate,
4        ACTOR.image,
5        AVG(FILM.rating)
6  FROM ACTOR, FILM_STAR, FILM
7 WHERE ACTOR.actor_id IN
8      (SELECT actor_id
9       FROM FILM_STAR
10      WHERE FILM_STAR.film_id = 'tt3417422')
11 AND FILM_STAR.film_id = FILM.film_id
12 AND FILM_STAR.actor_id = ACTOR.actor_id
13 GROUP BY ACTOR.actor_id;
```

הדוגמה הנ"ל ממחירה לכל שחקן ראשי בסרט *tt3417422* את המזהה, השם, תאריך הלידה, התמונה, וממוצע הדירוגים של כל הסרטים בהם הוא השתתף.

הטבלאות מתוך מסד הנתונים שהשאילתת עשתה בהם שימוש: FILM_STARS, ACTOR ו הטבלה FILM שמקשרת בין הסרט לשחקנים הראשיים שלו.

יעול השאלות

כפי שמצוין בחלק הראשון על עיצוב מסד הנתונים, ניתן לראות שהוספנו אינדקס לכל عمودה שהופיעה בפקודת ORDER BY, WHERE, GROUP BY ו/or副标题. עבור הפקודות האלה, השימוש באינדקס חוסך הרבה זמן העיבוד, וזאת כי הפקודות האלה שולפות את המידע לפי סדר מסוים, ולכן האינדקסים יכולים לתרום תרומה משמעותית בשיפור זמן העיבוד של השאלות. נזכיר כי עבור primary keys לא הגדרנו אינדקסים נוספים, כי ה- primary key עצמו מגדיר אינדקס זה.

מבנה הקוד

קוד הפרויקט נמצא בתיקייה SRC לפי הסדר הבא:

- SRC/APPLICATION-SOURCE-CODE בـ Flask:
 - תיקייה זו מכילה את קבצי אפליקציה הדואת web המשמשת

- server.py: תוכנית פיתון ש谋ריצה את שרת הדואת Flask. קובץ זה מכיל את כל הפונקציות שפותפעלות את השרת, מתחברות לשרת הדואת DB ומתשאלות אותו לקבלת מידע. הפונקציות הראשיות של השרת הן הפונקציות שמרנדרות את קבצי הדואת html של האתר:

- * הפונקציה "home()" מרנدرת את דף הבית עם התוכן שלו
- * הפונקציה "movie" מרנדרת את דף תוצאת החיפוש עם כל התכנים והאינפורמציה על הסרט המבוקש

הfonקציות שמתשאלות את הדואת DB משתמשות בסה"כ בשאלות שתווארו בהרחבה בסעיף שמתיחס לשאלות המרכזיות.

- static: מכיל קבצי CSS ושתי תמונות בירית מחדל שהשתמשנו בהן בשלא נמצאו תמונות מתאימות בـ API.

- templates: מכיל את שני קבצי html שאוטם אנחנו מרנדרים ואשר מרחיבים את html. כמו כן יש בתיקייה קובץ html עבור תוצאות חיפוש שלא הצלחו (not_found.html).

- SRC/CREATE-DB-SCRIPT.sql - זה הקובץ שמייצר את הטבלאות והאינדקסים של מסד הנתונים.

- SRC/API-DATA-RETRIEVE.py - קובץ זה מחולק לשני חלקים מרכזיים:

1. הפקת המידע באמצעות הפונקציה `retrieve_data` - פונקציה זו אחראית על איסוף המידע מכל מקורות המידע (שיטווארו בהמשך) ושמירותם בקבצי xml לוקליים כדי לגבות את מסד הנתונים.

2. הכנסת המידע משלב 1 למסד הנתונים באמצעות הפונקציה `insert_data_into_db`. הפונקציה הזאת מבצעת parsing לקבצי xml משלב 1 ומכניסה את המידע למסד הנתונים. העיר שtat הקובץ הזה ניתן להריץ אך ורק אחרי שמייצרים את הטבלאות באמצעות הקובץ CREATE-DB-SCRIPT.sql. כמו כן, ניתן לשירות לבצע את שלב 2 ולהימנע מגישות מיותרות לـ API אם משתמשים בקבצי XML הlokליים שמגבים את הדואת DB.

- SRC/HTML_IMDB_FILES - תיקייה זו מכילה 20 דפי HTML שנלקחו מאתר IMDb ושמם קיבלנו את הדואת TOP 1000 Popular Movies.

• SRC/XML_FILES - תיקייה זו מכילה קבצי XML שמהווים גיבוי למידע שיש במסד הנתונים. קבצים אלה מאפשרים לנו לגבות את הדואת DB, ולהימנע מקריאות מיותרות לـ API וUMBZO זמן מיותר על הקריאה הלאה אם נרצה למלא את הדואת DB מחדש.

מקורות המידע

מסד הנתונים שלנו מכיל מידע ממקורות מידע אחדים:

1. אתר האינטרנט IMDb:

קישור:

<https://www.imdb.com>

מהאתר זהה הורדנו דפי html שמכילים בין היתר את המזהים של 1000 הסרטים הכי פופולריים. מזהים אלה שימושו אותנו כדי להוציא מידע מה APIs האחרים שאיתם עבדנו, ושבהם כל סרט מזוהה ע"י המזהה של הסרט ב-IMDb. כדי להוציא את המידע מדף html היינו צריכים לעשיות parsing לדפי ה- html ולהוציא מכם מחרוזות שתואמות לבנייה הייחודי של מזהים הסרטים כפי שהם מוגדרים באתר. כל עמודות film_id של מסד הנתונים מכילות בעצם מידע שהפקנו מהמקור הזה.

הקוד הרלוונטי: הפונקציה `read_popular_movies` שבקובץ `API-DATA-RETRIEVE.py`

2. API details OTT :

קישור:

<https://rapidapi.com/gox-ai-gox-ai-default/api/ott-details>

המשק הזה סיפק לנו את המידע על השירותים `streaming services`. בהתחלה הורדנו את המידע שאנחנו צריכים ושמרנו אותו בקובץ xml לוקלי. קובץ ה-xml שימש אותנו כקובץ גיבוי עבור מסד הנתונים. בשלב השני קראנו את הקובץ ה-xml והכנסנו את המידע למסד הנתונים. הטבלה `FILM_PROVIDER` מאכלה את המידע שהילכנו מ-

הקוד הרלוונטי: הפונקציה `read_providers` שבקובץ `API-DATA-RETRIEVE.py`

3. API IMDb8 :

קישור:

<https://rapidapi.com/apidojo/api/imdb8>

המשק הזה סיפק לנו את מרבית המידע השמור במסד הנתונים. חוץ מזהי הסרטים, שירותי הסטרימינג והקישורים של הטריילרים, כל המידע הופק באמצעות ה- API זהה. בדומה לעובדה עםOTT details API, גם כאן בחרנו לשמר את המידע המקומי על קבצי xml, כדי שייהה לנו מאיפה לשחזר את הנתונים במקרה הצורך (וגם כדי לחסוך בתשלומים עבור ה- API).

הפונקציות הרלוונטיות מתוך הקובץ: `API-DATA-RETRIEVE.py`

- `read_actors_ids_to_xml_file`
- `read_movies_details`
- `read_summary`
- `read_rating`
- `read_genres`
- `read_locations`
- `read_cast`
- `read_director`
- `read_cast_name`
- `read_awards`

4. אתר האינטרנט YouTube :

קישור:

<https://www.youtube.com>

באמצעות בקשות http לאתר <https://www.youtube.com> ניתן למצוא קישורים ב-YouTube עבור הטריילרים שנמצאים במסד הנתונים. מידע זה נשמר בעמודה `FILM_trailer` שבטבלה

הקוד הרלוונטי: הפונקציה `read_trailers` שבקובץ `API-DATA-RETRIEVE.py`

ספריות חיצונית

- Flask - ספריית ה- python backend (python) שהשתמשנו בה בפרויקט. כפי שנאמר בכיתה, בחרנו ב- Flask בגלל הפשטות שלה, ומכיון שהפרויקט שלנו לא עסק בבניית אתר מורכב מדי או בבחירה הספיקה לצרכים שלנו.
- Bleach - ספריה שמשמשת כדי להטמינו עם קלטיים זדוניים מצד המשתמש (למשל כדי להטמינו עם SQL injection). השתמשנו בספריה הזאת כדי למנוע מהמשתמש לחבל בסיס הנתונים ע"י הכנסת קלט ענייתי בתור שם של סרט.
- my-sql-connector-python - ספריה שהשתמשנו בה על מנת להתחבר לשרת ה- DB, ל问我 אותו ולעדרן את הטבלאות.

General Flow

כשהמשתמש מתחבר לאתר אז Flask מרים את הפונקציה "home()". הפונקציה הזאת מתחאלת את ה- db לקבלת תמונות ומצאים של 30 סרטים אקרים (קוראת לפונקציה () get_movie_poster). כמו כן היא מתחאלת את השרת לקבלת השמות של השחקנים שנולדו בחודש הנוכחי. לאחר מכן הפונקציה מרנדרת את הדף home.html עם המידע שהתקבל.

מתוך דף הבית:

- אם המשתמש ילחץ על פостר של סרט מלאה שמופיעים ברקע אז ירים את הפונקציה "movie"(Flask Search "Search" אז ירים את הפונקציה "get_details_by_id"). הפונקציה זו מתחאלת את ה- DB לקבלת המידע המלא על הסרט שנבחר (קוראת לפונקציה () get_details_by_id). כשמתקבל המידע הפונקציה מרנדרת את הדף movie.html עם כל התכנים שהתקבלו.
- אם המשתמש יקליד שם של סרט בשורת החיפוש וילחץ על "Search" אז ירים את הפונקציה "movie"(Flask Search "Search" אז ירים את הפונקציה "get_film_id_by_text"). שבודקת שהקלט אינם זדוני ומחפשת אם השם שהוכנס נמצא ב- DB. אם כן אז היא תרנדר את הדף movie.html עם התכנים של הסרט, ואחרת תרנדר את הדף not_found.html שמסביר למשתמש שהסרט שהוא חיפש לא נמצא.
- אם המשתמש יכנס טקסט וילחץ על "I Feel Lucky" אז ירים את הפונקציה "movie"(Flask Search "Search" אז ירים את הפונקציה "get_film_id_by_text"). היא תתחאל את ה- DB לגבי סרט שבעלילה שלו מופיע הטקסט שהוכנס ע"י המשתמש (קוראת () get_film_id_by_text). לאחר מכן הפונקציה תרנדר את הדף movie.html עם המידע שהתקבל מה- DB. אם ה- DB יחזיר יותר מסרט אחד, אז הפונקציה תבחר סרט רנדומלי מבין אלה שהוחזרו. ואם לא נמצא סרט מתאים אז הפונקציה תבחר סרט רנדומלי מתוך ה- DB.