

Cinema Bay | מסמך תיעוד

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 שנובעות מהמזהה שלו:

<u>film_id</u>	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 לטריילר ב- YouTube
- rating: הדירוג של הסרט לפי אתר IMDb
- director: שם הבמאי

אינדקסים ומפתחות:

	Column	Type	Primary/Foreign/Index
PRIMARY	film_id	BTREE	Primary
title_index	title	BTREE	Index
year_index	year	BTREE	Index
summary_index	summary	FULLTEXT	Index
rating_index	rating	BTREE	Index
director_index	director	BTREE	Index

שיקולים בקביעת הסכמה של הטבלה: המידע שהטבלה FILM מכילה נשלף מה- APIs במספר איטרציות, ולכן בהתחלה חשבנו לייצר טבלאות קטנות מ- 2-3 עמודות, כך ש- film_id יהיה מפתח ראשי בכולן וימפה לאותם attributes שנשלפו ביחד. בסוף החלטנו לעבוד עם טבלה גדולה שתכיל את כל ה- attributes הנ"ל ביחד, וזאת מכיוון ש- film_id קובע את כולן, כך שסכמה רחבה תחסוך לנו הרבה פעולות של join כשנרצה לחלץ מידע על סרט מסוים.

אופטימיזציות: העמודות tile, year, rating ו- director של הטבלה FILM שימשו אותנו בשאילתות בפקודות WHERE, GROUP BY ו- ORDER BY. מסד הנתונים שלנו מכיל כ- 100,000 רשומות, ולכן היה חשוב לנו ליצור אינדקס לכל עמודה שהשליפה באחת מהשאילתות (השאילתות מופיעות בהמשך) מתבצעת לפיה. הוספת האינדקסים שיפרה את מהירות שליפת הנתונים מה- DB משמעותית. כמו כן, האפליקציה שלנו מבצעת חיפוש בעמודה summary המכילה מידע מטיפוס TEXT, לכן גם עבור העמודה הזאת הגדרנו אינדקס מטיפוס FULLTEXT, וזאת כדי שהחיפוש בתקציר של הסרט יתבצע מהר ככל הניתן.

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

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

• film_id: מזהה של הסרט

• location: אתר צילום

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: מכיוון שלכל סרט יש הרבה אתרי צילום, וכל אתרי צילום יכול להיות משותף עם כמה סרטים, אז שמרנו את המידע על אתרי הצילום בטבלה נפרדת. בחרנו לא לשים את המידע על אתרי הצילום בטבלה FILM כדי שלא יהיו לנו כפילויות במידע בטבלה FILM.

אופטימיזציות: אין לנו שאילתות שמחפשות לפי אתר הצילום לכן לא היה צורך שם באינדקס, אולם ישנן שאילתות שמסננות לפי ה־ film_id ולכן הוספנו שם אינדקס (ליעול פקודות join למשל).

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

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

• film_id: מזהה של הסרט

• genre: ז'אנר שהסרט משתייך אליו (לכל סרט יכולים להיות כמה כאלה)

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
genre_index	genre	BTREE	Index

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

אופטימיזציות: הוספנו אינדקס לעמודות genre ו־ film_id כדי לשפר את זמן השליפה של השאילתות שמסתמכות על הערך של העמודה genre ו/או film_id.

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 זהים לשיקולים של הטבלה FILM_LOCATION (לכל סרט יכולים להיות כמה שירותי סטרימינג מתאימים ולהפך).
אופטימיזציות: אין לנו שאילתות שמסננות רשומות לפי שירות הסטרימינג, אבל ישנן שאילתות שמסננות לפי film_id לכן הוספנו שם אינדקס.

5. FILM_AWARD - טבלה שמייצגת את הפרסים שסרט 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 זהים לשיקולים של הטבלה FILM_LOCATION (סרט יכול לזכות במספר פרסים ולהפך).
אופטימיזציות: הוספנו אינדקס עבור film_id כדי לייעל שאילתות שמסננות לפי הערך של film_id.

6. ACTOR - טבלה שמייצגת שחקן/ית עם כל ה- attributes שנובעות מהמזהה שלו:

<u>actor_id</u>	actor_name	birthdate	image
-----------------	------------	-----------	-------

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

אינדקסים ומפתחות:

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

שיקולים בקביעות הסכמה של הטבלה: הטבלה הנ"ל מייצגת שחקן/ית, ולכן דאגנו שהיא תהיה נפרדת מהטבלאות של הסרטים כדי שלא יהיו לנו כפילויות במידע בטבלאות (הקישור בין השחקנים והסרטים ייעשה ע"י join), וכדי שפעולות העדכון של הטבלאות יהיו פשוטות ככל הניתן.

אופטימיזציות: הוספנו אינדקסים לעמודות actor_name ו- birthdate, וזאת כי יש לנו שאילתות שמחפשות לפי העמודות האלה, ולכן במקרה כזה האינדקסים יכולים לשפר את זמן השליפה מהטבלה.

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

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

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

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
actor_id	actor_id	BTREE	Foreign, references actor_id of ACTOR
actor_id_index	actor_id	BTREE	Index

שיקולים בקביעות הסכמה של הטבלה: השיקולים זהים לשיקולים של הטבלה FILM_LOCATION (לסרט יכולים להיות כמה שחקנים ראשיים ולהפך).

אופטימיזציות: הוספנו אינדקסים לעמודות film_id ו- actor_id כדי לייעל את השאילתות שמסננות לפי שתי העמודות האלה.

הערה: אחרי שתיארנו את הטבלאות של מסד הנתונים נעיר שהטבלאות נקבעו כך שלא תהיה אף הפרה של BCNF, כאשר התלויות במקרה שלנו הן:

- בטבלה FILM:

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

- בטבלה ACTOR:

actor_id → actor_name, birthdate, image

לסיכום,

- שימוש בטבלאות גדולות יותר (שמכילות עוד attributes) יביא בהכרח להפרה של BCNF, מה שיגרום לשכפולי מידע מיותרים לגמרי.
- חלוקת הטבלאות שלנו לטבלאות עוד יותר קטנות תביא אף היא לשכפולי מידע, כי הרי אם נחלק את FILM (למשל) לשתי טבלאות, אז ה־ film_id יצטרך 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 בינואר, ממויינים (בסדר יורד) לפי ממוצע הדירוג של הסרטים שבהם השתתפו.

הטבלאות מתוך מסד הנתונים שהשאלתה עשתה בהם שימוש: ACTOR, FILM וכמובן הטבלה FILM_STAR שמקשרת בין הטבלאות.

2. החזרת הז'אנרים שאליהם סרט נתון משתייך ביחד עם המיקום של הסרט באותו ז'אנר (יחסית לסרטים האחרים שמשתייכים לז'אנר), 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 שהמזהה שלו הוא tt3417422 בכל אחד מהז'אנרים שלהם הסרט משתייך, כאשר התוצאה ממויינת לפי המיקומים בסדר עולה.

הטבלאות מתוך מסד הנתונים שהשאלתה עשתה בהם שימוש: FILM והטבלה FILM_GENRE שמקשרת את הסרט לז'אנרים שאליהם הוא משתייך.

3. החזרת המזהים של הסרטים שיש להם מעל x שחקנים ראשיים משותפים עם סרט נתון, ביחד עם מספר השחקנים המשותפים, כאשר התוצאה ממויינת בסדר עולה לפי שנת ההוצאה של הסרטים. השאילתה הזו עזרה לנו כדי לנחש את הסרטים שמשתייכים 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 מתוך מסד הנתונים שהשאילתה עשתה בהם שימוש: FILM והטבלה FILM_STAR שמקשרת בין הסרט והשחקנים הראשיים שהיה להם תפקיד בסרט.

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

```
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 לייעל את החיפוש בטקסט הגדרנו FULLTEXT INDEX על העמודה summary של הטבלה FILM.

הטבלאות מתוך מסד הנתונים שהשאילתה עשתה בהם שימוש: הטבלה FILM שמכילה את העמודה summary.

5. החזרת שמות השחקנים שהשתתפו בסרטים שבוימו ע"י במאי נתון ואת מספר הסרטים (של אותו במאי) שבהם הם השתתפו, כאשר התוצאה ממויינת בסדר יורד של מספר 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 בסרטים שבוימו ע"י David Yates ואת מספר הסרטים (של אותו במאי) שבהם הם השתתפו, כאשר התוצאה ממויינת בסדר יורד של מספר הסרטים.

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

6. בהנתן מזהה של סרט x , מספר שנים y ו- f שבר, להחזיר את המזהים של כל הסרטים ותמונות הפוסטר שלהם בתנאי שיש להם לפחות ז'אנר משותף אחד עם הסרט x , שהמרחק בין שנות ההוצאה שלהם ושנת ההוצאה של הסרט x קטן מ- y , והמרחק בין הדירוג שלהם לזה של x קטן מ- f . השאליתה היא שימשה אותנו כדי להמליץ למשתמש על סרטים בהתבסס 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 הם נחתכים עם הסרט שהמזהה שלו הוא `tt3417422` בז'אנר אחד, ושנת ההוצאה שלהם רחוקה משנת ההוצאה שלו ב- 4 שנים לכל היותר, ושהמרחק בדירוג ביניהם קטן מ- 0.5.

הטבלאות מתוך מסד הנתונים שהשאליתה עשתה בהם שימוש: FILM והטבלה FILM_GENRE שמקשרת את הסרט לז'אנרים שאליהם הוא משתייך.

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 שהוא השתתף בהם, ביחד עם מספר הסרטים מכל ז'אנר, כאשר התוצאה ממויינת בסדר יורד לפי מספר הסרטים.

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

8. לכל שחקן מחזירים 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 מחזירה לכל שחקן ראשי בסרט tt3417422 את המזהה, השם, תאריך הלידה, התמונה, וממוצע הדירוגים של כל הסרטים בהם הוא השתתף.

הטבלאות מתוך מסד הנתונים שהשאלתה עשתה בהם שימוש: FILM, ACTOR והטבלה FILM_STAR שמקשרת בין הסרט לשחקנים הראשיים שלו.

ייעול השאלות

כפי שמופיע בחלק הראשון על עיצוב מסד הנתונים, ניתן לראות שהוספנו אינדקס לכל עמודה שהופיעה בפקודת WHERE, GROUP BY או ORDER BY. עבור הפקודות האלה, השימוש באינדקס חוסך הרבה בזמן העיבוד, וזאת כי הפקודות האלה שולפות את המידע לפי סדר מסוים, ולכן האינדקסים יכולים לתרום תרומה משמעותית בשיפור זמן העיבוד של השאלות. נעיר כי עבור primary keys לא הגדרנו אינדקסים נוספים, כי ה־ primary key עצמו מגדיר אינדקס כזה.

מבנה הקוד

קוד הפרויקט נמצא בתיקייה SRC לפי הסדר הבא:

- SRC/APPLICATION-SOURCE-CODE - תיקייה זו מכילה את קבצי אפליקציית ה־ web המשתמשת ב־ Flask:

- server.py: תכנית פייתון שמריצה את שרת ה־ Flask

קובץ זה מכיל את כל הפונקציות שמתפעלות את השרת, מתחברות לשרת ה־ DB ומתשאלות אותו לקבלת מידע. הפונקציות הראשיות של השרת הן הפונקציות שמרנדרות את קבצי ה־ html של האתר:

* הפונקציה "home()" מרנדרת את דף הבית עם התוכן שלו

* הפונקציה "movie" מרנדרת את דף תוצאת החיפוש עם כל התכנים והאינפורמציה על הסרט המבוקש

הפונקציות שמתשאלות את ה־ DB משתמשות בסה"כ בשאילתות שתוארו בהרחבה בסעיף שמתייחס לשאילתות המרכזיות.

- static: מכיל קבצי CSS ושתי תמונות ברירת מחדל שהשתמשו בהן כשלא נמצאו תמונות מתאימות ב־ API.

- templates: מכיל את שני קבצי ה־ html שאותם אנחנו מרנדרים ואשר מרחיבים את base.html. כמו כן יש בתיקייה קובץ html עבור תוצאות חיפוש שלא הצליחו (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 (שיתוארו בהמשך) ושמידתם 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 ה־ xml משלב 1 ומכניסה את המידע למסד הנתונים.

נעיר שאת הקובץ הזה ניתן להריץ אך ורק אחרי שמייצרים את הטבלאות באמצעות הקובץ CREATE-DB-SCRIPT.sql. כמו כן, ניתן ישירות לבצע את שלב 2 ולהימנע מגישות מיותרות ל־ API אם משתמשים בקבצי ה־ XML הלוקליים שמגבים את ה־ DB.

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

- SRC/XML_FILES - תיקייה זו מכילה קבצי XML שמהווים גיבוי למידע שיש במסד הנתונים. קבצים אלה מאפשרים לנו לגבות את ה־ DB, ולהימנע מקריאות מיותרות ל־ API ומבזבז זמן מיותר על הקריאות האלה אם נרצה למלא את ה־ DB מחדש.

מקורות המידע

מסד הנתונים שלנו מכיל מידע ממקורות מידע שונים:

1. אתר האינטרנט IMDb:

קישור:

<https://www.imdb.com>

מהאתר הזה הורדנו דפי html שמכילים בין היתר את המזהים של 1000 הסרטים הכי פופולריים. מזהים אלה שימשו אותנו כדי להוציא מידע מה-APIs האחרים שאיתם עבדנו, ושהם כל סרט מזהה ע"י המזהה של הסרט ב-IMDb. כדי להוציא את המידע מדפי ה-html היינו צריכים לעשות parsing לדפי ה-html ולהוציא מהם מחרוזות שתואמות למבנה הייחודי של מזהי סרטים 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. OTT details API:

קישור:

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

הממשק הזה סיפק לנו את המידע על ה-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 קובץ ה-xml והכנסנו את המידע למסד הנתונים. הטבלה FILM_PROVIDER מאכלסת את המידע שחילצנו מ-OTT details.

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

3. IMDb8 API:

קישור:

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

הממשק הזה סיפק לנו את מרבית המידע השמור במסד הנתונים. חוץ ממזהי הסרטים, שירותי הסטרימינג והקישורים של הטריילרים, כל המידע הופק באמצעות ה-API הזה. בדומה לעבודה עם OTT details API, גם כאן בחרנו לשמור את המידע לוקלית על קבצי xml, כדי שיהיה 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> הישגנו קישורים ב-Youtube עבור הטריילרים שנמצאים במסד הנתונים. מידע זה נשמר בעמודה trailer שבטבלה FILM.

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

ספריות חיצוניות

- Flask - ספריית ה־ backend (python) שהשתמשנו בה בפרויקט. כפי שנאמר בכיתה, בחרנו ב־ Flask בגלל הפשטות שלה, ומכיוון שהפרויקט שלנו לא עסק בבניית אתר מורכב מדי אז Flask בהחלט הספיקה לצרכים שלנו.
- Bleach - ספרייה שמשמשת כדי להתמודד עם קלטים זדוניים מצד המשתמש (למשל כדי להתמודד עם SQL injection). השתמשנו בספרייה הזאת כדי למנוע מהמשתמש לחבל בבסיס הנתונים ע"י הכנסת קלט בעייתי בתור שם של סרט.
- my-sql-connector-python - ספרייה שהשתמשנו בה על מנת להתחבר לשרת ה־ DB, לתשאל אותו ולעדכן את הטבלאות.

General Flow

כשהמשתמש מתחבר לאתר אז Flask מריץ את הפונקציה `home()`. הפונקציה הזאת מתשאלת את ה־ db לקבלת תמונות ומזהים של 30 סרטים אקראיים (קוראת לפונקציה `get_movie_poster()`). כמו כן היא מתשאלת 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 פוסטר של סרט מאלה שמופיעים ברקע אז Flask יריץ את הפונקציה `"movie"`. פונקציה זו מתשאלת 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 Flask יריץ את הפונקציה `"movie()"` שבודקת שהקלט אינו 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 יכניס טקסט וילחץ על `"I Feel Lucky"` אז Flask יריץ את הפונקציה `"movie()"` אבל הפעם היא תתשאל את ה־ DB לגבי סרט שבעלילה שלו מופיע הטקסט שהוכנס ע"י המשתמש (קוראת ל־ `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 שהחוזרו. ואם לא נמצא סרט מתאים אז הפונקציה תבחר סרט רנדומלי מתוך ה־ DB.