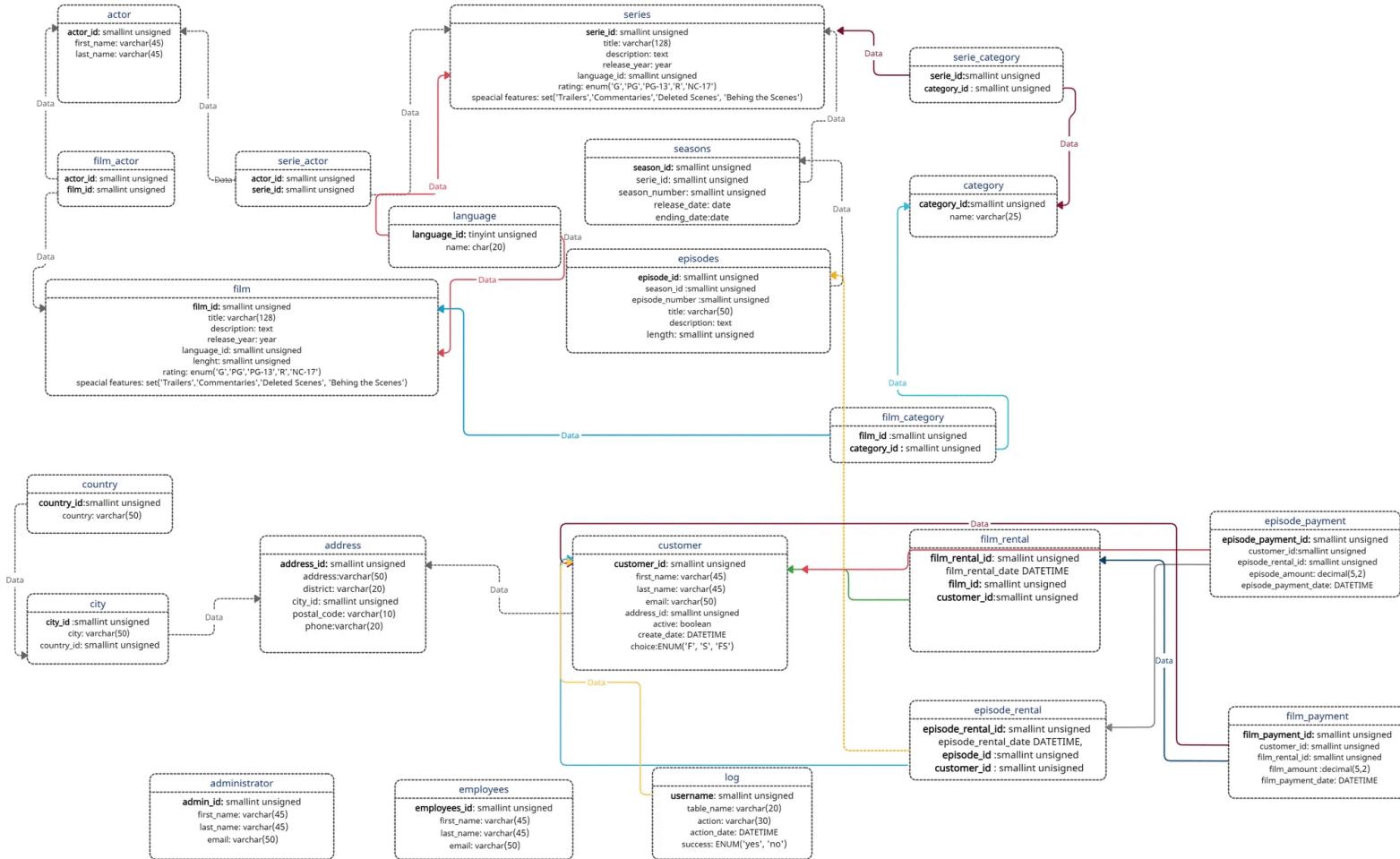


Movishop documentation(greek)



Υπόμνημα:

- [Πληροφορίες σχετικά με τον κώδικα](#)
- [Οδηγίες χρήσης της εφαρμογής](#)

IMPORTANT

*Οι οδηγίες για το β' μέρος είναι απαραίτητες για την ομαλή λειτουργία του προγράμματος και δεν πρέπει να προσπεραστούν!

Μέρος Α

- 1) Οι τρεις κατηγορίες χρηστών που δημιουργήσαμε περιγράφονται από τον παρακάτω κώδικα:

Για τους **customers** (πελάτες):

```
CREATE TABLE customer (
    customer_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    first_name VARCHAR(45) NOT NULL,
    last_name VARCHAR(45) NOT NULL,
    email VARCHAR(50) DEFAULT NULL,
    address_id SMALLINT UNSIGNED NOT NULL,
    active BOOLEAN NOT NULL DEFAULT TRUE,
    create_date DATETIME NOT NULL,
    choice ENUM('F', 'S', 'FS'),
    PRIMARY KEY (customer_id),
    CONSTRAINT fk_customer_address FOREIGN KEY (address_id) REFERENCES address (address_id)
    ON DELETE cascade ON UPDATE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Για τους **employees** (υπαλλήλους):

```
create table employees(
    employees_id smallint unsigned not null AUTO_INCREMENT,
    first_name varchar(45) not null,
    last_name varchar(45) not null,
    email varchar(50) not null,
    PRIMARY KEY(employees_id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Για τους **administrator** (διαχειριστές):

```
create table administrator(
    admin_id smallint unsigned not null AUTO_INCREMENT,
    first_name varchar(45) not null,
    last_name varchar(45) not null,
    email varchar(50) not null,
    PRIMARY KEY(admin_id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Κάθε πελάτης χάρη στο πεδίο τύπου **enum** εγγράφεται για να βλέπει μόνο ταινίες,σειρές ή και τα δύο.Τα κόστη υπολογίζονται αργότερα στο **backend**(server) με την μορφή **object** το οποίο θα περιγράψουμε στο μέρος Β'. Το σχεσιακό μοντέλο φαίνεται στην αρχή αυτού του doc και περιλαμβάνεται και στο rar αρχείο.

- 2) Για να επιτευχθεί η προσθήκη σειρών έπρεπε να κάνουμε τροποποιήσεις στην υπάρχουσα βάση δεδομένων όπως το να αφαιρέσουμε τον πίνακα `inventory`, ο οποίος πλέον έχει ενταχθεί στους `rental` και `payment` για τους οποίους έχουμε φτιάξει έναν για τις σειρές(για τα επεισόδια των σειρών) και έναν για τις ταινίες.

Προσθήκη σειρών:

Για να ενσωματώσουμε σειρές στην υπηρεσία(ονομάστηκε movieshop) φτιάξαμε έναν πίνακα για τις σειρές(series),ένα για τους κύκλους(seasons) και ένα για τα επεισόδια κάθε κύκλου σειράς(episodes). Πιο συγκεκριμένα:

Ο πίνακας `series`:

```
CREATE TABLE series (
    serie_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    title VARCHAR(128) NOT NULL,
    description TEXT DEFAULT NULL,
    release_year YEAR DEFAULT NULL,
    language_id TINYINT UNSIGNED DEFAULT NULL,
    rating ENUM('G', 'PG', 'PG-13', 'R', 'NC-17') DEFAULT 'G',
    special_features SET('Trailers', 'Commentaries', 'Deleted Scenes', 'Behind the Scenes')
    DEFAULT NULL,
    PRIMARY KEY (serie_id),
    CONSTRAINT fk_serie_language FOREIGN KEY (language_id)
        REFERENCES language (language_id) ON DELETE cascade ON UPDATE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Ο πίνακας `seasons`:

```
CREATE TABLE seasons(
    season_id SMALLINT unsigned NOT NULL,
    serie_id SMALLINT UNSIGNED NOT NULL,
    season_number SMALLINT unsigned NOT NULL,
    release_date date,
    ending_date date,
    PRIMARY KEY(season_id),
    constraint fk_serie_id foreign key (serie_id) REFERENCES series(serie_id) ON DELETE
    CASCADE on update cascade
)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Ο πίνακας `episodes`:

```
CREATE TABLE episodes(
    episode_id SMALLINT UNSIGNED NOT NULL,
    season_id smallint unsigned NOT NULL,
    episode_number SMALLINT UNSIGNED NOT NULL,
    title VARCHAR(50) NOT NULL,
    description text,
    length SMALLINT UNSIGNED DEFAULT NULL,
    PRIMARY KEY(episode_id),
    constraint fk_season_id FOREIGN KEY (season_id) REFERENCES seasons(season_id) on
    delete cascade on update CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Επιπλέον εκτός από τους πίνακες `episode_rental`, `episode_payment`, `film_rental` και `film_payment` που κατασκευάσαμε, δημιουργήσαμε πίνακες `film_category` και `serie_category` οι οποίες χρησιμοποιούν τον πίνακα `category` ώστε να κρατούν το `category` κάθε σειράς και ταινίας, σε αντίστοιχο ύφος δημιουργήθηκαν και οι πίνακες `film_actor` και `serie_actor`. Αναλυτικότερα:

`episode_rental`:

```
create table episode_rental(
    episode_rental_id SMALLINT UNSIGNED NOT NULL auto_increment,
    episode_rental_date DATETIME,
    episode_id SMALLINT UNSIGNED NOT NULL,
    customer_id SMALLINT UNSIGNED NOT NULL,
    primary key (episode_rental_id,episode_id,customer_id),
    constraint fk_customer_ep_id FOREIGN KEY (customer_id) REFERENCES customer(customer_id)
ON DELETE CASCADE on update cascade,
    CONSTRAINT fk_inventory foreign key (episode_id) references episodes(episode_id) on
    delete cascade on update cascade

)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

episode_payment:

```
create table episode_payment(
    episode_payment_id smallint unsigned auto_increment ,
    customer_id smallint unsigned NOT NULL,
    episode_rental_id smallint unsigned NOT NULL,
    episode_amount decimal(5,2) NOT NULL,
    episode_payment_date DATETIME NOT NULL,
    PRIMARY KEY (episode_payment_id),
    constraint fk_customer_epay_id FOREIGN KEY (customer_id) REFERENCES
customer(customer_id) ON DELETE CASCADE on update cascade,
    constraint fk_episode_rental_id FOREIGN KEY (episode_rental_id) REFERENCES
episode_rental(episode_rental_id) on delete CASCADE on update cascade
)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

film_rental:

```
create table film_rental (
    film_rental_id smallint NOT NULL AUTO_INCREMENT,
    film_rental_date DATETIME,
    film_id smallint unsigned NOT NULL,
    customer_id smallint unsigned NOT NULL,
    primary key (film_rental_id,film_id,customer_id),
    constraint fk_customer_id FOREIGN KEY (customer_id) REFERENCES customer(customer_id) ON
DELETE CASCADE on update cascade,
    constraint fk_film_inventory foreign key (film_id) REFERENCES film(film_id) ON DELETE
CASCADE ON UPDATE cascade
)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

film_payment:

```
create table film_payment (
    film_payment_id smallint auto_increment,
    customer_id smallint unsigned NOT NULL,
    film_rental_id smallint not null,
    film_amount decimal(5,2) not null,
    film_payment_date DATETIME,
    PRIMARY KEY (film_payment_id),
    constraint fk_customer_fp_id FOREIGN KEY (customer_id) REFERENCES
customer(customer_id) ON DELETE CASCADE on update cascade,
    constraint fk_film_rental_id FOREIGN KEY (film_rental_id) REFERENCES
film_rental(film_rental_id) on delete CASCADE on update cascade
)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

film_category:

```
CREATE TABLE film_category (
    film_id SMALLINT UNSIGNED NOT NULL,
    category_id smallint UNSIGNED NOT NULL,
    PRIMARY KEY (film_id,category_id),
    CONSTRAINT fk_film_id FOREIGN KEY (film_id) REFERENCES film (film_id) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_film_category_category FOREIGN KEY (category_id) REFERENCES category (category_id) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

serie_category:

```
CREATE TABLE serie_category (
    serie_id SMALLINT UNSIGNED NOT NULL,
    category_id smallint UNSIGNED NOT NULL,
    PRIMARY KEY (serie_id,category_id),
    CONSTRAINT fk_serie_category_film FOREIGN KEY (serie_id) REFERENCES series (serie_id) ON DELETE cascade ON UPDATE CASCADE,
    CONSTRAINT fk_serie_category_category FOREIGN KEY (category_id) REFERENCES category (category_id) ON DELETE cascade ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Γεμίζουμε τους πίνακες με το αρχείο sql **insert_tables**.

Όλοι οι πίνακες βρίσκονται στο αρχείο sql **create_tables**

3) Stored procedures:

1) Ο κώδικας της procedure που επιστρέφει τις πέντε πρώτες ταινίες ή σειρές με τις περισσότερες ενοικιάσεις είναι:

```
CREATE PROCEDURE `most_rent` (IN lt CHAR(1), IN number INT, IN imerominia1 DATE, IN imerominia2 DATE)
BEGIN
    IF(lt like 'm') THEN
        SELECT film.title, film.film_id,film.release_year
        FROM film
        INNER JOIN film_rental ON film_rental.film_id = film.film_id
        WHERE film_rental.film_rental_date BETWEEN imerominia1 AND imerominia2
        GROUP BY film_id
```

```

ORDER BY count(*) DESC
LIMIT number;

ELSEIF(lt like 's') THEN
    SELECT series.title, series.serie_id,series.release_year,count(*)
    FROM series INNER JOIN seasons ON series.serie_id = seasons.serie_id
    INNER JOIN episodes ON episodes.season_id = seasons.season_id
    INNER JOIN episode_rental ON episode_rental.episode_id= episodes.episode_id
    WHERE episode_rental.episode_rental_date BETWEEN imerominia1 AND imerominia2
    GROUP BY series.serie_id
    ORDER BY count(*) DESC
    LIMIT number;

ELSE
    SELECT 'wrong input choice';

END IF;
END$
```

Αρχικά παίρνουμε σαν παραμετρούς μια μεταβλητή με την οποία διαχωρίζουμε αν θα ψάξουμε για σειρές ή ταινίες τύπου **char**, μια τύπου **int** η οποία αντιπροσωπεύει τον αριθμό που θέλουμε να ψάξουμε και άλλες δύο για την ημερομηνία. Έπειτα ανάλογα με τον char χαρακτήρα που δώσαμε θα κάνει **select** από τον πίνακα **films** ή **series** αντίστοιχα(αν είναι 'm' ή 's') έπειτα με **inner joins** θα πάρει τα στοιχεία από τους πίνακες **rental**. (Αυτη την procedure χρησιμοποιούμε για να πάρουμε τις 5 πρώτες ταινίες και σειρές στο μέρος β)

Χαρακτηριστικό είναι και το παρακάτω παράδειγμα με τα αποτελέσματα που τυπώνονται:

```
call most_rent('m', 5, '2005-03-02', '2006-03-02');
```

	title	film_id
1	TRADING PINOCCHIO	902
2	STING PERSONAL	846
3	NECKLACE OUTBREAK	618
4	SUGAR WONKA	859
5	GREEDY ROOTS	379

2) (3.2 procedure)

```
CREATE PROCEDURE `get_rental_of_customer` (IN email_c VARCHAR(50), IN imerominia DATE)
BEGIN
    DECLARE id SMALLINT;
    DECLARE film_rental_of_day INT;
    DECLARE serie_rental_of_day INT;
    DECLARE film_rental_day INT;
    DECLARE episode_rental_day INT;
    DECLARE c VARCHAR(10);

    SELECT choice INTO c FROM customer where email LIKE email_c;
    SELECT customer_id INTO id FROM customer WHERE email LIKE email_c;

    IF(c LIKE 'F') then
        SELECT customer.customer_id, count(*) AS film_rental_day FROM customer
        INNER JOIN film_rental ON customer.customer_id = film_rental.customer_id
        WHERE film_rental_date between CONCAT(imerominia, " 00:00:00") and CONCAT(imerominia, "
23:59:59") AND customer.customer_id = id;

    ELSEIF(c like 'S') then
        SELECT customer.customer_id, count(*) AS episode_rental_day FROM customer
        INNER JOIN episode_rental ON customer.customer_id = episode_rental.customer_id
        WHERE episode_rental_date between CONCAT(imerominia, " 00:00:00") and
CONCAT(imerominia, " 23:59:59") AND customer.customer_id = id;

    ELSEIF(c like 'FS') then
        SELECT count(*) INTO film_rental_of_day FROM customer
        INNER JOIN film_rental ON customer.customer_id = film_rental.customer_id
        WHERE film_rental_date between CONCAT(imerominia, " 00:00:00") and CONCAT(imerominia, "
23:59:59") AND customer.customer_id = id;
        SELECT count(*) INTO serie_rental_of_day FROM customer
        INNER JOIN episode_rental ON customer.customer_id = episode_rental.customer_id
        WHERE episode_rental_date between CONCAT(imerominia, " 00:00:00") and
CONCAT(imerominia, " 23:59:59") AND customer.customer_id = id;
        SELECT 'customer has rent: ', film_rental_of_day+serie_rental_of_day AS
FilmSerie_rental_day;

    ELSE
        SELECT 'error, email doesnt exist' AS message;

    END IF;
END
```

Αρχικά αποθηκεύουμε σε δύο μεταβλητές το **choice** και το **customer_id** τα οποία επιστρέφουν τις ζητούμενες τιμές με βάση την επιλογή του email ώστε μέσω **if/else** conditions να δούμε την επιλογή που έχει κάνει ο χρήστης ώστε να μπούμε

στο κατάλληλο if και να επιστρέψουμε τις ενοικιάσεις που έχει κάνει μια συγκεκριμένη ημερομηνία ελέγχοντας μέσω του **customer_id** εάν είναι του ζητούμενου πελάτη.

Αποτελέσματα:

```
call get_rental_of_customer('SANDRA.MARTIN@sakilacustomer.org', '2003-02-13');
```

		customer has rent:	FilmSerie_rental_day
		Filter	Filter
	1	customer has rent:	2

3) (3.3 procedure)

```
CREATE PROCEDURE `income`()
BEGIN
    SELECT t1.m month,
           COALESCE(t1.incomeFilm, 0) incomeFilm,
           COALESCE(t2.incomeSerie, 0) incomeSerie
    FROM
    (
        SELECT
            MONTH(film_payment_date) m,
            COALESCE(sum(film_amount),0) incomeFilm
        FROM film_payment
        GROUP BY MONTH(film_payment_date)
        ORDER BY MONTH(film_payment_date) ASC
    ) t1
    LEFT JOIN
    (
        SELECT
            MONTH(episode_payment_date) m,
            COALESCE(sum(episode_amount),0) incomeSerie
        FROM episode_payment
        GROUP BY MONTH(episode_payment_date)
        ORDER BY MONTH(episode_payment_date) ASC
    ) t2 ON t1.m = t2.m
    UNION
    SELECT t2.m month,
           COALESCE(t1.incomeFilm, 0) incomeFilm,
           COALESCE(t2.incomeSerie, 0) incomeSerie
```

```

FROM
(
    SELECT
        MONTH(film_payment_date) m,
        COALESCE(sum(film_amount),0) incomeFilm
    FROM film_payment
    GROUP BY MONTH(film_payment_date)
    ORDER BY MONTH(film_payment_date) ASC
) t1
RIGHT JOIN
(
    SELECT
        MONTH(episode_payment_date) m,
        COALESCE(sum(episode_amount),0) incomeSerie
    FROM episode_payment
    GROUP BY MONTH(episode_payment_date)
    ORDER BY MONTH(episode_payment_date) ASC
) t2 ON t1.m = t2.m;
END

```

Όσο αναφορά το **call income()** χρησιμοποιήσαμε **UNION** και **left/right join** ώστε να πάρουμε κάθε μήνα στον οποίο είχαμε έσοδα είτε αυτός ο μήνας προέκυπται από τα **film** είτε από τα **episodes** είτε και από τα δύο μαζί.

Αποτελέσματα:

```
call income();
```

	month	incomeFilm	incomeSerie
1	1	10.98	0.00
2	2	10.97	0.20
3	3	2.99	0.00
4	5	44.89	0.60
5	6	61.85	0.20
6	7	161.68	1.40
7	8	126.70	1.20
8	4	0.00	0.40

4) (3.4 procedure)

a)

```
CREATE PROCEDURE `get_actors` (IN name1 VARCHAR(20), IN name2 VARCHAR(20))
BEGIN
    select first_name, last_name FROM actor
    WHERE last_name BETWEEN name1 AND name2
    ORDER BY last_name;
END
```

Σε αυτή την περίπτωση πήραμε ώς είσοδο δύο επωνύματα ή κάποια αρχικά και επιστρέψαμε κάθε επώνυμο ανάμεσα τους.

Αποτελέσματα:

```
CALL get_actors('Aco', 'Alm');
```

	first_name	last_name
1	Emery	Aguirre
2	Elsie	Aguirre
3	Simeon	Allen

b)

```
CREATE PROCEDURE `get_actor1` (IN eponumo VARCHAR(20))
BEGIN
    DECLARE plithos_same_actor INT;
    SELECT first_name, last_name FROM actor
    WHERE last_name LIKE eponumo;
    SELECT found_rows() INTO plithos_same_actor;
```

```

IF(plithos_same_actor = 0) THEN
    SELECT 'NO SAME ACTOR WITH THIS LAST NAME';

END IF;

END

```

Εδώ πήραμε ως είσοδο ένα επώνυμο και επιστρέψαμε κάθε ηθοποιό που έχει το ίδιο επώνυμο, μέσω της **found_rows()** προσδιορίζουμε τι χρειαζόμαστε ακόμα για το υπόλοιπο αποτέλεσμα και έπειτα μέσω της select στο **found_rows()** αποθηκεύουμε σε μια μεταβλητή **INT** το πλήθος των ηθοποιών που επιστρέψαμε και σε περιπτωση που δεν υπάρχει κανείς ηθοποιός επιστρέφουμε το ανάλογο μήνυμα.

Αποτελέσματα:

```
CALL get_actor1('Horn');
```

	first_name	last_name
	Filter	Filter
1	Nylah	Horn
2	Gabrielle	Horn

Ολες οι procedures βρίσκονται στο αρχείο sql **procedure**

4) Triggers

(Επειδή κάποιοι triggers περιείχαν και άλλους μέσα (όπως ο 4.1 και 4.2) θεωρήσαμε σκόπιμο να μην βάλουμε τον κώδικα στην παρακάτω αιτιολόγηση αλλα υπάρχει στο αρχείο sql **triggers**)

1) (4.1 trigger)

Θα παραθέσουμε τα αποτελέσματα των εκτελέσεων των trigger για insert,delete,update στον πίνακα film_rental. Για τους υπόλοιπους 3 πίνακες ακολουθήθηκε παρόμοιος κώδικας για την υλοποίηση.

Αποτελέσματα:

Για insert:

```
INSERT INTO film_rental VALUES  
(17000, '2005-08-21 11:13:35', 23, 16);  
  
select * from log;
```

select * from log;						
Input To Search Data						
	Q	*username varchar(50)	*table_name varchar(20)	action varchar(30)	action_date datetime	success enum('yes','no')
	1	SANDRA.MARTIN@sakil	film_rental	insert	2022-02-13 18:14:09	yes

Για delete:

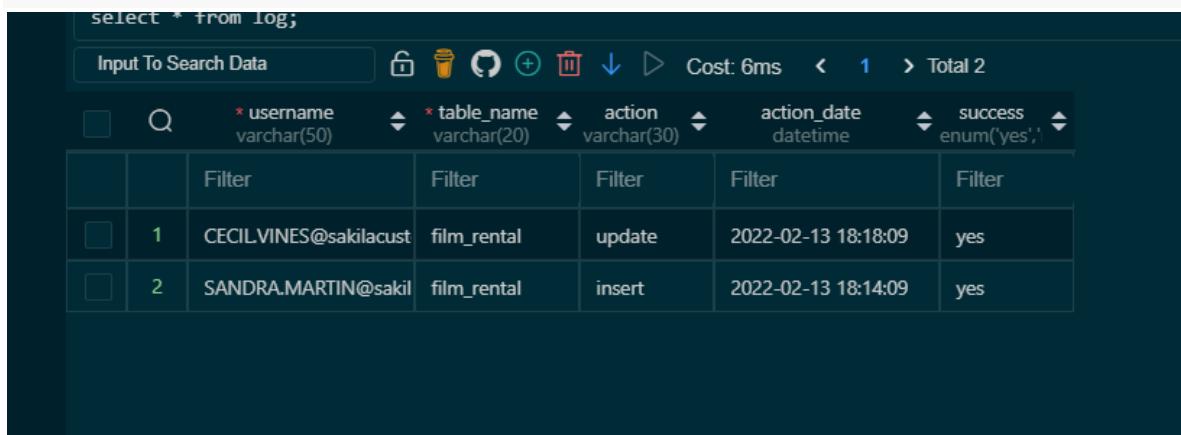
```
DELETE FROM film_rental where film_rental_id=5664;
```

```
select * from log;
```

SELECT * FROM log LIMIT 100;						
Input To Search Data						
	Q	*username varchar(50)	*table_name varchar(20)	action varchar(30)	action_date datetime	success enum('yes','no')
	1	CECIL.VINES@sakila	customer	update	2022-02-13 18:26:32	yes
	2	SANDRA.MARTIN@sakil	film_rental	insert	2022-02-13 18:26:11	yes
	3	WESLEY.BULL@sakila	film_rental	delete	2022-02-13 18:28:22	yes

Για update:

```
UPDATE film_rental  
SET film_rental_id= 2028  
where film_rental_id = 2029;  
  
select * from log;
```



The screenshot shows the MySQL Workbench interface. At the top, there is a command window with the following SQL code:

```
UPDATE film_rental  
SET film_rental_id= 2028  
where film_rental_id = 2029;  
  
select * from log;
```

Below the command window is a results grid titled "Input To Search Data". The results show two rows of data from the "log" table:

	Q	* username varchar(50)	* table_name varchar(20)	action varchar(30)	action_date datetime	success enum('yes')
	1	CECILVINES@sakilacust	film_rental	update	2022-02-13 18:18:09	yes
	2	SANDRA.MARTIN@sakil	film_rental	insert	2022-02-13 18:14:09	yes

2) (4.2 trigger)

Αρχικά για την υλοποίηση αυτού του trigger κατασκευάσαμε μια νέα stored procedure(**set_amount**) την οποία δεν ζητόταν. Η συγκεκριμένη stored procedure θέτει τις ζητούμενες τιμές κόστους στα **film**,**series** με βάση την εκφώνηση. Την δημιουργήσαμε ώστε να την καλέσουμε στον trigger και να κάνει την ανάλογη έκπτωση αν χρειάζεται αλλιώς να τον χρεώνει κανονικά την αρχική τιμή. Για την υλοποίηση τον trigger για το **film_rental**, **episode_rental** φτιάξαμε παρόμοιο κώδικα. Τα αποτελέσματα είναι τα εξης:

Ο συγκεκριμένος customer=33 έχει εγγραφεί μονο για ταινίες

```
INSERT INTO film_rental VALUES  
(16700, '2005-08-21', 691, 33);
```

```
select * from film_payment;
```

	Q	* film_payment_id smallint	* customer_id smallint unsigned	* film_rental_id smallint	* film_amount decimal(5,2)	film_payment_date datetime
	78	13403	497	15919	0.99	2005-08-23 18:01:31
	79	13404	497	12698	4.99	2006-02-14 15:16:03
	80	13433	499	1355	2.99	2005-06-15 13:13:59
	81	13782	512	2028	4.99	2005-06-17 13:10:59
	82	13806	512	12786	0.99	2006-02-14 15:16:03
	83	14119	525	3993	6.99	2005-07-06 23:37:06
	84	14738	549	8316	4.99	2005-07-29 03:38:49
	85	14891	556	772	5.99	2005-05-29 13:08:06
	86	14892	556	1083	3.99	2005-05-31 11:04:48
	87	14895	556	2986	0.99	2002-06-20 08:50:28
	88	14909	556	14176	2.99	2005-08-21 03:09:23
	89	15623	583	5462	0.99	2005-07-09 22:56:53
	90	15633	583	10800	4.99	2005-08-01 22:07:44
	91	15967	596	6197	4.99	2005-07-11 12:09:51
	92	15980	596	14417	0.99	2005-08-21 11:13:35
	93	15982	16	17000	0.30	2022-02-11 00:00:00
	94	15985	33	16700	0.20	2022-02-11 00:00:00

* Η τελευταία γραμμή του πίνακα μας δείχνει ότι του έκανε έκπτωση στην ταινία 50% (από 0.4 πήραμε 0.2)

Ο συγκεκριμένος customer=16 εχει εγγραφεί και για ταινίες και για σειρές

```
INSERT INTO episode_rental VALUES  
(15500, '2003-02-13',40,16);
```

```
select * from episode_payment;
```

*Η τελευταία γραμμή μου επιστρέφει το αποτέλεσμα. Έχει γίνει έκπτωση στο επεισόδιο 50% (από 0.10 πήραμε 0.05)

	* episode_payment_id smallint unsigned	* customer_id smallint unsigned	* episode_rental_id smallint unsigned	* episode_amount decimal(5,2)	* episode_payment_da datetime
4	1030	33	300	0.20	2001-07-07 12:10:24
5	1040	52	400	0.20	2005-07-06 10:04:55
6	1050	142	500	0.20	2005-07-07 15:29:35
7	1060	162	600	0.20	2005-07-07 13:16:55
8	1070	195	700	0.20	2001-05-30 06:16:06
9	1080	512	800	0.20	2005-08-21 03:09:23
10	2000	512	900	0.20	2005-05-31 11:04:48
11	2100	596	1000	0.20	2005-05-29 13:08:06
12	2200	596	2000	0.20	2005-04-29 07:45:00
13	2300	402	3000	0.20	2005-08-01 22:07:44
14	2400	309	4000	0.20	2005-07-09 22:56:53
15	2500	309	5000	0.20	2006-02-14 15:16:03
16	2600	227	6000	0.20	2005-06-17 13:10:59
17	2700	252	7000	0.20	2005-07-27 07:51:11
18	3000	512	8000	0.20	2005-07-07 20:09:01
19	3200	309	9000	0.20	2005-08-22 18:22:44
20	3400	142	15422	0.20	2005-08-21 19:39:28
21	3401	16	15500	0.05	2022-02-11 00:00:00

3) (4.3 trigger)

```
CREATE TRIGGER `BLOCK_UPDATE`
BEFORE UPDATE ON `customer` FOR EACH ROW BEGIN
    IF(new.email != old.email) Then
        SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'Not allow to change your email';
    END IF;
END$
```

Σε περίπτωση που πάει να αλλάξει το email του ο πελάτης βγάζει μήνυμα ότι δεν επιτρέπεται

Αποτέλεσμα:

```
UPDATE customer
SET email = 'anna.WILLIAMS@sakilacustomer.org'
WHERE email = 'LINDA.WILLIAMS@sakilacustomer.org';
```

EXECUTE FAIL

UPDATE customer SET email = 'anna WILLIAMS@sakilacustomer.org' WHERE email = 'LINDA WILLIAMS@sakilacustomer.org';

Message :

Not allow to change your email

Μέρος Β

Προσοχή: Τα παρακάτω βήματα δεν πρέπει να παραληφθούν!

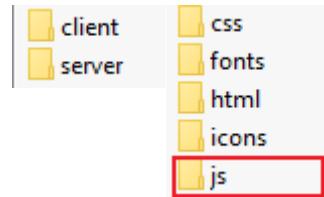
Για το μέρος β αποφασίσαμε να φτιάξουμε μια εφαρμογή η οποία περιλαμβάνει **server** που επικοινωνεί με την βάση δεδομένων(γραμμένος σε **node javascript**) και το γραφικό κομμάτι και η εφαρμογή γραμμένη σε **html/css και javascript** χρησιμοποιώντας ένα module για να δημιουργούμε desktop-apps που είναι το **electron**(σ' αυτο το module έχουν γραφεί εφαρμογές όπως το spotify, vs-code, discord etc..).

Οδηγίες για να τρέξουμε την εφαρμογή:

Αρχικά απαραίτητη προϋπόθεση για την λειτουργία της εφαρμογής είναι να είναι εγκατεστημένο το περιβάλλον **node** (javascript compiler) το οποίο μας δίνει πρόσβαση στα **node-modules** που έχουμε χρησιμοποιήσει και μας επιτρέπει να τρέχουμε javascript locally στο pc μας. [Node.js \(nodejs.org\)](https://nodejs.org)

Αφου γίνει η εγκατάσταση του περιβάλλοντος **node js** ανοίγουμε το rar και βλέπουμε ότι η εφαρμογή αποτελείται από δύο φακέλους έναν για το **server** και έναν για τον **client**(ο **server** χρησιμοποιεί sockets για την επικοινωνία με τον **client**).

Ανοίγουμε δύο **terminal** και πηγαίνουμε στο φάκελο που έχουμε τον **server** στο ένα και τον φάκελο **js**(ανήκει στον φάκελο client) στο άλλο(με εντολές **cd**).



Γράφουμε την εντολή **npm init** (και στα δύο **terminals**)η οποία θα δημιουργήσει τον φάκελο **node-modules** και έπειτα την εντολή **npm install** ή οποια θα κατεβάσει όλα τα **node-modules** που έχουμε χρησιμοποιήσει (σύμφωνα με το **package.json**). Αφού τελειώσει η εγκατάσταση, πρέπει στο **config.js** αρχείο να βάλουμε τα credentials της βάσης(αυτό το **object** θα χρησιμοποιηθεί για σύνδεση της βάσης με τον **server** με το **module mysql**)

```
let config = {  
    host:"localhost",  
    user:"root",  
    database:"tvondemand",  
    password:"password",  
};  
  
module.exports = config;
```

Κάνουμε τις απαραίτητες αλλαγές στο αρχείο **config.js** που φαίνεται και παραπάνω θέτοντας ως password τον κωδικό που έχουμε θέσει στην βάση user...etc. Επιπλέον αν έχουμε παλιά έκδοση sql θα πρέπει να τρέξουμε στην **mysql** τον παρακάτω κώδικα:

```
ALTER USER 'root'@'localhost' IDENTIFIED with mysql_native_password by 'password';
```

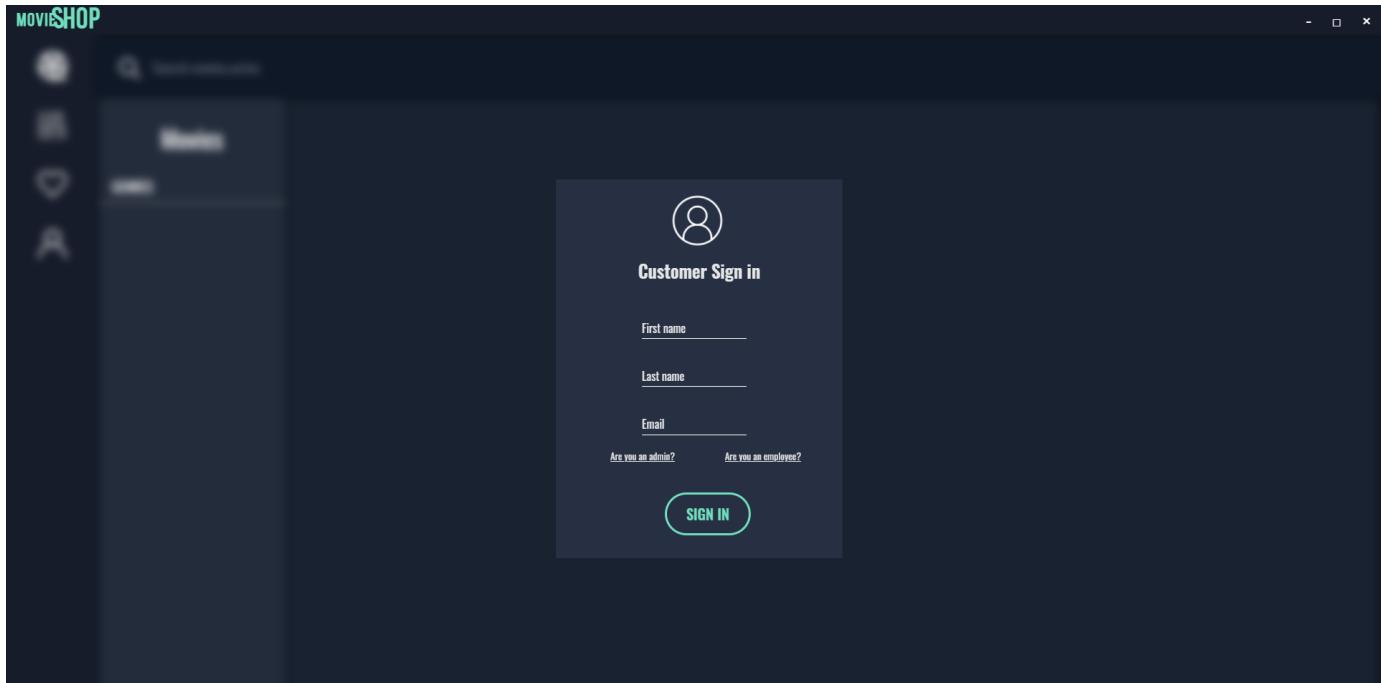
Όπου **root** ο **user**, **localhost** το **host** και **password** το **password** που έχουμε δώσει στο **config.js**.

Τώρα είμαστε έτοιμοι να τρέξουμε την εφαρμογή:

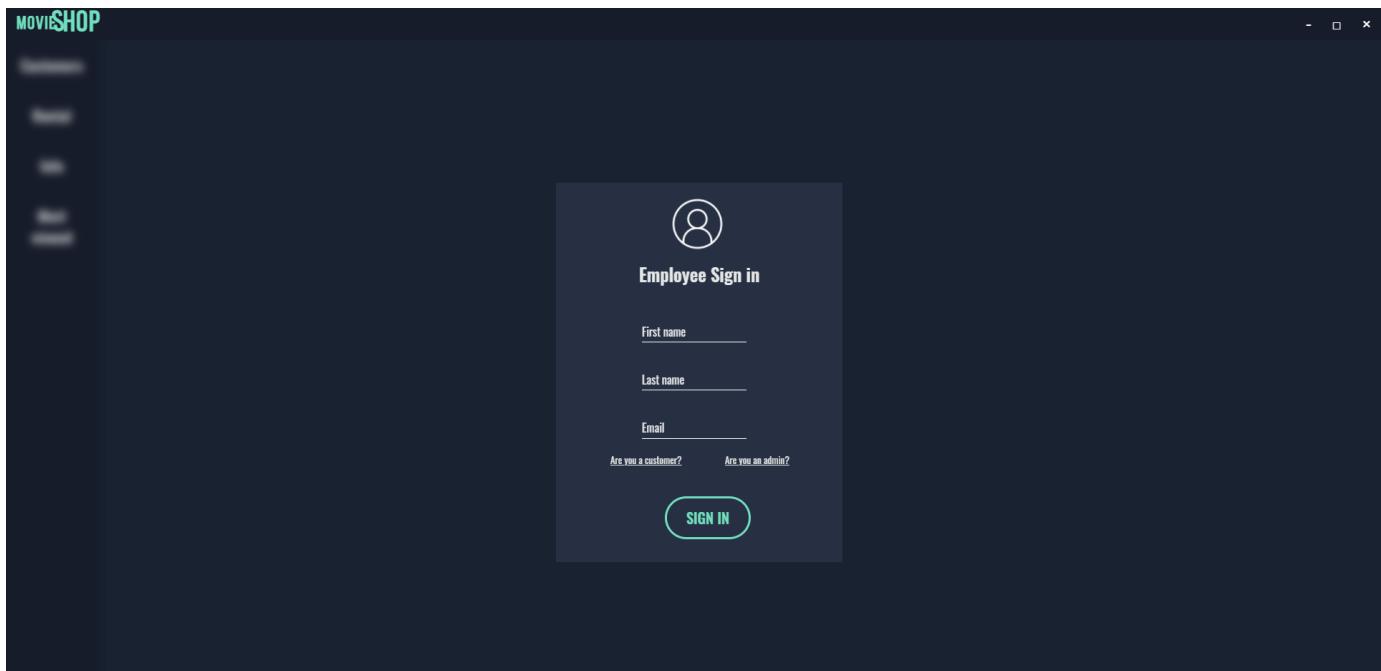
Γράφουμε αρχικά **node server.js** στο terminal που δείχνει στον **server** και **npm start** σ' αυτό που δείχνει στον **client** (dirname/client/js). Αν όλα τα βήματα έχουν τελεστεί σωστά θα πρέπει το terminal του server να γράφει:

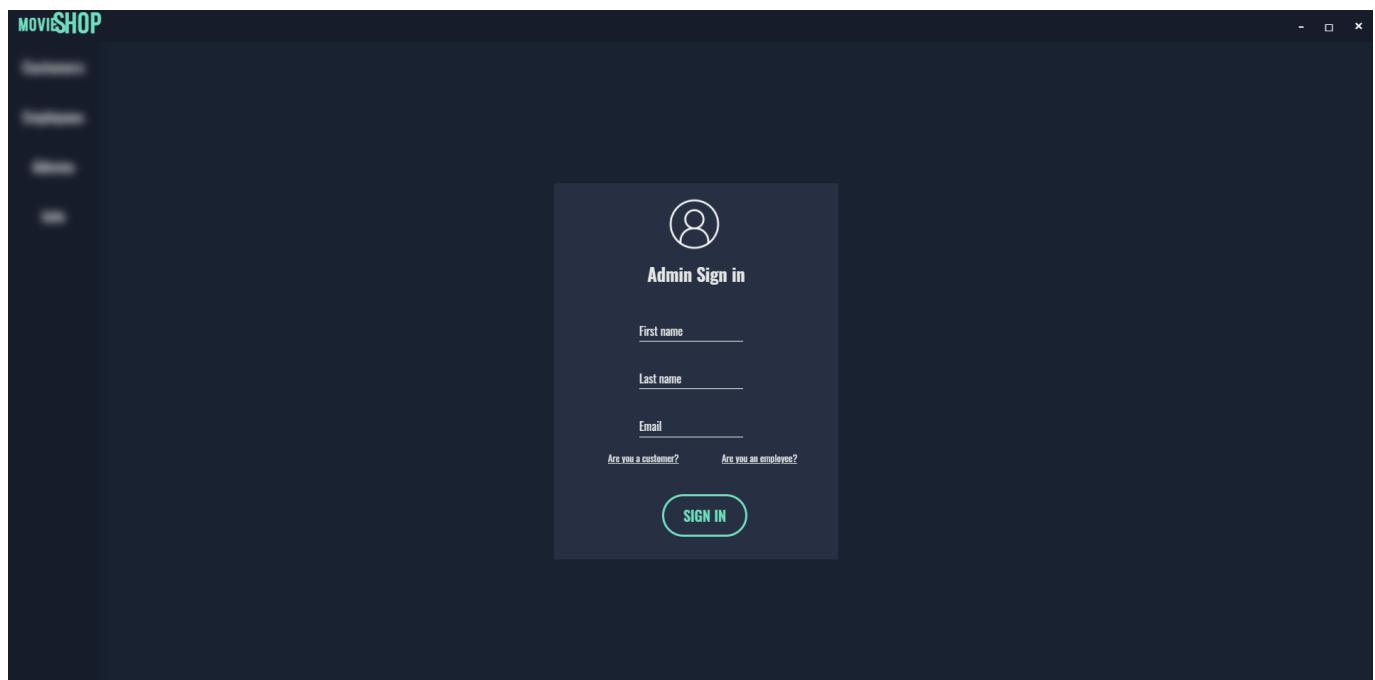
```
Server running on port : 5000
Connected to sql database!
```

Και θα πρέπει να έχει ανοίξει το πρόγραμμα και να φαίνεται το παρακάτω interface:



Για να συνδεθούμε βάζουμε τα credentials είτε ως **customer** είτε ως **administrator** είτε ως **employee** αφού έχουμε επιλέξει και το κατάλληλο κουμπί και τυπώνεται μήνυμα επιτυχίας ή αποτυχίας αντίστοιχα.



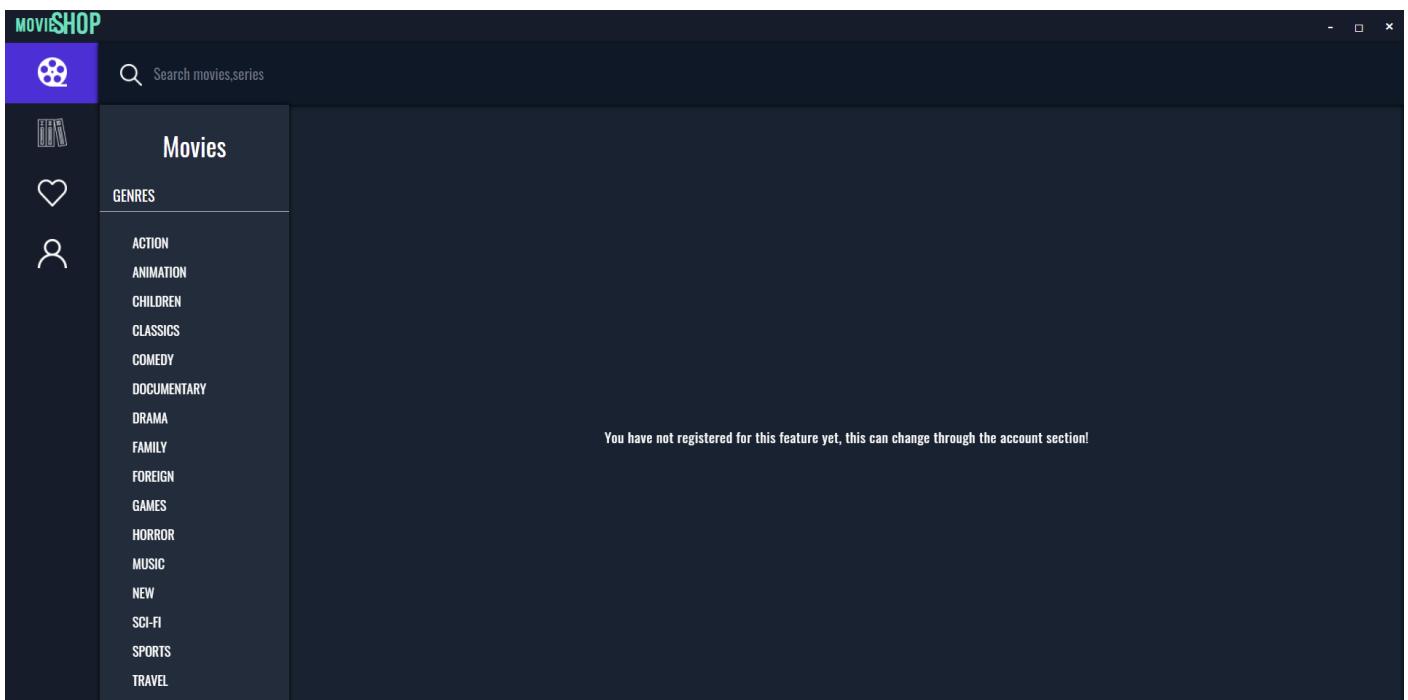


Customer:

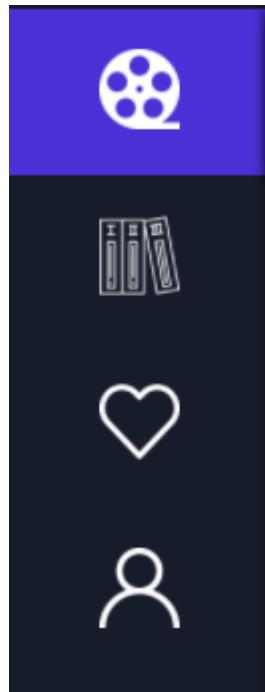
Με την επιτυχή σύνδεση του χρήστη θα εμφανιστεί το περιεχόμενο των ταινιών αν αυτός ο χρήστης έχει δικαιώμα για ταινίες

A screenshot of the "MOVIESHOP" application window from the previous image, but from the perspective of a regular customer. The left sidebar shows movie-related icons: a film reel, a stack of books, a heart, and a person. The "Movies" section is selected, showing a list of genres: ACTION, ANIMATION, CHILDREN, CLASSICS, COMEDY, DOCUMENTARY, DRAMA, FAMILY, FOREIGN, GAMES, HORROR, MUSIC, NEW, SCI-FI, SPORTS, and TRAVEL. The main content area displays a grid of movie thumbnails under the heading "68 MOVIES". Each thumbnail includes the movie title and year. The titles and years visible are: ANACONDA CONFESSIONS (2006), ANTITRUST TOMATOES (2006), ANYTHING SAVANNAH (2006), ARTIST COLD BLOODED (2006), BAREFOOT MANCHURIAN (2006), BEAST HUNCHBACK (2006), BERETS AGENT (2006), CHEAPER CLYDE (2006), COAST RAINBOW (2006), CONQUERER NUTS (2006), CROSSROADS CASUALTIES (2006), and CURTAIN VIDEOTAPE (2006).

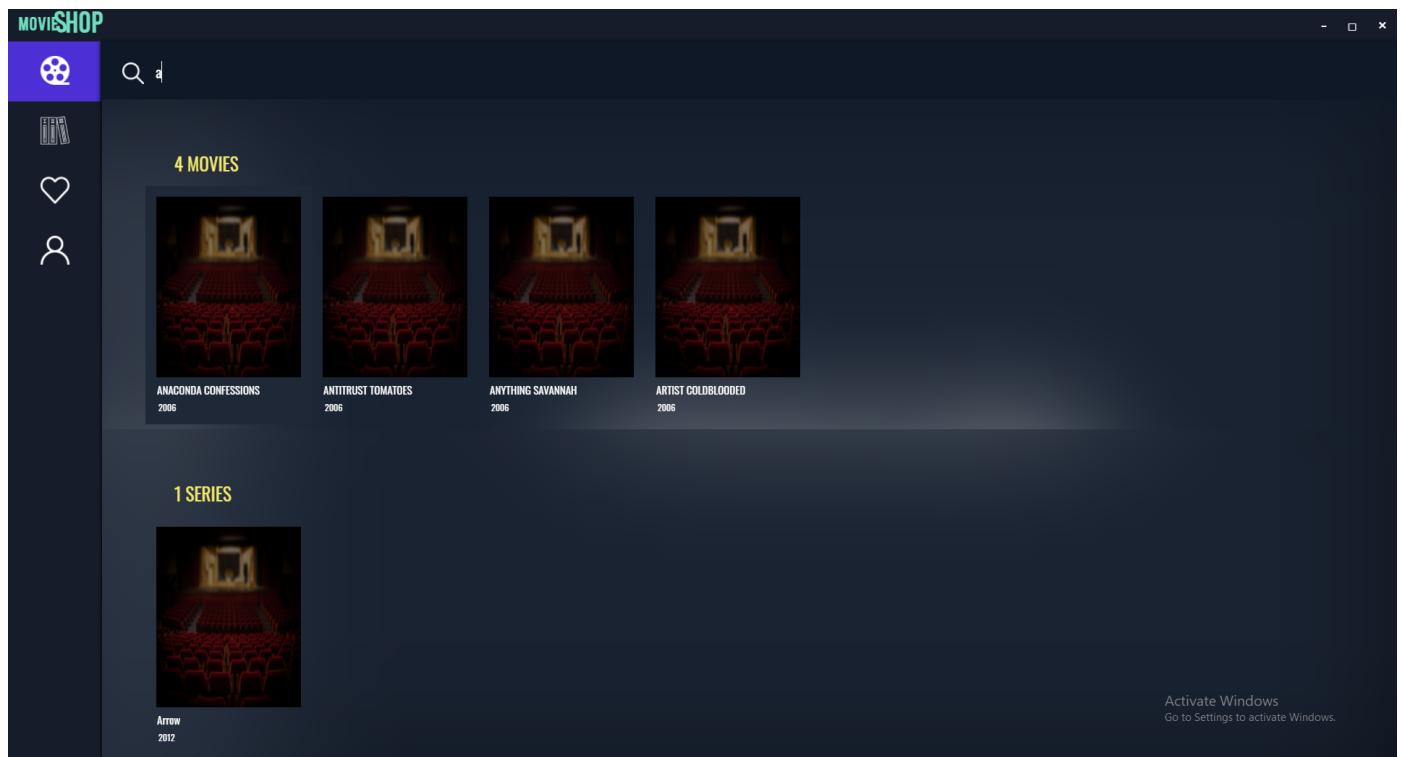
αλλιώς θα εμφανίζει



Επιπλέον ο χρήστης έχει την επιλογή από το **left bar**



να επιλέξει ανάμεσα σε ταινίες, σειρές, να δεί το υλικό που έχει στην κατοχή του και τα στοιχεία του λογαριασμού του. Δίνεται επίσης η δυνατότητα να αναζητήσει σειρές ή ταινίες πληκτρολογώντας στο **search bar**.



Όλα αυτά έχουν επιτευχθεί με την χρήση sockets στα οποία κάνουμε emit στον server ένα μήνυμα και αυτος μας το επιστρέφει.

Παράδειγμα για το πώς παίρνουμε τις ταινίες:

(θα δώσουμε ένα παράδειγμα γιατί ο κώδικας αποτελείται πάνω από 5000 γραμμές και δεν μπορούμε να τα αναλύσουμε ολα, αλλα ακολουθούν το ίδιο ύφος)

```

films.addEventListener("click", ()=>{
    films.style.backgroundColor = "#4d30d5";
    if(divClicked != films) divClicked.style.backgroundColor = "";
    document.querySelector('.search-content').style.display = "none";
    document.querySelector('.accountDiv').style.display = "none";
    search.value = '';
    category.innerHTML = "Movies";
    divClicked = films;
    document.querySelector('.content').innerHTML = "";

    if(customer_choice == 'FS' || customer_choice == 'F'){
        socket.emit('getFilms','');
    }

    socket.on('takeFilms',movies=>{
        createMovie(movies,'.content');
    });
}

```

```

        }
        else notRegistered();
    });
}

```

Έχοντας πάρει το **choice** από τον πίνακα **customer** από το login, χρησιμοποιούμε **if/else** για να ελέγξουμε αν ανήκει σε “F”(μόνο ταινίες) ή “FS”(και σειρές και ταινίες). Αν ισχύει ένα από τα δύο στέλνουμε μήνυμα στο **server** με την γραμμή

```
socket.emit('getFilms','');
```

Περνώντας στο αρχείο **server.js** διαχειριζόμαστε αυτό το μήνυμα ως εξής:

```

socket.on("getFilms",()=>{
    getFilms("film").then(results=>{
        socket.emit("getFilms",results);

    });
});
}

```

Με την εντολή **socket.on** ‘ακούμε’ το μύνημα “**getFilms**” και καλούμε την μέθοδο **getFilms** η οποία:

```

function getFilms(tableName){

    return new Promise((resolve, reject)=>{

        let query = "SELECT film.film_id,title, description, release_year,language.name as language , length, rating, category.name as category FROM film inner join language on film.language_id = language.language_id inner join film_category on film_category.film_id = film.film_id inner join category on category.category_id = film_category.category_id ORDER BY title asc;"

        connection.query(query,(err, results) =>{
            if(err) throw err;

            else{

                resolve(JSON.parse(JSON.stringify(results)));
            }
        })
    });

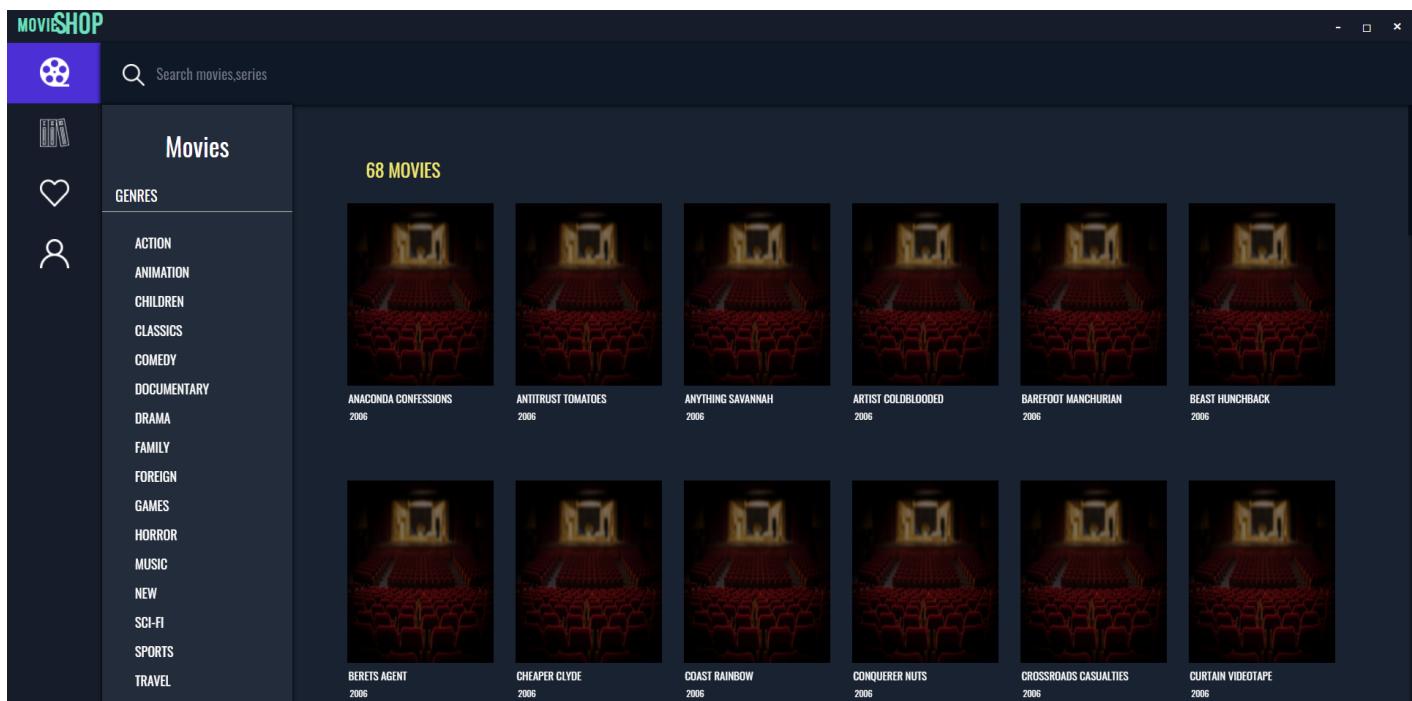
}

```

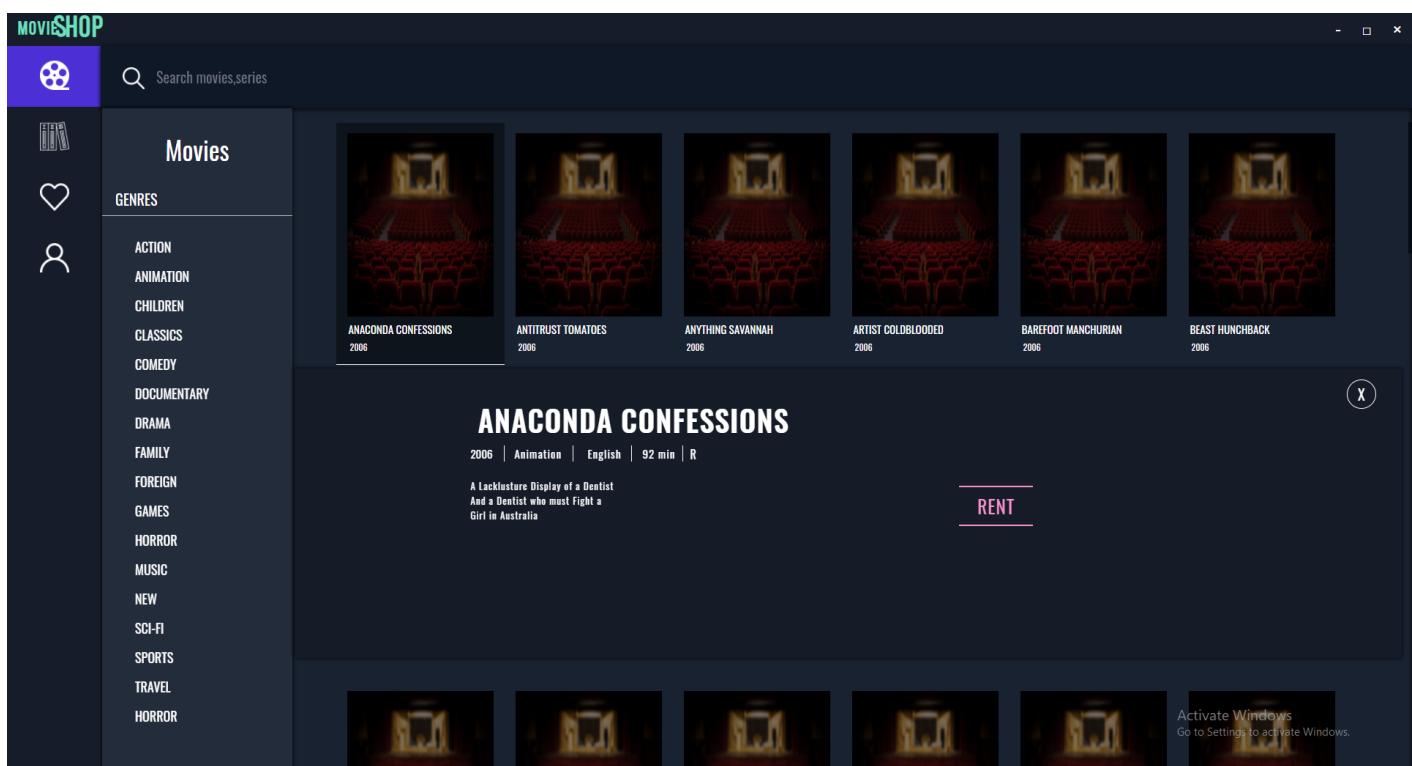
επιστρέφει μια υπόσχεση (θέλουμε να περιμένουμε μέχρι να ολοκληρωθεί η function μερος του ασύγχρονου προγραμματισμού(async functions)) στην οποία

έχουν γίνει **resolve** τα αποτελέσματα στα οποία έχουμε τρέξει ένα query select όπως φαίνεται και παραπάνω χρησιμοποιώντας το object connection που είναι αυτό που συνδέει το **server** με την βάση δεδομένων. Έπειτα παίρνουμε τα αποτελέσματα αυτά και με **socket.emit** τα στέλνουμε πίσω στον client. O client καλεί την function **createMovie()** η οποία δημιουργεί τα γραφικά που βλέπουμε.

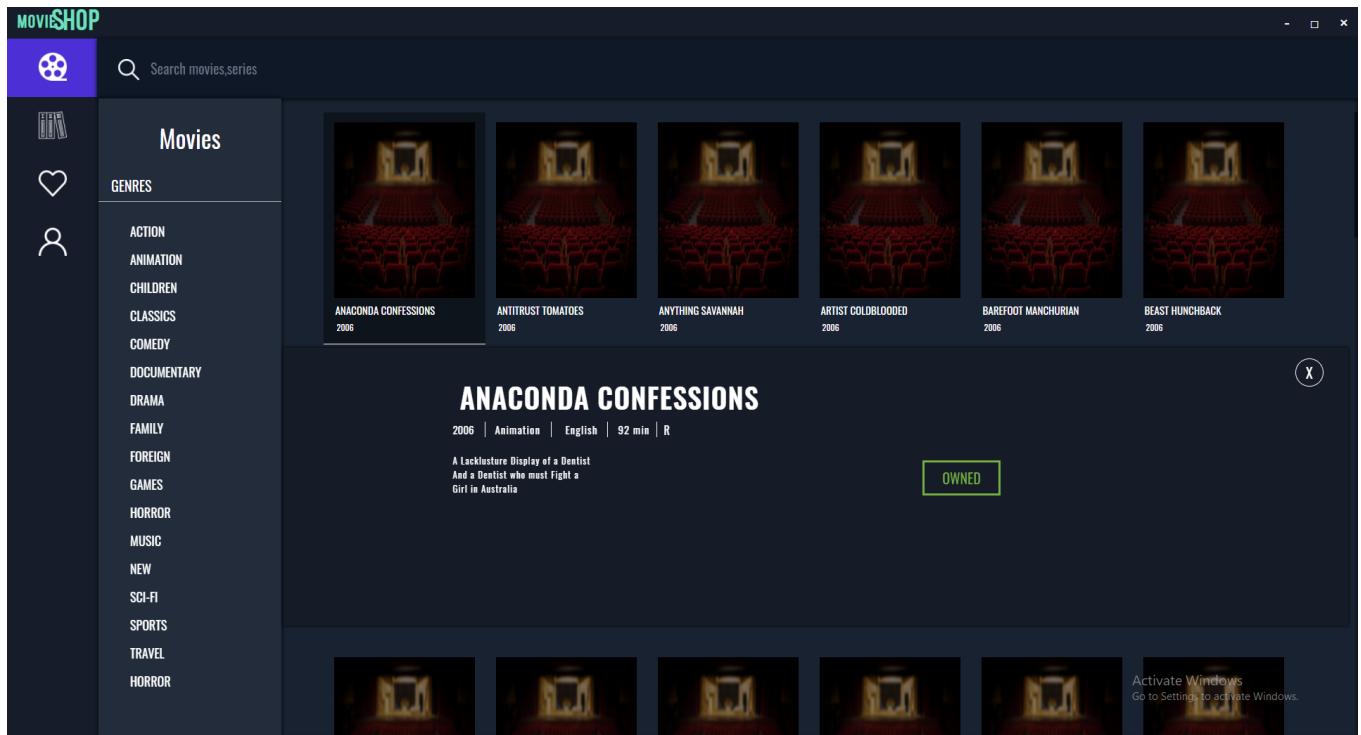
Ταινίες:



Πατώντας σε κάθε ταινία μπορούμε να την κάνουμε **rent**



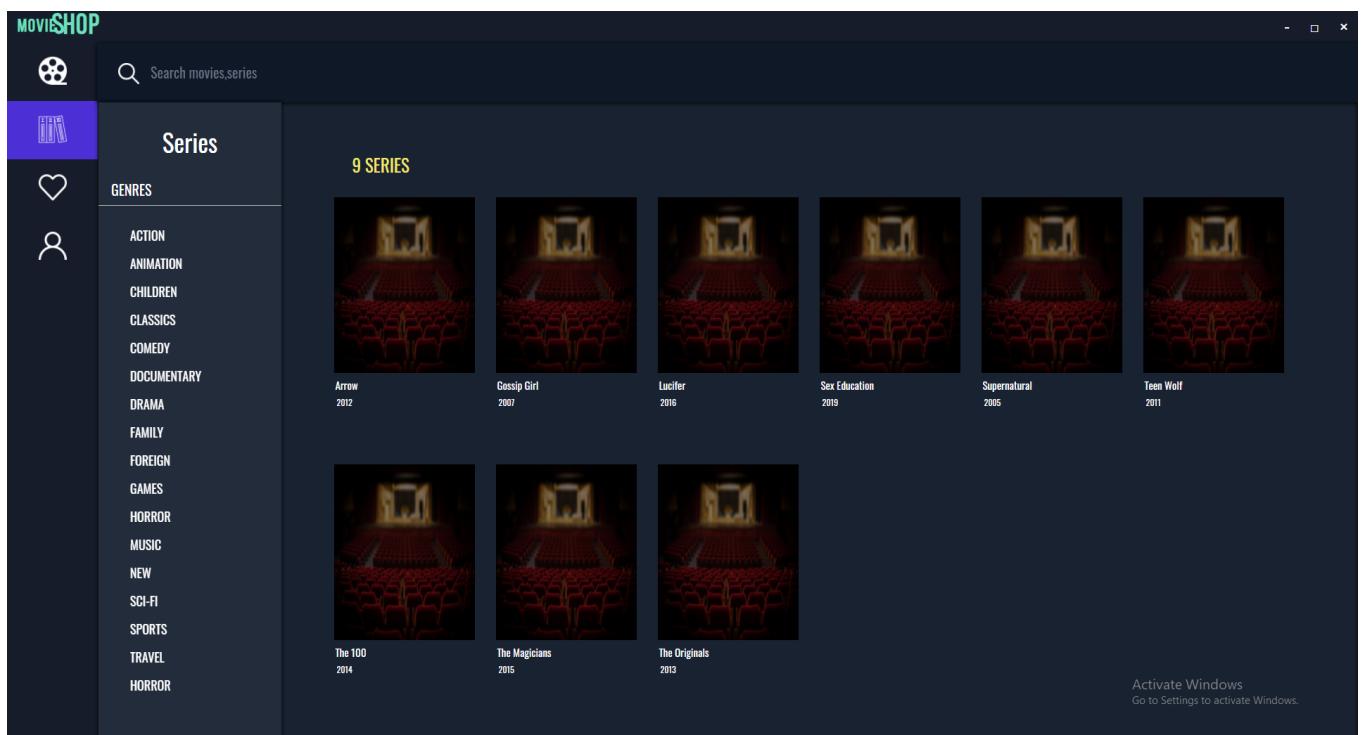
ή να μας βγάλει ένδειξη ότι τήν έχουμε ήδη κάνει rent(owned)



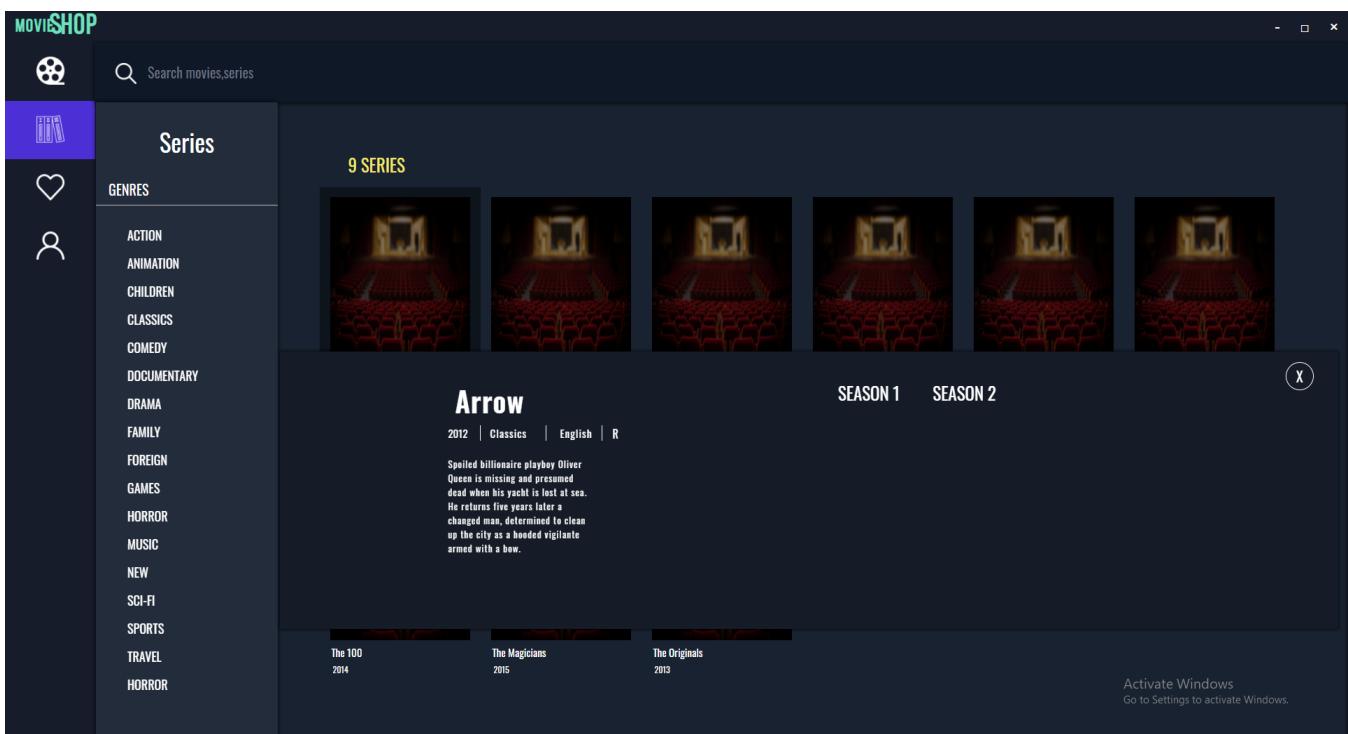
Ενώ ταυτόχρονα παρουσιάζονται και πληροφορίες για την ταινία όπως η περιγραφή, ο τίτλος, το έτος κυκλοφορίας, κτλ.

Σειρές:

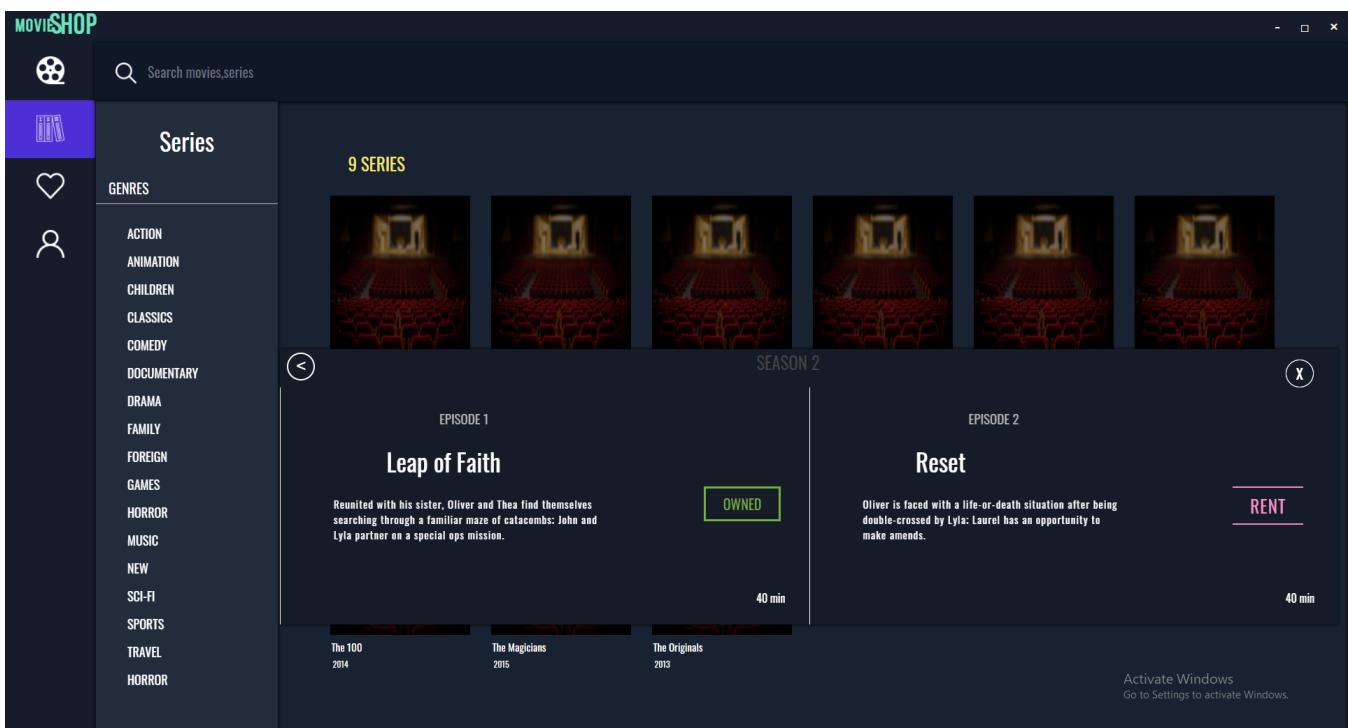
Πατώντας το δεύτερο κουμπί από το **left bar** έχουμε πρόσβαση στις σειρές αν ο χρήστης είναι εγγεγραμμένος για αυτό το υλικό.



Πατώντας σε κάθε σειρά εμφανίζεται:

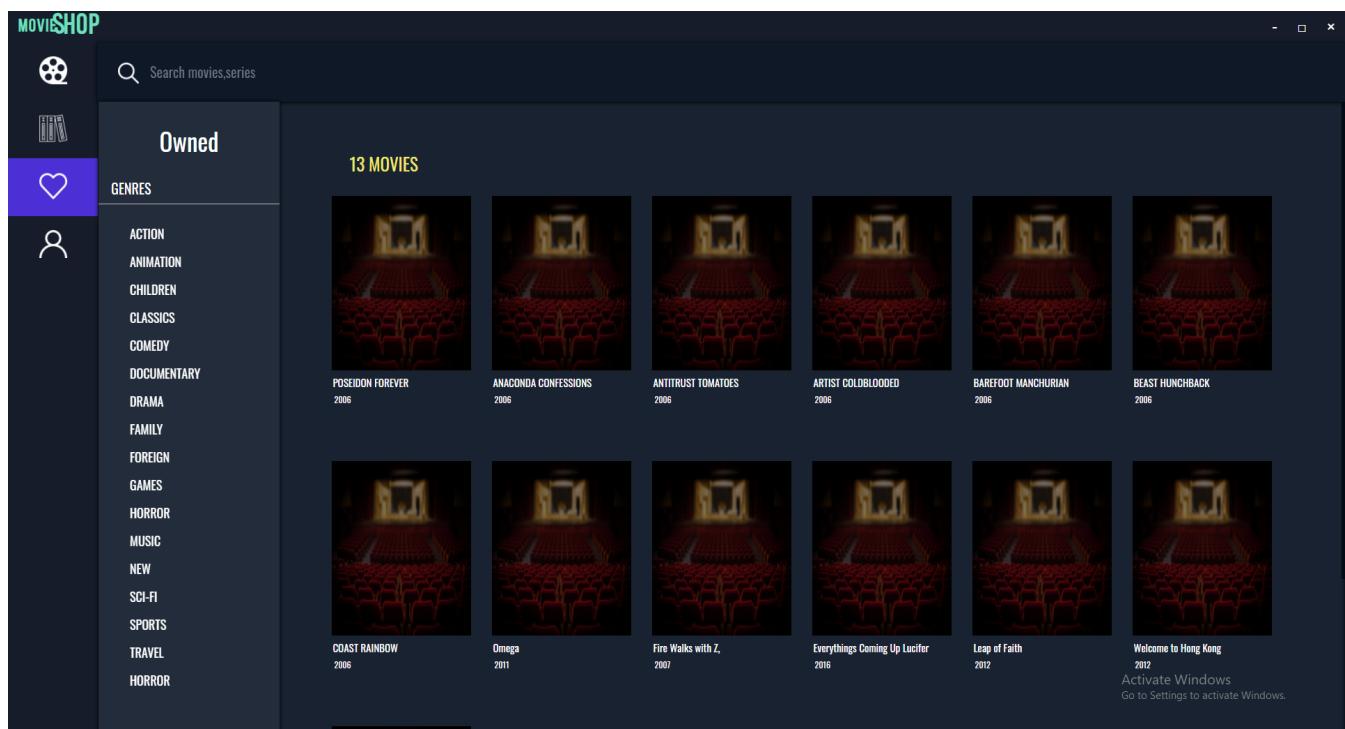


οι πληροφορίες της σειράς και επιλογή για κάθε **season**, ενώ αν επιλέξουμε σεζόν εμφανίζονται τα επεισόδια τα οποία όπως και οι ταινίες μας εμφανίζουν αν θέλουμε να τα νοικίασουμε ή αν τα έχουμε ήδη ενώ ταυτόχρονα εμφανίζονται και πληροφορίες για αυτά.

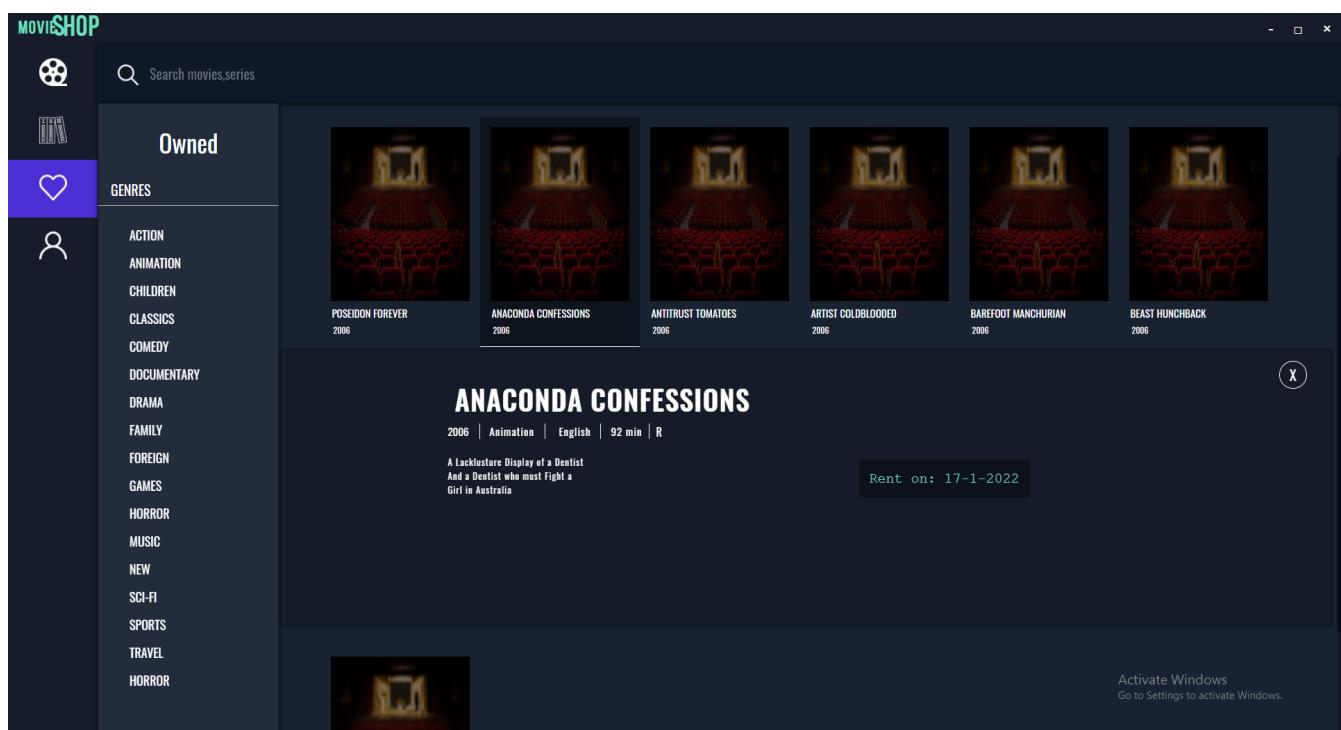


Owned:

Για να έχουμε πρόσβαση στο περιεχόμενο ενοικιάσεων αρκεί να πατήσουμε το τρίτο κουμπί του **left bar** τότε θα δούμε τα επεισόδια και τις ταινίες που έχουμε νοικιάσει.

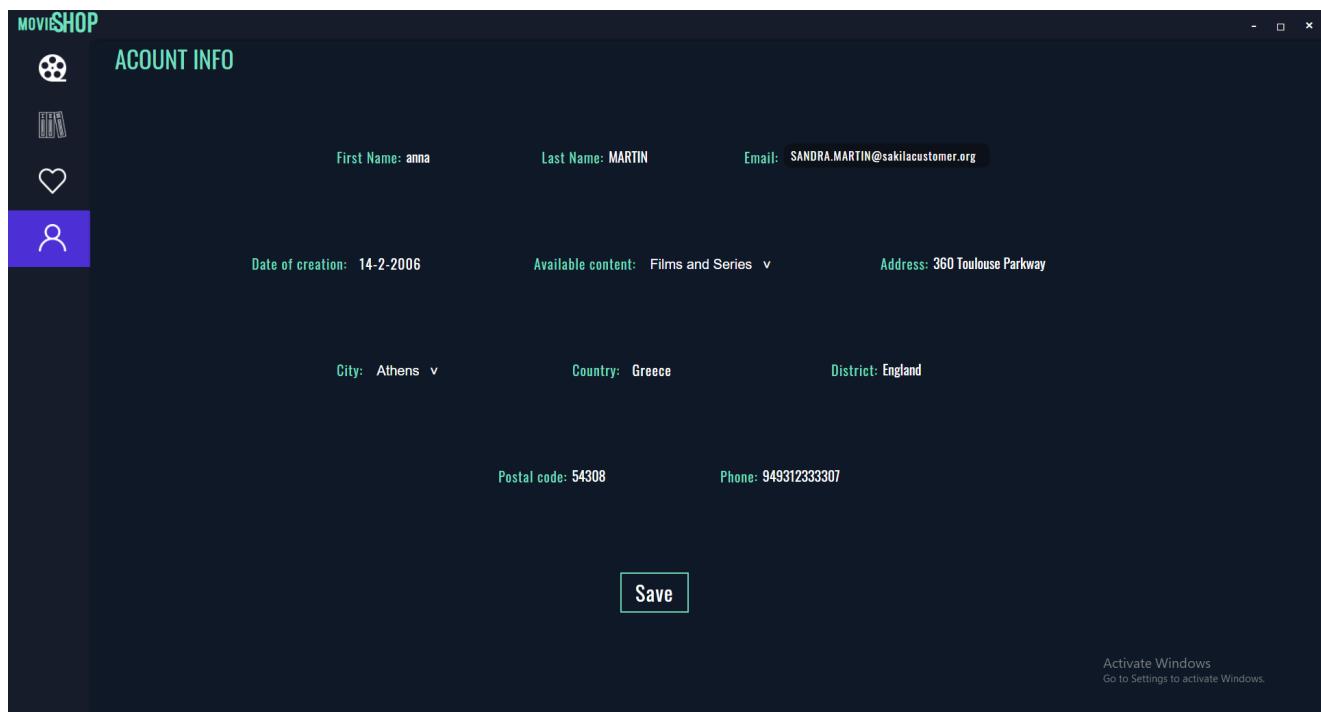


Πατώντας σε κάθε ταινία/επεισόδιο μας εμφανίζεται η ημερομηνία που νοικιάσαμε την ταινία.(Γενικά έχουμε κάνει την παραδοχή ότι όταν ενοικιάζεται μια ταινία ο χρήστης την έχει για πάντα).



Account:

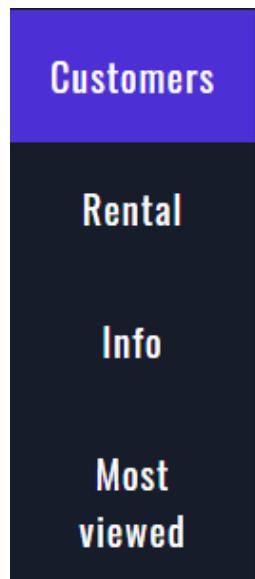
Για να έχουμε πρόσβαση στα στοιχεία του λογαριασμού μας επιλέγουμε το τέταρτο(και τελευταίο) κουμπί του **left bar** και εμφανίζεται:



Μπορούμε να επεξεργαστούμε όλα τα στοιχεία εκτός από το **email**(παραπάνω έχουμε αλλάξει το όνομα και την πόλη) και πατώντας **save** στέλνεται στο **server socket** και κάνει **update** τους πίνακες **address** και **customer** με τις πληροφορίες που στέλνονται.

Employee:

Στην σύνδεση ενός **employee** εμφανίζεται ένα **left bar** με επιλογές:



Customers για να μπορεί να βλέπει τους υπάρχοντες πελάτες και να επεξεργάζεται τα στοιχεία τους, **Rental** για να μπορεί να βλέπει ποιές ταινίες έχει νοικιάσει ο κάθε πελάτης, **Info** για να μπορεί να επεξεργάζεται πληροφορίες όπως ηθοποιοί, ταινίες, σειρές, διευθύνσεις κτλ και **Most Viewed** για να μπορεί να δει τις πέντε πρώτες ταινίες και σειρές που έχουν ενοικιαστεί τον τελευταίο μήνα.

Customers:

MOVIESHOP

Customers

Rental

Info

Most viewed

13 CUSTOMERS

C BROTHERS CHRIS.BROTHERS@sakilacustomer.org	chris erdas CECIL.VINES@sakilacustomer.org	DAN PAINÉ DAN.PAINE@sakilacustomer.org	DARREN WINDHAM DARREN.WINDHAM@sakilacustomer.org	GILBERT SLEDGE GILBERT.SLEDGE@sakilacustomer.org	LAUREN HUDSON LAUREN.HUDSON@sakilacustomer.org	LINDA WILLIAMS LINDA.WILLIAMS@sakilacustomer.org
MAE FLETCHER MAE.FLETCHER@sakilacustomer.org	MARSHALL THORN MARSHALL.THORN@sakilacustomer.org	MATTIE HOFFMAN MATTIE.HOFFMAN@sakilacustomer.org	NELSON CHRISTENSON NELSON.CHRISTENSON@sakilacustomer.org	SANDRA MARTIN SANDRA.MARTIN@sakilacustomer.org	VIRGINIA GREEN VIRGINIA.GREEN@sakilacustomer.org	

Activate Windows
Go to Settings to activate Windows.

Δείχνει όλους τους πελάτες της βάσης και επιλέγοντας κάποιον εμφανίζει τις πληροφορίες τους τις οποίες μπορεί να επεξεργαστεί(όλες εκτός από το email):

MOVIESHOP

Customer INFO

Customers

Rental

Info

Most viewed

First Name: DAN **Last Name:** PAINE **Email:** DAN.PAINE@sakilacustomer.org

Date of creation: 14-2-2006 **Available content:** Films and Series **Address:** 808 Naala-Porto

City: Stockport **Country:** United Kingdom **District:** England

Postal code: 41060 **Phone:** 553452430707

Save

Activate Windows
Go to Settings to activate Windows.

Rental:

Εμφανίζονται πάλι οι πελάτες:

MOVIE SHOP

Customers

Rental

Info

Most viewed

13 CUSTOMERS

C BROTHERS
CHRIS.BROTHERS@zakiacustomer.org

chris erdas
CECILVINES@zakiacustomer.org

DAN PAINÉ
DANPAINÉ@zakiacustomer.org

DARREN WINDHAM
DARREN.WINDHAM@zakiacustomer.org

GILBERT SLEDGE
GILBERT.SLEDGE@zakiacustomer.org

LAUREN HUDSON
LAUREN.HUDSON@zakiacustomer.org

LINDA WILLIAMS
LINDA.WILLIAMS@zakiacustomer.org

MAE FLETCHER
MAE.FLETCHER@zakiacustomer.org

MARSHALL THORN
MARSHALL.THORN@zakiacustomer.org

MATTIE HOFFMAN
MATTIE.HOFFMAN@zakiacustomer.org

NELSON CHRISTENSON
NELSON.CHRISTENSON@zakiacustomer.org

SANDRA MARTIN
SANDRA.MARTIN@zakiacustomer.org

VIRGINIA GREEN
VIRGINIA.GREEN@zakiacustomer.org

Activate Windows
Go to Settings to activate Windows.

Αυτή την φορά όμως όταν πατήσουμε πάνω σ' έναν του εμφανίζει τις ταινίες που έχει νοικιάσει από τις οποίες μπορεί να δει την ημερομηνία που έγινε η ενοικίαση.

MOVIE SHOP

Customers

Rental

Info

Most viewed

6 MOVIES

Rental of customer: GILBERT SLEDGE

X

SUGAR WONKA
2006

DECEIVER BETRAYED
2006

WATERFRONT DELIVERANCE
2006

PUNK DIVORCE
2006

COAST RAINBOW
2006

GREEDY ROOTS
2006

Activate Windows
Go to Settings to activate Windows.

MOVIESHOP

Customers Rental of customer: GILBERT SLEDGE

Rental

Info

Most viewed

6 MOVIES

SUGAR WONKA
2006

DECEIVER BETRAYED
2006

WATERFRONT DELIVERANCE
2006

PUNK DIVORCE
2006

COAST RAINBOW
2006

GREEDY ROOTS
2006

SUGAR WONKA

2006 | Animation | English | 114 min | PG

A Touching Story of a Dentist And a Database Administrator who must Conquer a Astronaut in An Abandoned Amusement Park

Rent on: 19-6-2005

Activate Windows
Go to Settings to activate Windows.

Info:

Πατώντας το κουμπί **Info** εμφανίζεται μια επιλογή για τις κατηγορίες που ο χρήστης μπορεί να ενημερώσει(σε άλλες να διαγράψει, αλλες να διαγράψει ή να τροποποιήσει).

MOVIESHOP

Customers

Rental

Info

ACTORS MOVIES SERIES LANGUAGES CATEGORIES ADDRESSES CITIES COUNTRIES

Most viewed

Activate Windows
Go to Settings to activate Windows.

Επιλέγοντας την κάθε μια μπορεί να εκτελέσει τις παραπάνω πράξεις.

Πχ στο actor:

MOVIE SHOP

Customers Rental Info Most viewed

ACTORS MOVIES SERIES LANGUAGES CATEGORIES ADDRESSES CITIES COUNTRIES

Add new Actor Delete Actor

First name: _____

Last name: _____

ADD

First name: _____

Last name: _____

DELETE

Activate Windows
Go to Settings to activate Windows.

Στα movies:

MOVIE SHOP

Customers Rental Info Most viewed

ACTORS MOVIES SERIES LANGUAGES CATEGORIES ADDRESSES CITIES COUNTRIES

68 MOVIES ADD FILM

ANACONDA CONFESSIONS ANITRUST TOMATOES ANYTHING SAVANNAH ARTIST COLD BLOODED BAREFOOT MANCHURIAN BEAST HUNCHBACK BERETS AGENT

2006 2006 2006 2006 2006 2006 2006

CHEAPER CLYDE COAST RAINBOW CONQUERER NUTS CROSSROADS CASUALTIES CURTAIN VIDEOTAPE DADDY PITTSBURGH DECEIVER BETRAYED

2006 2006 2006 2006 2006 2006 2006

Activate Windows
Go to Settings to activate Windows.

Movieshop

- Customers
- Rental
- Info**
- ACTORS
- MOVIES**
- SERIES
- LANGUAGES
- CATEGORIES
- ADDRESSES
- CITIES
- COUNTRIES

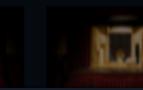
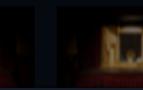
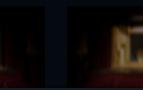
Most viewed

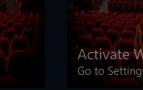
68 MOVIES [ADD FILM](#)

ADD FILM X

TITLE: _____ RELEASE YEAR: _____ LENGTH: _____ [SAVE](#)

DESCRIPTION: _____ CATEGORY: [Choose category](#) LANGUAGE: [Choose language](#) RATING: [Choose rating](#)

CHEAPER CLYDE COAST RAINBOW CONQUERER NUTS CROSSROADS CASUALTIES CURTAIN VIDEOTAPE DADDY PITTSBURGH DECEIVER BETRAYED

Activate Windows
Go to Settings to activate Windows.

Στα addresses:

Movieshop

- Customers
- Rental
- Info**
- ACTORS
- MOVIES
- SERIES
- LANGUAGES
- CATEGORIES
- ADDRESSES**
- CITIES
- COUNTRIES

Most viewed

ADD ADDRESS

ADDRESS	DISTRICT	CITY	POSTAL CODE	PHONE	
1497 Yuzhou Drive	England	London	3433	246810237916	delete
1515 Korla Way	England	York	57197	959467760895	delete
1557 Ktahya Boulevard	England	Bradford	88002	720998247660	delete
1584 Ljubertsy Lane	England	Southampton	22954	285710089439	delete
1740 Le Mans Loop	Pays de la Loire	Le Mans	22853	168476538960	delete
1764 Jalib al-Shuyukh Parkway	Galicia	Santiago de Compostela	77642	84794532510	delete
1926 Gingog Street	Sisilia	Syrakusa	22824	469738825391	delete
1986 Sivas Place	Friuli-Venezia Giulia	Udine	95775	182059202712	delete
231 Kaliningrad Place	Lombardia	Bergamo	57833	575081026569	Activate Windows Go to Settings to activate Windows. delete

Movieshop

- Customers
- Rental
- Info**

Most viewed

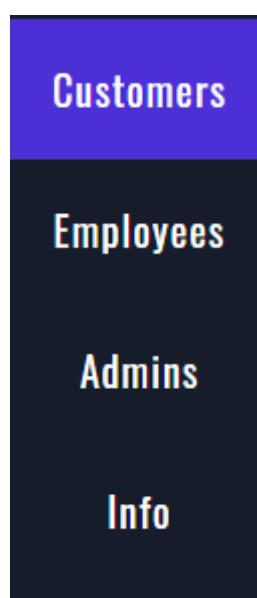
ACTORS	MOVIES	SERIES	LANGUAGES	CATEGORIES	ADDRESSES	CITIES	COUNTRIES
ADD ADDRESS							
ADDRESS:	DISTRICT:	CITY: <input type="text" value="Choose city"/>		<input type="button" value="X"/>			
POSTAL CODE:	PHONE:	<input type="button" value="Save"/>					
1584 Ljubertsy Lane	England	Southampton	22954	285710089439	<input type="button" value="delete"/>		
1740 Le Mans Loop	Pays de la Loire	Le Mans	22853	168476538960	<input type="button" value="delete"/>		
1764 Jaiib al-Shuyukh Parkway	Galicia	Santiago de Compostela	77642	84794532510	<input type="button" value="delete"/>		
1926 Gingog Street	Sisilia	Syrakusa	22824	469738825391	<input type="button" value="delete"/>		
1986 Sivas Place	Friuli-Venezia Giulia	Udine	95775	182059202712	<input type="button" value="delete"/>		
231 Kaliningrad Place	Lombardia	Bergamo	57833	575081026569	<input type="button" value="Activate Windows"/> Activate Windows Go to Settings to activate Windows <input type="button" value="delete"/>		

Most Viewed:

Επιλέγοντας το κουμπί **Most viewed** εμφανίζονται οι πέντε ταινίες και τα πέντε επεισόδια που έχουν ενοικιαστεί περισσότερες φορές τον τελευταίο μήνα (στο **server** καλείται η **procedure 3.2**).

Administrator:

Στην σύνδεση ενός **Administrator** εμφανίζεται ένα **left bar** με επιλογές:



Customers για να μπορεί να βλέπει τους υπάρχοντες πελάτες και να τους διαγράφει ή να προσθέτει νέο πελάτη, **Employee** για να μπορεί να βλέπει τους υπάρχοντες υπαλλήλους να μπορεί να τους διαγράφει να πρόσθετει νέο ή να αλλάζει κάποιον σε **administrator**, **Admins** για να μπορεί να βλέπει τους υπάρχοντες **administrators** και να μπορεί να τους αλλάξει σε υπαλλήλους, **Info** για να μπορεί να βλέπει τα έσοδα ανα μήνα και να αλλάζει τις τιμές των ταινιών.

Customers:

Όταν πατηθεί αυτό το κουμπί εμφανίζονται οι πελάτες και επιλογές για διαγραφη(αν πατήσει πάνω σε πελάτη) ή προσθήκη νέου πελάτη αν πατήσει το κουμπί **Create Customer**.

CREATE CUSTOMER

C BROTHERS CHRIS.BROTHERS@sakilacustomer.org	chris erdas CECIL.VINES@sakilacustomer.org	DAN PAINES DAN.PAINES@sakilacustomer.org	DARREN WINDHAM DARREN.WINDHAM@sakilacustomer.org	GILBERT SLEDGE GILBERT.SLEDGE@sakilacustomer.org	LAUREN HUDSON LAUREN.HUDSON@sakilacustomer.org	LINDA WILLIAMS LINDA.WILLIAMS@sakilacustomer.org
MAE FLETCHER MAE.FLETCHER@sakilacustomer.org	MARSHALL THORN MARSHALL.THORN@sakilacustomer.org	MATTIE HOFFMAN MATTIE.HOFFMAN@sakilacustomer.org	NELSON CHRISTENSON NELSON.CHRISTENSON@sakilacustomer.org	SANDRA MARTIN SANDRA.MARTIN@sakilacustomer.org	VIRGINIA GREEN VIRGINIA.GREEN@sakilacustomer.org	

Activate Windows
Go to Settings to activate Windows.

MOVIE SHOP

Customers

CREATE CUSTOMER

DELETE

C BROTHERS CHRIS.BROTHERS@akilacustomer.org	chris erdas CECILVINES@akilacustomer.org	DAN PAINÉ DAN.PAINE@akilacustomer.org	DARREN WINDHAM DARREN.WINDHAM@akilacustomer.org	GILBERT SLEDGE GILBERT.SLEDGE@akilacustomer.org	LAUREN HUDSON LAUREN.HUDSON@akilacustomer.org	LINDA WILLIAMS LINDA.WILLIAMS@akilacustomer.org
MAE FLETCHER MAE.FLETCHER@akilacustomer.org	MARSHALL THORN MARSHALL.THORN@akilacustomer.org	MATTIE HOFFMAN MATTIE.HOFFMAN@akilacustomer.org	NELSON CHRISTENSON NELSON.CHRISTENSON@akilacustomer.org	SANDRA MARTIN SANDRA.MARTIN@akilacustomer.org	VIRGINIA GREEN VIRGINIA.GREEN@akilacustomer.org	

Activate Windows
Go to Settings to activate Windows.

MOVIE SHOP

Customer INFO

Customers

Employees

Admins

Info

X

First Name: [Input field]

Last Name: [Input field]

Email: [Input field]

Date of creation: 13-2-2022

Available content: Films and Series

Address: [Input field]

City: [Input field]

Country: [Input field]

District: [Input field]

Postal code: [Input field]

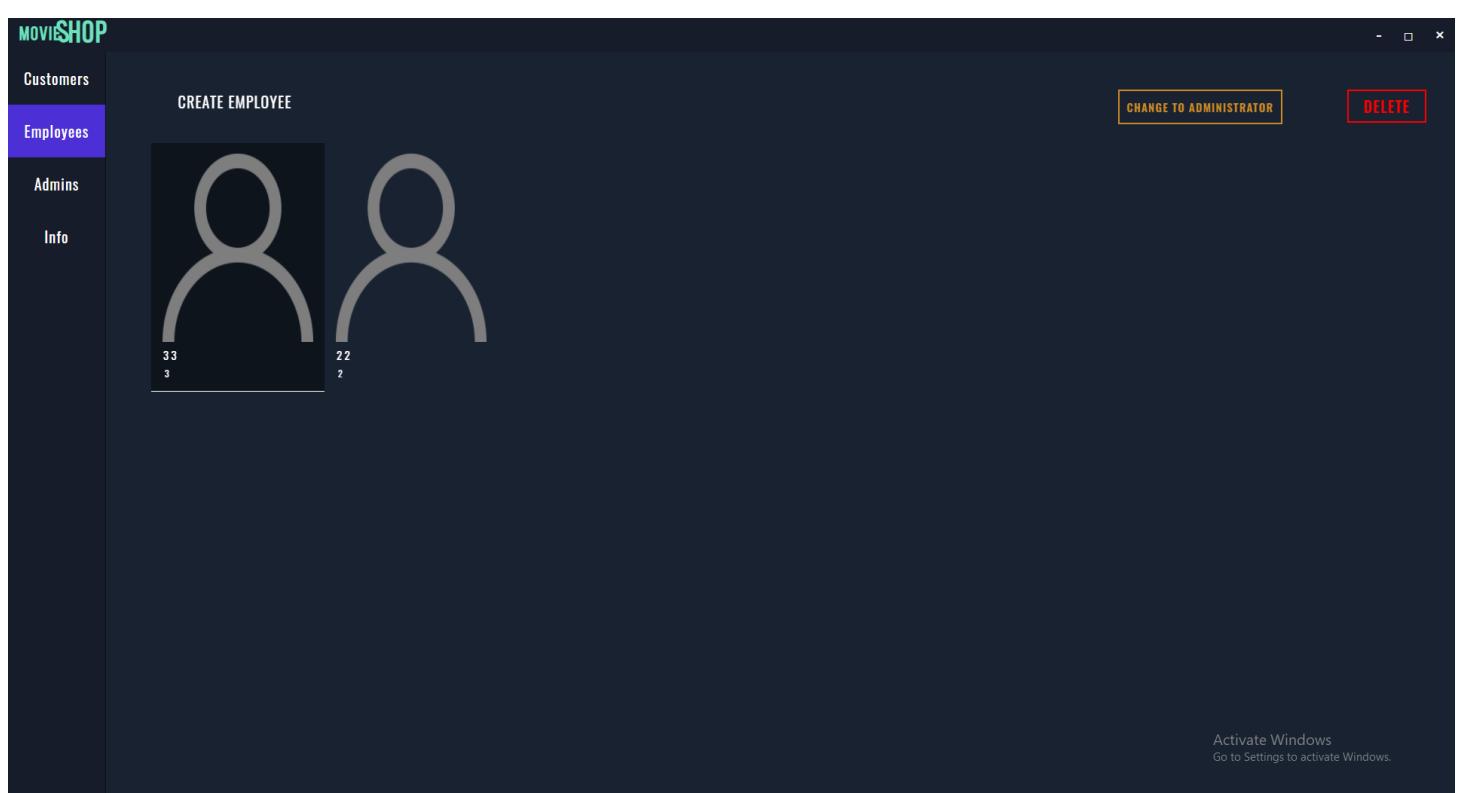
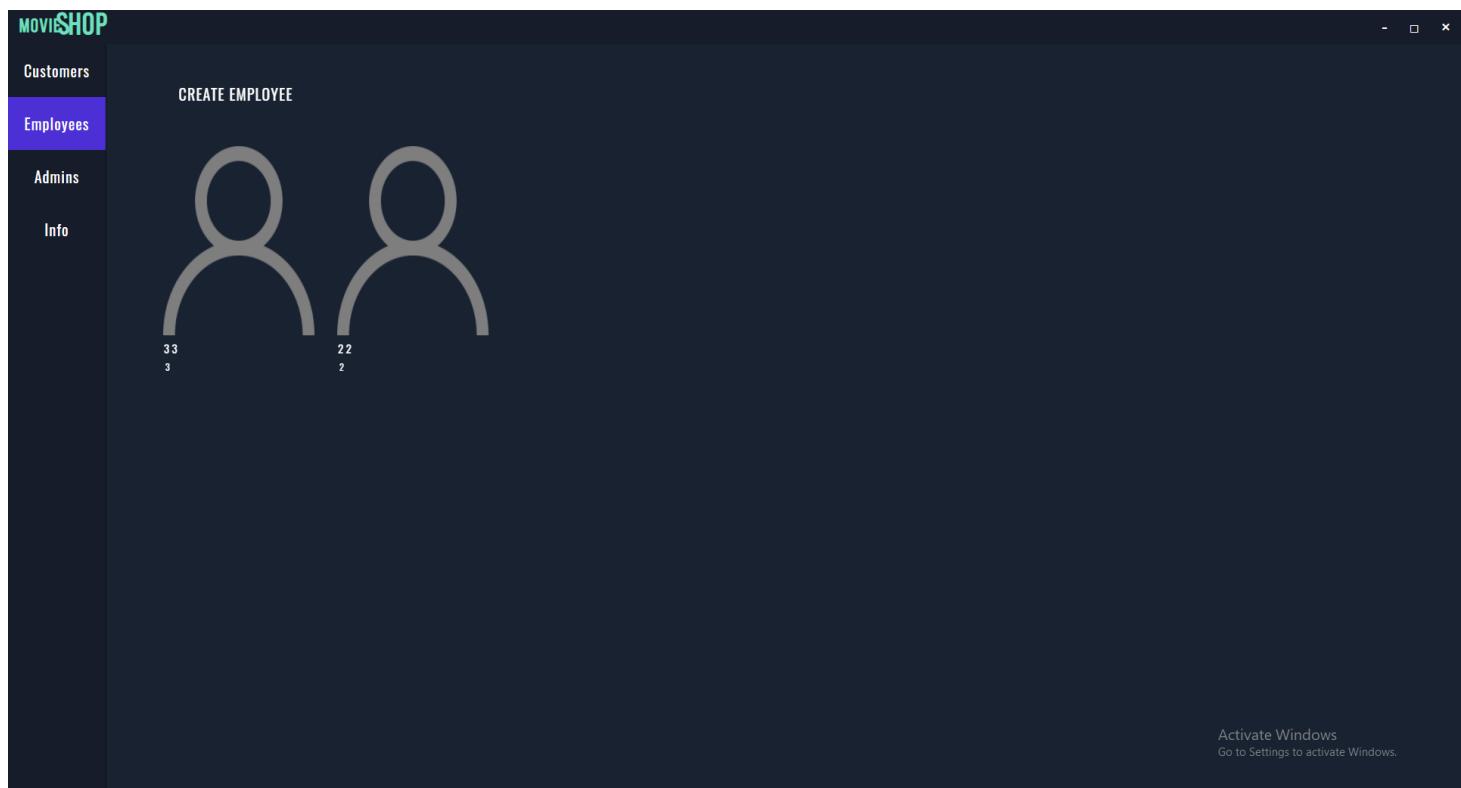
Phone: [Input field]

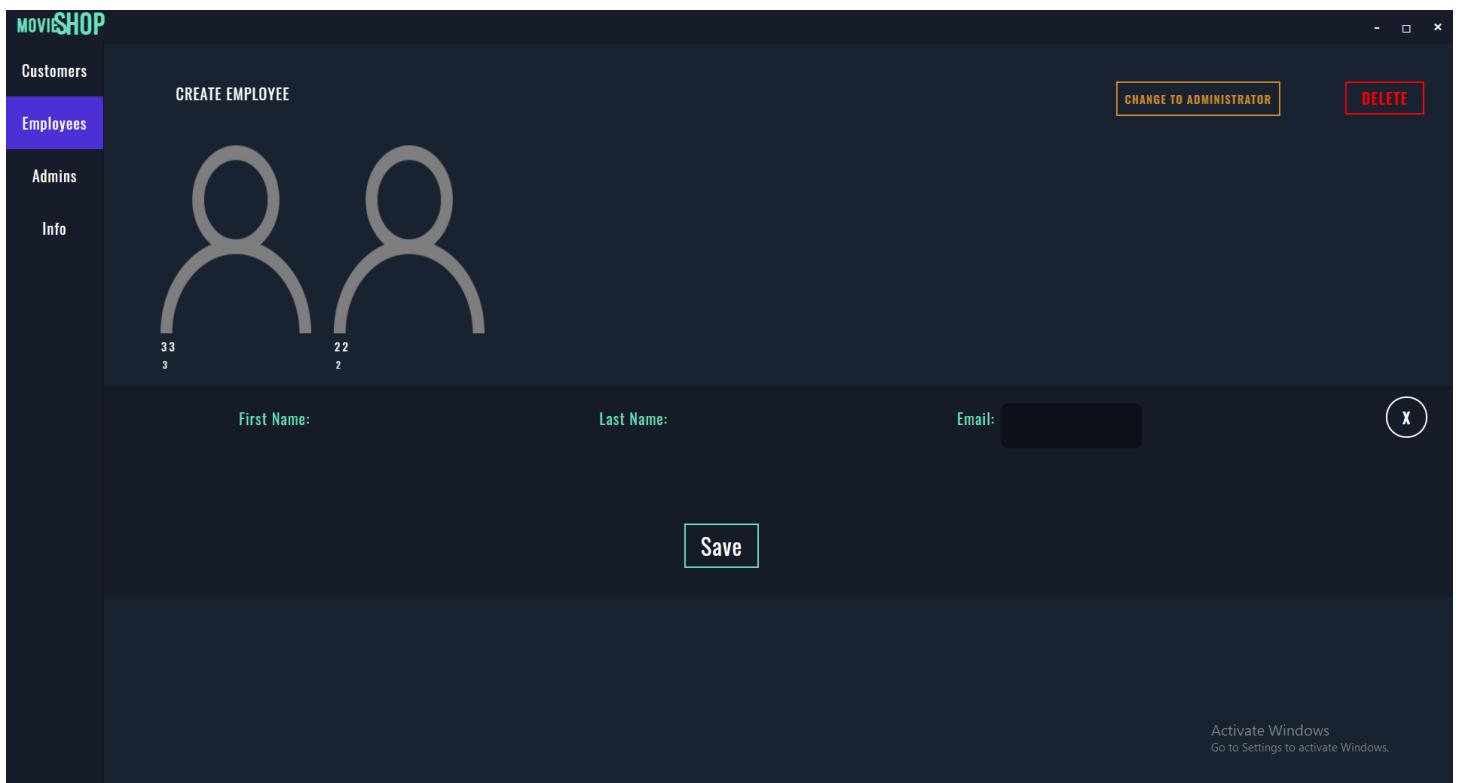
Save

Activate Windows
Go to Settings to activate Windows.

Employees:

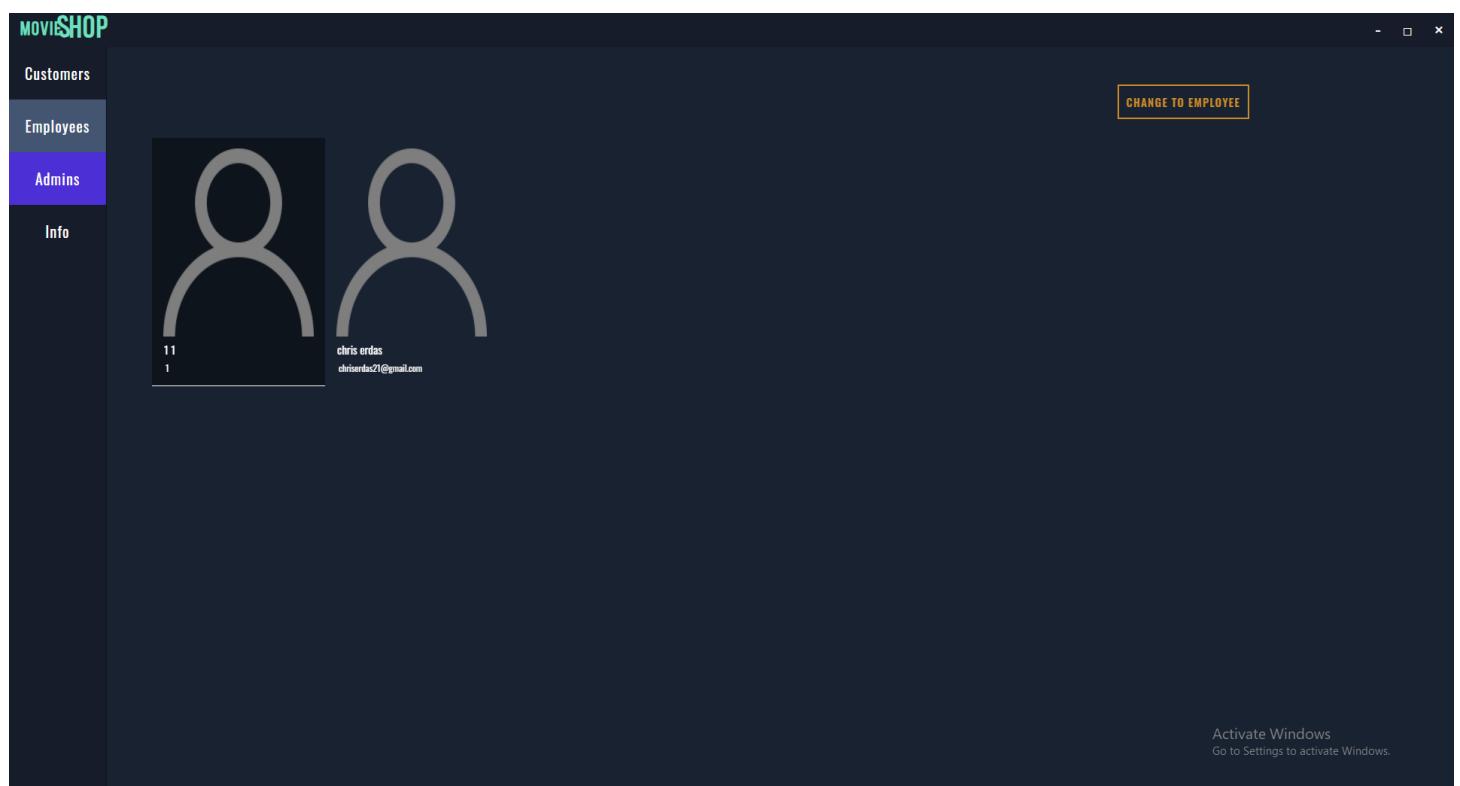
Πατώντας την επιλογή **Employees** εμφανίζονται οι υπάρχοντες υπάλληλοι και δίνονται οι αντίστοιχες επιλογές με τον **customer** με μία επιπλέον να μπορεί να αναβαθμιστεί ως διαχειριστής(πειραματικές τιμές διαθέτει το **όνομα** το **επίθετο** και το **email**).





Admins:

Ο χρήστης έχει τη δυνατότητα να βλέπει τους υπάρχοντες διαχειριστές και να τους αναθέσει σε υπαλλήλους.



Info:

Στην τελευταία επιλογή του admin μπορεί να βλέπει τα έσοδα χρησιμοποιώντας την **procedure 3.3** και να αλλάζει τις τιμές των ταινιών ανάλογα βέβαια με το είδος της συνδρομής.

The screenshot shows the MovieShop application interface. On the left, there is a vertical navigation bar with buttons for Customers, Employees, Admins, and Info. The Info button is currently selected and highlighted in purple. In the center, there is a section titled "CHANGE RENT PRICES". Below it, a sub-section titled "Films and Series" has two options: "Films" and "Series". To the right of these sections, there is a table showing monthly income data:

Month	Total Income
January	8.29
February	6.58
March	0
April	0
May	13.97
June	17.96
July	39.3
August	15.35
September	0
October	0
November	0
December	0

At the bottom right of the application window, there is a watermark that reads "Activate Windows Go to Settings to activate Windows."

This screenshot is similar to the previous one, showing the MovieShop application interface. The navigation bar, central sections, and monthly income table are identical. However, the "Films and Series" sub-section now has a dropdown menu open, labeled "Choose Type". The dropdown menu contains three options: "Films" and "Series", which are visible, and "Films and Series", which is the currently selected option and is bolded.

The screenshot shows a Windows application window titled "MovieSHOP". On the left is a vertical sidebar with menu items: "Customers", "Employees", "Admins", and "Info", where "Info" is highlighted in purple. At the top center are buttons for "CHANGE RENT PRICES", "Films and Series", "PRICE: movies: 0.3 Series: 0.1", and a "SAVE" button. The main area contains a table with 12 rows, each representing a month from January to December and its total income. The table has two columns: "Month" and "Total Income". The data is as follows:

Month	Total Income
January	8.29
February	6.58
March	0
April	0
May	13.97
June	17.96
July	39.3
August	15.35
September	0
October	0
November	0
December	0

Activate Windows
Go to Settings to activate Windows.

Πίνακας log:

Όπως ζητήθηκε και στο μέρος α', δημιουργήσαμε έναν πίνακα **log** όπου κρατάει την δραστηριότητα του κάθε χρήστη σε σχέση με το **username**(το email του).

```
create table log(
    username varchar(50),
    table_name VARCHAR(20) NOT NULL,
    action VARCHAR(30),
    action_date DATETIME,
    success ENUM('yes', 'no'),
    PRIMARY KEY(username),
    unique(username)
)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Στον server δημιουργήσαμε μία function η οποία με εντολή **insert into** προσθέτει στον πίνακα log τα νέα δεδομένα ενώ αν ο χρήστης υπάρχει χρησιμοποιεί την εντολή **on duplicate key update** για να ενημερώσει την συγκεκριμένη σειρά

```

function insertIntoLog(username,tableName,action,date,success) {

    let query = "Insert into
log(username,table_name,action,action_date,success) "+ 
    "values(?,?,?,?,?) ON DUPLICATE KEY UPDATE " +
    "username = ?, table_name = ?, action = ?, action_date = ?, success = ?"

    return new Promise((resolve, reject) => {

        connection.query(query,[username,tableName,action,date,success],(err, results)=>{
            if(err) throw err;
            resolve(results);
        });
    });
}

```

Έτσι από τον client στέλνετε το μήνυμα για την εκάστοτε κίνηση και το email το οποίο έχουμε πάρει κατα το authorization και προσθέτουμε αυτά ως παραμέτρους στην **insertIntoLog()**.

Έπιπλέον οσο αναφορά το τελευταίο ερώτημα έχουμε περιορίσει την εισαγωγή δεδομένων σε αρκετές περιπτώσεις όπως στα μήκη των **textareas** που παίρνουμε τα δεδομένα σε **dropdown menus** που έχουμε υλοποιήσει όπως στην εισαγωγή **πόλεων, χωρών, rating**(στο **add movies/add series**) και στον **employee** στο τμήμα **Info**.

Για οποιοδήποτε πρόβλημα επικοινωνήστε μαζί μας στο email: chriserdas21@gmail.com

Τελος