ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής

Εργασία Μαθήματος «Προγραμματισμός στο διαδίκτυο και στον παγκόσμιο ιστό»

«αριθμός άσκησης»	**ΤΕΛΙΚΗ ΑΣΚΗΣΗ ΜΑΘΗΜΑΤΟΣ**
Όνομα φοιτητή – Αρ. Μητρώου (όλων σε περίπτωση ομαδικής εργασίας)	ΝΙΚΟΛΑΣ ΠΑΤΕΡΑΣ – Π17172
	ΒΑΣΙΛΕΙΟΣ ΠΑΠΑΧΑΡΑΛΑΜΠΟΥΣ – Π17168
	ΑΝΔΡΕΑΣ ΘΕΟΔΩΡΙΔΗΣ – Π17164
Ημερομηνία Παράδοσης	29-06-2019

Εκφώνηση της άσκησης

Στόχοι εργασίας: Ολοκλήρωση λειτουργικότητας *3-tier εφαρμογής, ολοκλήρωση serverside τεχνολογιών (servlets και jsp), επικοινωνία με βάση δεδομένων, ολοκλήρωση λειτουργιών.*
Στην τελική εργασία του μαθήματος θα επεκτείνετε τις προηγούμενες ασκήσεις ώστε να δημιουργήσετε μία εφαρμογή τριών επιπέδων (3-tier), η οποία θα υλοποιεί τις λειτουργίες (μεθόδους) που ορίσατε στις προηγούμενες ασκήσεις.

Αναλυτικά Βήματα:

1. **Επέκταση web project προηγούμενης άσκησης**
   - **1.1.** Στην τελική εργασία θα επεκτείνετε τη λειτουργικότητα του web project που δημιουργήσατε στην προηγούμενη άσκηση και θα υλοποιήσετε όλη την ζητούμενη λειτουργικότητα για κάθε κατηγορία χρηστών.
2. **Ολοκλήρωση λειτουργιών όλων των κατηγοριών χρηστών (servlet)**
   - **2.1.** Λειτουργίες που αφορούν όλες τις κατηγορίες χρηστών:
     - **2.1.1.** Σύνδεση (login): Κάθε χρήστης θα πρέπει να συνδέεται με το μοναδικό username-password. Το password θα διατηρείται στη βάση salted+hashed. (Μπορείτε να δείτε ενδεικτικά βοηθητικές συναρτήσεις για τη διαδικασία παραγωγής salt και για τη χρήση συναρτήσεων hash, στο παράδειγμα 06-d στη σελίδα του μαθήματος).
     - **2.1.2.** Παρακολούθηση συνόδου (session management): Εάν η σύνδεση είναι επιτυχής, θα πρέπει η εφαρμογή σας να διατηρεί πληροφορία που αφορά τη σύνοδο του συγκεκριμένου χρήστη , από τη στιγμή της σύνδεσης μέχρι την αποσύνδεση του χρήστη. Πληροφορία που μπορεί να διατηρείται στο session είναι ενδεικτικά το username και ο ρόλος του εκάστοτε χρήστη. Η πληροφορία συνόδου θα πρέπει να "ακολουθεί" τον χρήστη κατά την πλοήγησή του στην εφαρμογή, μέχρι την αποσύνδεσή του.
     - **2.1.3.** Αποσύνδεση (logout): Κάθε χρήστης θα πρέπει να αποσυνδέεται με ασφάλεια από την εφαρμογή. Κατά την αποσύνδεση να υλοποιήσετε τον καθαρισμό της cache και την ακύρωση του session (invalidate session).
   - **2.2.** Υλοποίηση λειτουργιών ανά κατηγορία χρήστη:
     - **2.2.1.** Πελάτες: Προβολή προγράμματος προβολών ταινιών για συγκεκριμένο χρονικό διάστημα, διαθεσιμότητα για κάποια προβολή, κράτηση εισιτηρίων για κάποια προβολή, προβολή ιστορικού κρατήσεων κτλ.
     - **2.2.2.** Διαχειριστές περιεχομένου: Ολοκλήρωση των λειτουργιών που υλοποιήσατε στην Άσκηση 2 και άλλων απαραίτητων λειτουργιών (π.χ. τροποποίηση προγράμματος προβολών).
     - **2.2.3.** Διαχειριστές εφαρμογής: Προσθήκη και διαγραφή διαχειριστή περιεχομένου.
3. **Ολοκλήρωση επιπέδου Δεδομένων**
   - **3.1.** Μπορείτε να προβείτε σε όποιες τροποποιήσεις θεωρείτε απαραίτητες στη βάση δεδομένων που έχετε δημιουργήσει από την προηγούμενη άσκηση. Προσθέσετε επιπλέον δοκιμαστικά δεδομένα στη βάση όπου απαιτείται.
4. **Ολοκλήρωση επιπέδου προβολής (html, jsp)**
   - **4.1.** Σε συνέχεια του προηγούμενου βήματος, υλοποιήστε όλες τις απαραίτητες σελίδες που χρειάζονται για την προβολή/εμφάνιση των αποτελεσμάτων, όπως jsp και html σελίδες. Ενδεικτικά, μπορείτε για κάθε λειτουργία των χρηστών, να δημιουργήσετε μία jsp σελίδα που θα λαμβάνει τα αποτελέσματα από το αντίστοιχο servlet και θα δημιουργεί δυναμικά τη σελίδα που θα προβάλει το αντίστοιχο αποτέλεσμα.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1 Γενική Περιγραφή της λύσης

Στο τελευταίο στάδιο της εργασίας υλοποιήσαμε την σελίδα της επιχείρησης πλήρως με όλες τις λειτουργίες της και χωρίς να έχει καμία τρύπα ασφαλείας στους ελέγχους της κτλ. Αρχικά για το **password encryption** των χρηστών χρησιμοποιήσαμε **MD5**, όσο για το μυστικό κωδικό **salt** χρησιμοποιήθηκε ένας τυχαίος συνδυασμός. Το password encryption χρησιμοποιείται σε δυο μέρη της εργασίας, τα οποία είναι κατά την σύνδεση και εγγραφή του χρήστη.
Στην συνέχεια δημιουργήσαμε μια στήλη **is_online** στον πίνακα **users** η οποία θα αλλάζει τιμή όταν ο χρήστης συνδεθεί και όταν η session του καταστραφεί. Τον έλεγχο για καταστροφή του session κάνει η κλάση **SessionListener.java** η οποία εκτελεί ένα event (Listener) και αυτόματα θα αλλάξει την τιμή της στήλης. Επίσης το αρχείο **default.jsp** περιέχει το menu που θα ακολουθεί τον χρήστη σε όλες τις σελίδες και ανάλογα με τις άδειες που έχει θα του εμφανίζει τις κατάλληλες επιλογές.
Επιπρόσθετα, η υλοποιημένες κλάσεις που έχουν σκοπό την οργάνωση του κώδικα (prepareStatements, queries, κτλ) καλούνται αντίστοιχα εκεί που χρειάζεται.
Οι υπόλοιπες **JSP** σελίδες ακολουθούν την ίδια δομή κώδικα περίπου με μόνο αλλαγές στα queries, στους ελέγχους ανάλογα και στο **CSS**.

2 Βάση Δεδομένων

2.1 Μοντέλο Οντοτήτων-Σχέσεων

Το Μοντέλο Οντοτήτων-Σχέσεων βρίσκεται στο αρχείο **Montelo-Ontotiton-Sxeseon.docx**.

2.2 Επεξήγηση Πινάκων

Με την εντολή **CREATE TABLE** ορίσαμε τους **πίνακες** μας με τα ονόματα «**users**» «**films**» «**provoles**» «**screens**» «**transactions**».

- Ο πίνακας «**users**» περιέχει εφτά (**8**) στήλες: **user_id**, **username**, **password**, **full_name**, **email**, **telephone**, **is_online**, **usertype**.
- Ο πίνακας «**films**» περιέχει τέσσερεις (**4**) στήλες: **film_id**, **title**, **categories**, **description**.
- Ο πίνακας «**provoles**» περιέχει εφτά (**7**) στήλες: **provoli_id**, **film_id**, **screen_id**, **start_date**, **end_date**, **number_of_reservations**, **is_available**.
- Ο πίνακας «**screens**» περιέχει τέσσερεις (**4**) στήλες: **screen_id**, **title**, **screen_number**, **number_of_seats**, **is_3d**.
- Ο πίνακας «**transactions**» περιέχει πέντε (**5**) στήλες: **transaction_id**, **user_id, provoli_id**, **seats**, **transaction**.
- Οι στήλες **users.user_id, films.film_id, provoles.provoli_id, screens.screen_id, transactions.transaction_id** έχουν τους αντίστοιχους τύπους δεδομένων **BIGSERIAL** (64-bit auto-increment integer) δηλαδή είναι ένας ακέραιος αριθμός 64-bit ο οποίος αυξάνεται κατά ένα κάθε φορά που προθέτεται μια νέα εγγραφή.
- Όσες στήλες έχουν τον περιορισμό **NOT NULL**, δεν μπορούν να πάρουν κενό όρισμα και πρέπει να έχουν αναγκαστικά μια.
- Τα πρωτεύων κλειδιά (**PRIMARY KEYS**) δηλαδή **users.user_id, films.film_id, provoles.provoli_id, screens.screen_id, transactions.transaction_id** είναι οι στήλες στις οποίες απαγορεύεται οποιεσδήποτε δύο εγγραφές στην ίδια στήλη να έχουν ταυτόχρονα την ίδια τιμή.

- Τέλος ο περιορισμός **UNIQUE** εξασφαλίζει ότι όλες οι τιμές των στηλών **users.email**, **users.username** και **film.title** θα είναι διαφορετικές.

2.3 Επεξήγηση Trigger και Συνάρτησης

Query:

```
CREATE OR REPLACE FUNCTION On_Transaction_Change_Res() RETURNS trigger AS
	$BODY$
	BEGIN
		IF (TG_OP = 'INSERT') THEN
			UPDATE Provoles PR
			SET Number_Of_Reservations = PR.Number_Of_Reservations + NEW.Reserved_Seats
		  WHERE PR.provoli_id = NEW.provoli_id;
			RETURN NEW;
		ELSIF (TG_OP = 'DELETE') THEN
			UPDATE Provoles PR
			SET Number_Of_Reservations = PR.Number_Of_Reservations - OLD.Reserved_Seats
		  WHERE PR.provoli_id = OLD.provoli_id;
			RETURN OLD;
		END IF;
	END;
	$BODY$
	LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION On_Transaction_Change_Res_Update() RETURNS trigger AS
	$BODY$
	BEGIN
		IF (TG_OP = 'UPDATE') THEN
			IF (NEW.Reserved_Seats IS NULL) THEN
				RAISE EXCEPTION 'Seats CANNOT BE NULL!';
			END IF;
			IF (NEW.Reserved_Seats < 1) THEN
				RAISE EXCEPTION 'Seats MUST BE 1 OR MORE!';
			ELSE
			  UPDATE Provoles PR
			  SET Number_Of_Reservations = NEW.Reserved_Seats
			  WHERE PR.provoli_id = NEW.provoli_id;
			END IF;
		END IF;
		RETURN NEW;
	END;
	$BODY$
	LANGUAGE plpgsql;

CREATE TRIGGER Change_Customer_Transaction
	AFTER INSERT OR DELETE
	ON Transactions
	FOR EACH ROW
	EXECUTE PROCEDURE On_Transaction_Change_Res();

CREATE TRIGGER Change_Customer_Transaction_Update
	BEFORE UPDATE OF Reserved_Seats
	ON Transactions
	FOR EACH ROW
	EXECUTE PROCEDURE On_Transaction_Change_Res_Update();
```

Δημιουργήσαμε δύο λειτουργίες (Που επιστρέφουν trigger) στην βάση δεδομένων που έχουμε.

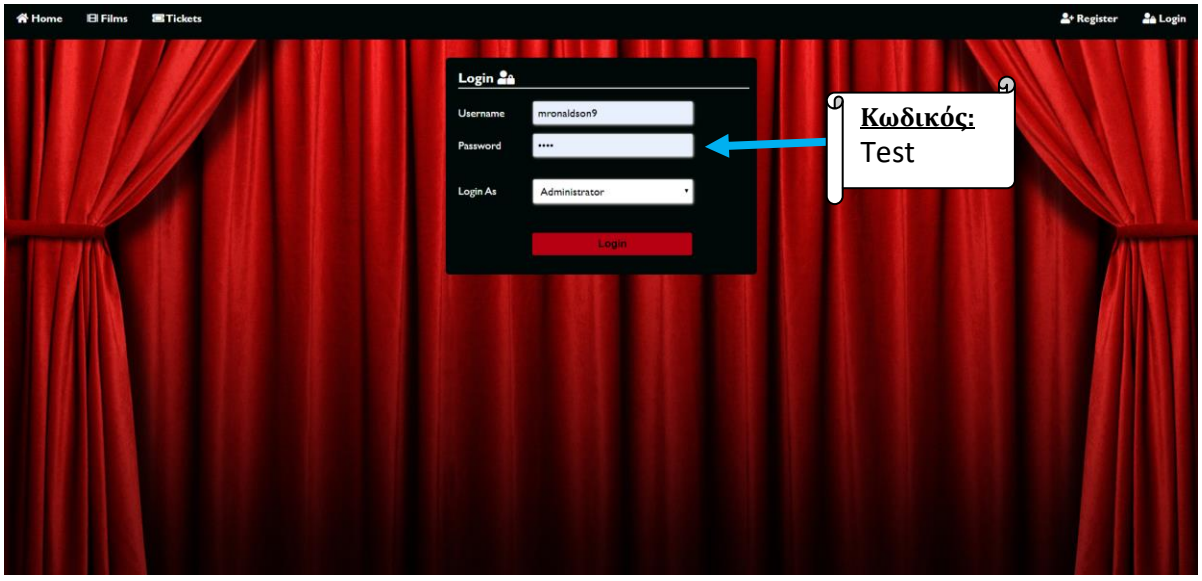
- Η πρώτη λειτουργία <<**On_Transaction_Change_Res()**>> ενεργοποιείται όταν γίνει μια αλλαγή (**UPDATE**) στις θέσεις που έχουν ήδη ζητηθεί. Στην περίπτωση μας χρησιμοποιούμε το **BEFORE UPDATE** στο **trigger** όταν γίνει αλλαγή στην στήλη **Reserved_Seats** στο πίνακα **Transactions**. Στην συνέχεια επεξεργαζόμαστε την νέα τιμή που δόθηκε και ελέγχουμε εάν το **TG_OP** είναι ίσο με **UPDATE**, τότε εκτελούνται και κάποιοι άλλοι έλεγχοι όπως εάν η νέα τιμή είναι **NULL** και εάν είναι μικρότερη του **1** διότι δεν θέλουμε να έχουμε **0** ή **αρνητικό αριθμό**. Εφόσον οι έλεγχοι επιτευχθούν επιτυχώς, θα ενημερωθεί η στήλη **Number_Of_Reservations** για την συγκεκριμένη προβολή και θα επιστραφεί η νέα τιμή που δόθηκε για την **Reserved_Seats**.
- Στην δεύτερη λειτουργία <<**On_Transaction_Change_Res_Update()**>> αντίστοιχα θα εκτελεστεί το **trigger** μετά την **εισαγωγή** ή **διαγραφή** από τον πίνακα και θα γίνουν οι κατάλληλες ενέργειες (Πρόσθεση ή αφαίρεση κρατημένων θέσεων στην **Number_Of_Reservations**).

3 Κώδικας προγράμματος

Ακολουθεί η αναλυτική περιγραφή του προγράμματος.

3.1 Κώδικας του αρχείου Login.jsp και Login_Request.jsp

Στο αρχείο **Login.jsp** υπάρχει η function <<**formCheck()**>> , στην συνάρτηση γίνεται δήλωση μεταβλητών **username** και **password** αλλά και της λίστας **errors**. Ακολούθως γίνεται έλεγχος για το αν τα πεδία **username** και **password** που έχει συμπληρώσει ο χρήστης είναι κενά και γίνεται ανάλογη ενημέρωση στην λίστα **errors** αν υπάρξουν. Εφόσον μετά τον έλεγχο της, υπάρξει κάποιο λάθος ενημερώνουμε με το ανάλογο μήνυμα τον χρήστη. Το αρχείο **Login_Request.jsp** καλείται όταν γίνεται υποβολή της φόρμας στο αρχείο **login.jsp**. Παίρνουμε τις παράμετρούς των πεδίων με την εντολή **request.getParameter**. Στην συνέχεια γίνεται σύνδεση στην βάση και προετοιμάζουμε ένα query το οποίο επιλέγει το **username** , **password** και **usertype** από την βάση το οποίο αντιστοιχίζει τις παράμετρούς που έδωσε ο χρήστης. Αν ταυτίζονται μεταξύ τους τότε συνδέεται στην σελίδα αλλιώς ενημερώνουμε τον χρήστη με το ανάλογο μήνυμα. Επίσης γίνεται αποκρυπτογράφηση στο κωδικού που έδωσε ο χρήστης και αμέσως γίνεται έλεγχος με query στην βάση αν ο κωδικός αντιστοιχεί με το όνομα του χρήστη και το είδος του χρήστη.

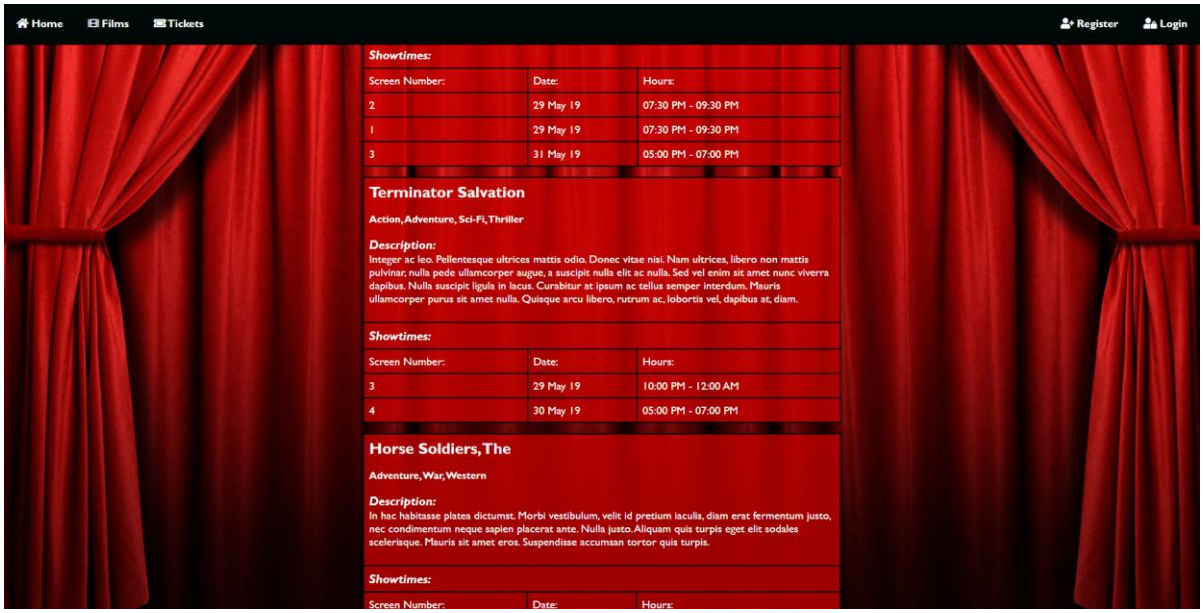


Στην βάση:

9	10	mronaldson9	9BEC8861661741...	Mel Ronaldson	mronal...	8321064101	true	Administrator

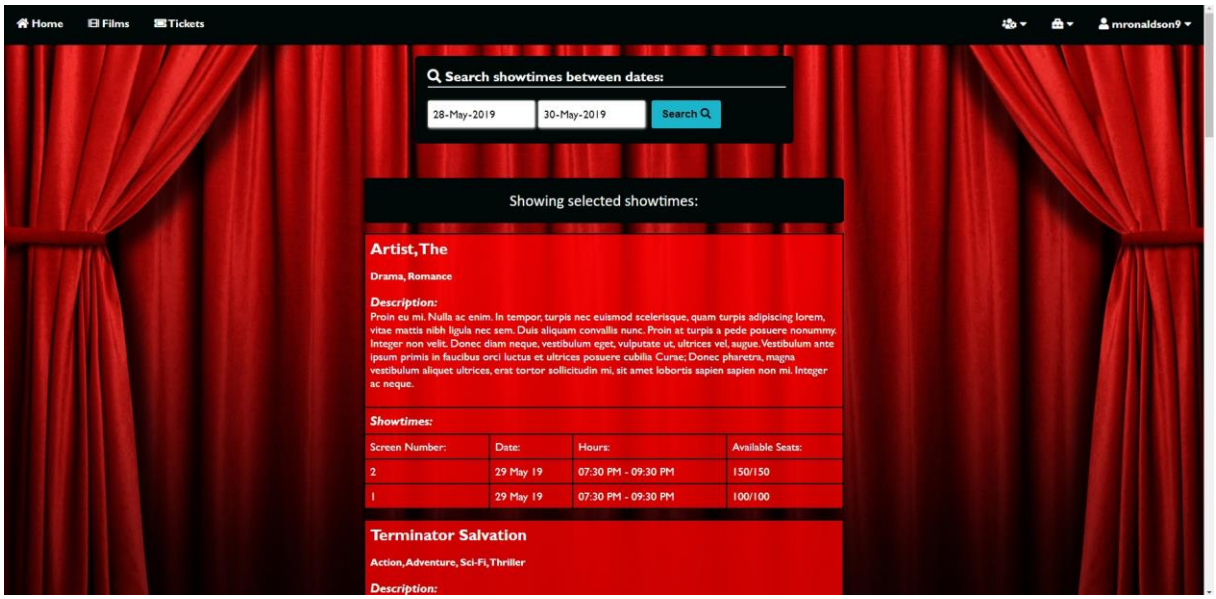
3.2 Κώδικας του αρχείου Films.jsp

Συνδέουμε το αρχείο **films.jsp** με την βάση δεδομένων που έχουμε, με τον πίνακα **films**. Χρησιμοποιούμε τα queries για να εμφανίσουμε τα στοιχεία των ταινιών, τις ημερομηνίες και τις ώρες που θα υπάρχει η προβολή, σε ποια οθόνη και τον τίτλο μαζί με την κατηγορία και το είδος της κάθε ταινίας. Επίσης έλεγχος για το αν είναι διαθέσιμη η ταινία και βγάζει το αντίστοιχο μήνυμα. Για να επιτευχθούν τα παραπάνω, γίνεται η αντιστοίχιση των στοιχείων με βάση το **title_id** και ακολούθως εμφανίζονται ανά γραμμή οι ταινίες με όλα τα στοιχεία.



3.3 Κώδικας του αρχείου Films_Specific_Showtimes.jsp

Στο αρχείο **Films.jsp** υπάρχει επιλογή ο χρήστης να κάνει αναζήτηση της ταινίας στην ημερομηνία που ο ίδιος επιθυμεί. Αντίστοιχα γίνεται ο έλεγχος των ταινιών που έχουν προβολή την ημερομηνία που έχει επιλεχθεί και εμφανίζονται στην σελίδα **Films_Specific_Showtimes.jsp**.



3.4 Κώδικας του αρχείου Previews_Reservations.jsp

Στο αρχείο **Previews_Reservations.jsp**, όταν ο χρήστης μεταβεί στην συγκεκριμένη σελίδα, πραγματοποιείται έλεγχος με query για να βρει τις προηγούμενες κρατήσεις που έχει κάνει. Αν δεν έκανε καμία κράτηση θα εμφανίσει το αντίστοιχο μήνυμα.

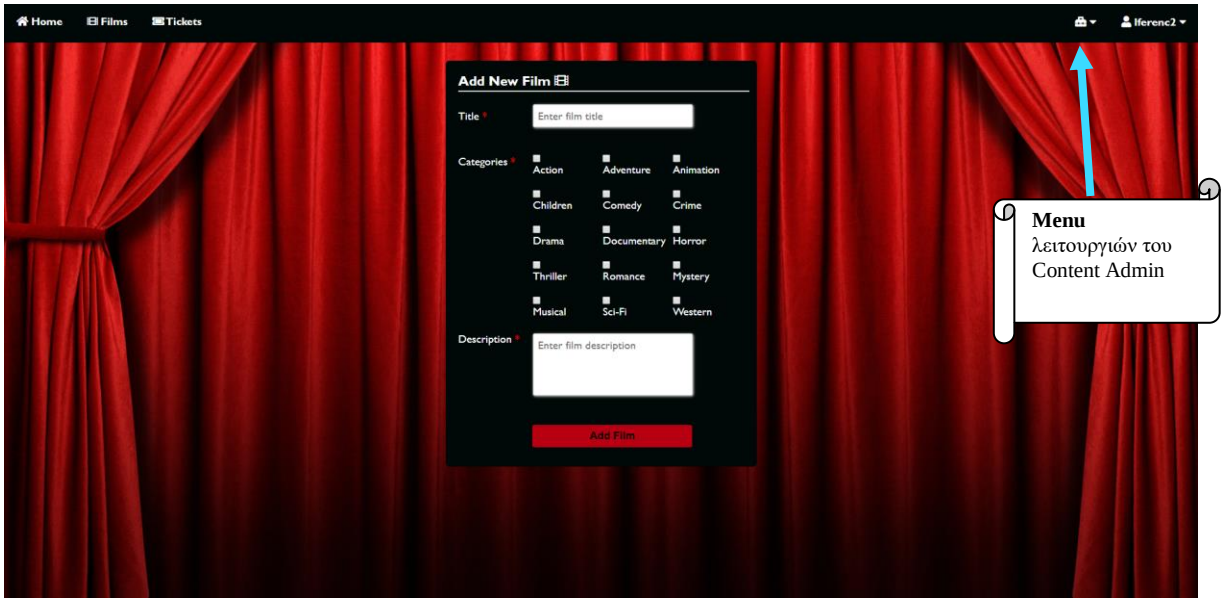
3.5 Κώδικας του αρχείου Panel_Users_View.jsp

Εάν ο χρήστης έχει την ιδιότητα ως διαχειριστής (Administrator), έχει την επιλογή με βάση το αρχείο **Panel_Users_View.jsp**, να εμφανίσει μια λίστα με όλους τους χρήστες, διαχειριστές

περιεχομένου και διαχειριστές που έχουν λογαριασμό. Επίσης όχι μόνο μπορεί να δει τα στοιχεία τους αλλά και αν είναι συνδεδεμένη εκείνη την στιγμή.

3.6 Κώδικας του αρχείου Panel_Film_Add.jsp και Panel_Film_Add_Request.jsp

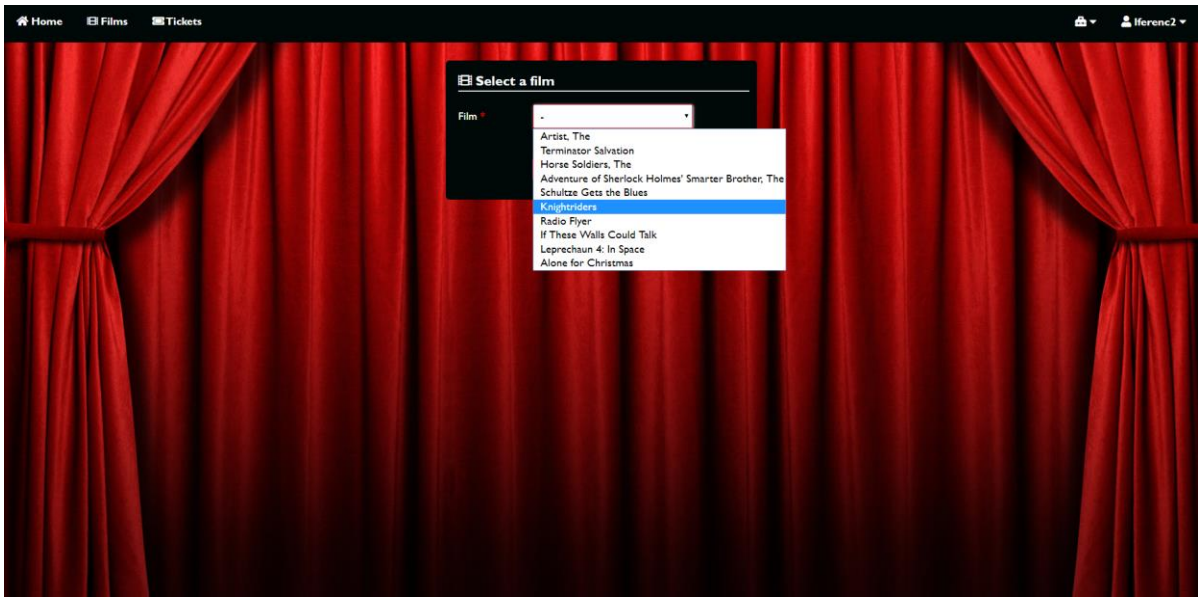
Αρχικά στο αρχείο **Panel_Film_Add.jsp** έχουμε μια φόρμα στην οποία ένας διαχειριστής μπορεί να προσθέσει μια ταινία στην βάση. Στην συνέχεια το αρχείο **Panel_Film_Add_Request.jsp** που θα καλεστεί όταν γίνει **submit** από την φόρμα του προηγούμενου αρχείου μέσω του action. Χρησιμοποιήσαμε ένα πεδίο τύπου **hidden** για να προσθέτουμε και να αφαιρούμε τις τιμές των **checkboxes** που είναι επιλεγμένες (**checked**). Μετά από αυτό παίρνουμε τις τιμές τις προηγούμενης φόρμας και γίνεται έλεγχος εάν υπάρχει ήδη η ταινία μέσα στον πίνακα, επίσης, σε όλες τις σελίδες που ένας χρήστης πρέπει να είναι συνδεδεμένος για εκτελέσει τις λειτουργίες τους, γίνεται έλεγχος εάν ο χρήστης είναι συνδεδεμένος γιατί μπορεί να έληξε το session του ή να μην έχει τα κατάλληλα δικαιώματα (Δηλαδή να μην είναι admin, κτλ.). Επομένως θα εμφανιστεί το κατάλληλο μήνυμα και όλοι οι έλεγχοι είναι εντάξει θα προστεθεί η ταινία στον πίνακα.

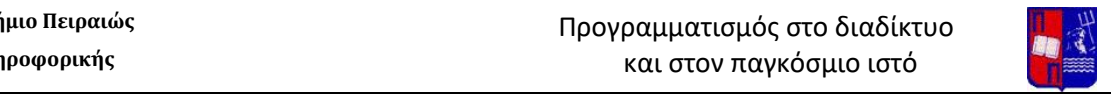


3.7 Κώδικας του αρχείου Tickets_1.jsp, Tickets_2.jsp Tickets_Final.jsp και Tickets_Final_Request.jsp

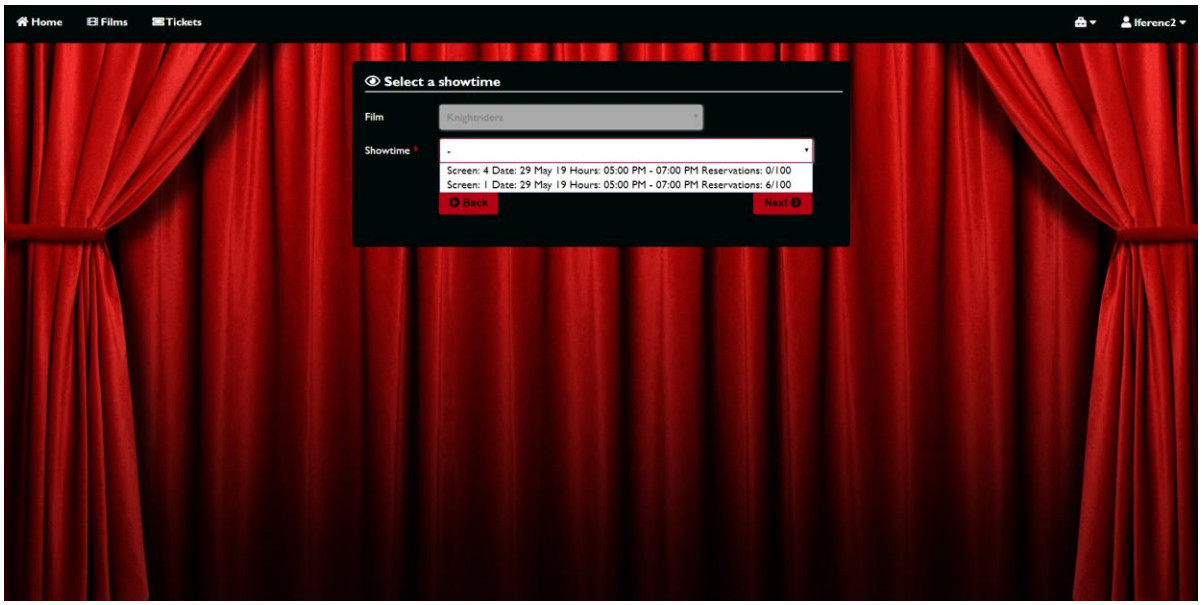
Σε αυτό το κομμάτι υλοποιήθηκε η φόρμα με την οποία ο χρήστης θα μπορεί να αγοράζει εισιτήρια για τις ταινίες και προβολές που επιθυμεί.

Καταρχάς, στο αρχείο **Tickets_1.jsp** παίρνουμε όλη την στήλη **title** του πίνακα **films** από την βάση και την τοποθετούμε ανά γραμμή μέσα στο **select** σαν **option** έτσι ώστε να έχουμε επιλογή για όλες τις ταινίες όπως φαίνεται στο πιο κάτω screenshot.

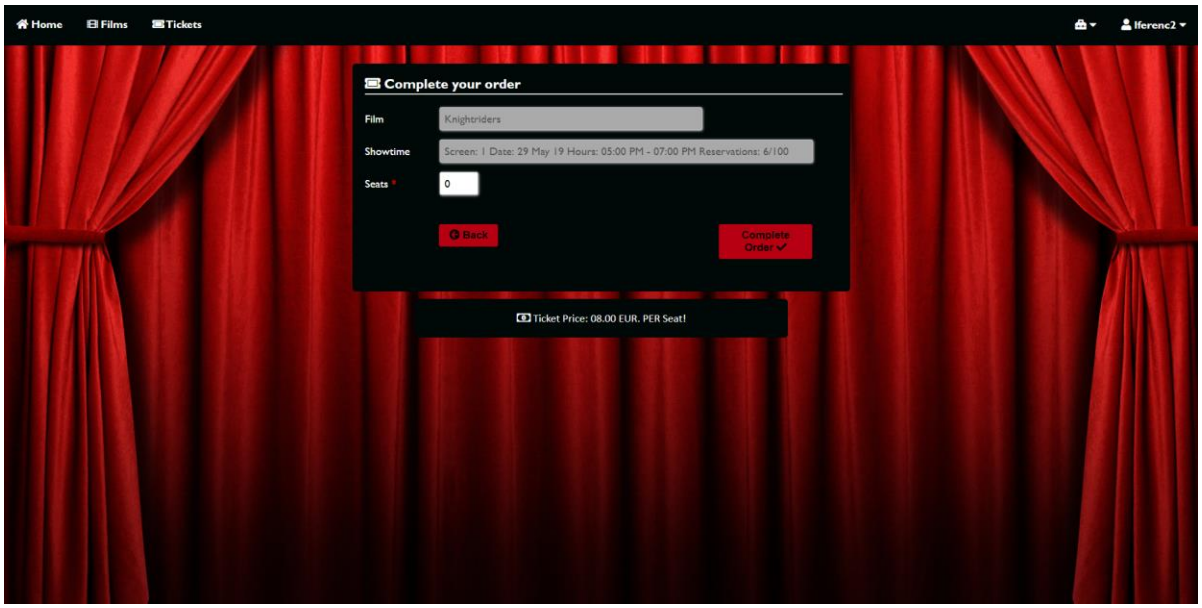




Στην συνέχεια αφού επιλεχθεί ταινία και γίνει submit μεταφερόμαστε στο αρχείο **Tickets_2.jsp** όπου θα εμφανιστεί άλλη φόρμα με την προηγούμενη επιλογή του χρήστη στο πεδίο ταινίας και τώρα θα έχει την επιλογή να διαλέξει προβολή. Επίσης χρησιμοποιείται πεδίο τύπου **hidden** το οποίο θα βοηθήσει στην μεταφορά των τιμών στα άλλα αρχεία (Το ίδιο και στο αρχείο **Tickets_Final.jsp**).

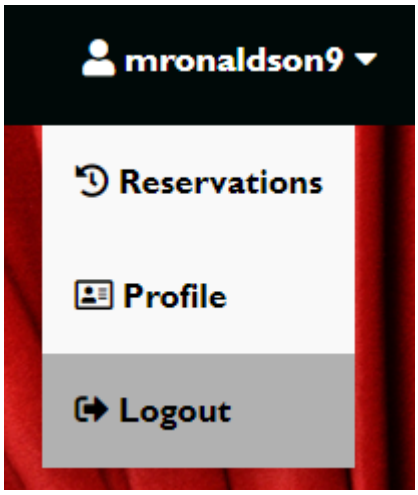


Εφόσον γίνουν οι επιλογές και ο χρήστης κάνει υποβολή θα αποθηκευτεί η συναλλαγή στον πίνακα **transactions** και θα γίνει αλλαγή στο **number_of_reservations** της προβολής αυτομάτος λογο του **trigger** που δημιουργήσαμε πριν.



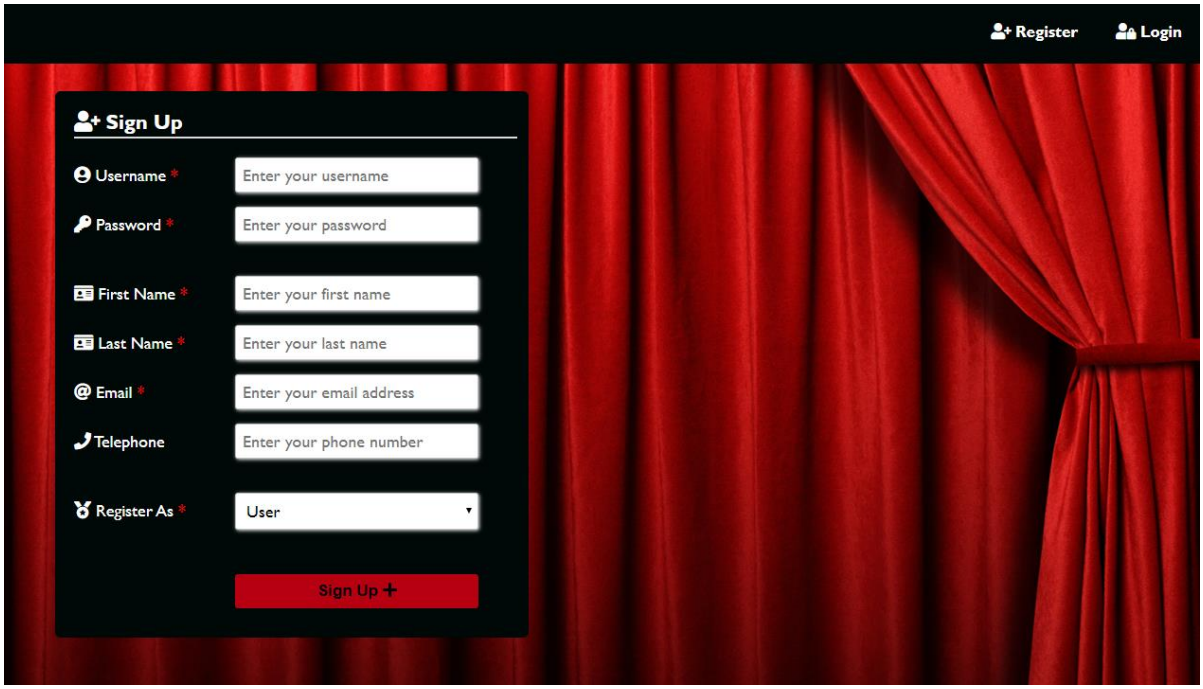
3.8 Κώδικας του αρχείου logout.jsp

Όταν ο χρήστης επιλέξει την επιλογή του "**Logout**" στο menu τότε το session του θα καταστραφεί. Επίσης θα γίνει clear το cache με τις εντολές **setHeader** διότι το έγγραφο δεν πρέπει ποτέ να αποθηκεύεται προσωρινά και δεν πρέπει να αποθηκεύεται σε προσωρινή θέση στο δίσκο. Αυτή η κεφαλίδα προορίζεται για την αποφυγή ακούσιων αντιγράφων ευαίσθητων πληροφοριών.



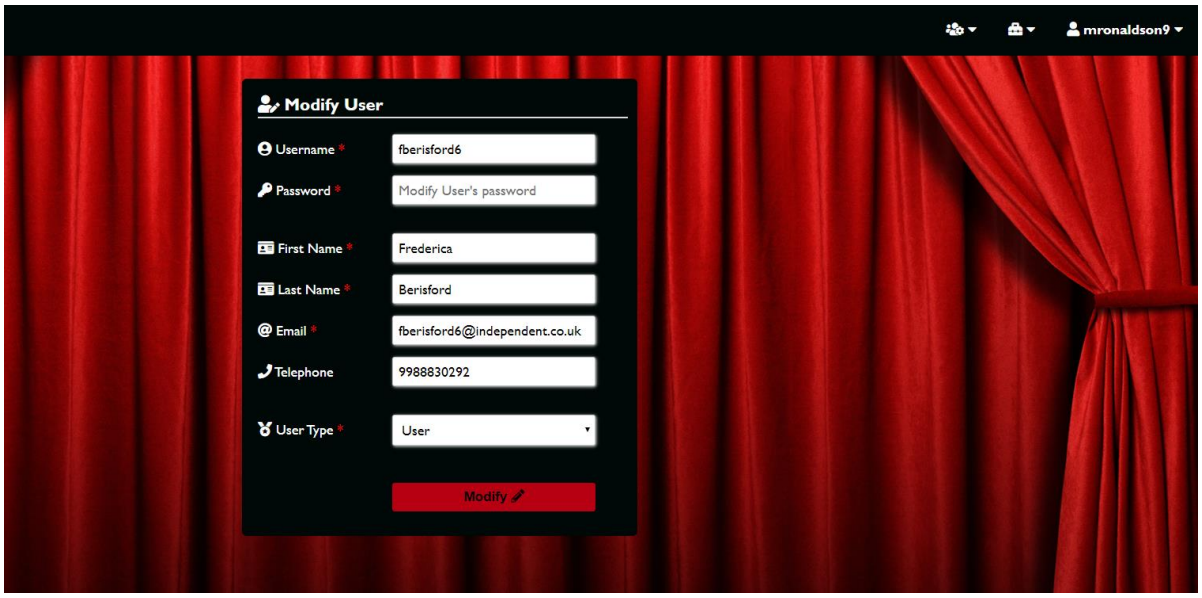
3.9 Κώδικας του αρχείου Register.jsp και Register_Request.jsp

Όταν ο χρήστης συμπληρώσει την ανάλογη φόρμα με τα στοιχεία του, θα γίνουν οι αντίστοιχοι έλεγχοι για να έχουμε σωστά δεδομένα. Στην συνέχεια στο **Register_Request.jsp** θα γίνει κωδικοποίηση του κωδικού σε **MD5** χρησιμοποιώντας τον συνδυασμό **salt** που έχουμε. Τέλος θα προσδεθεί ο χρήστης στον πίνακα **users**.



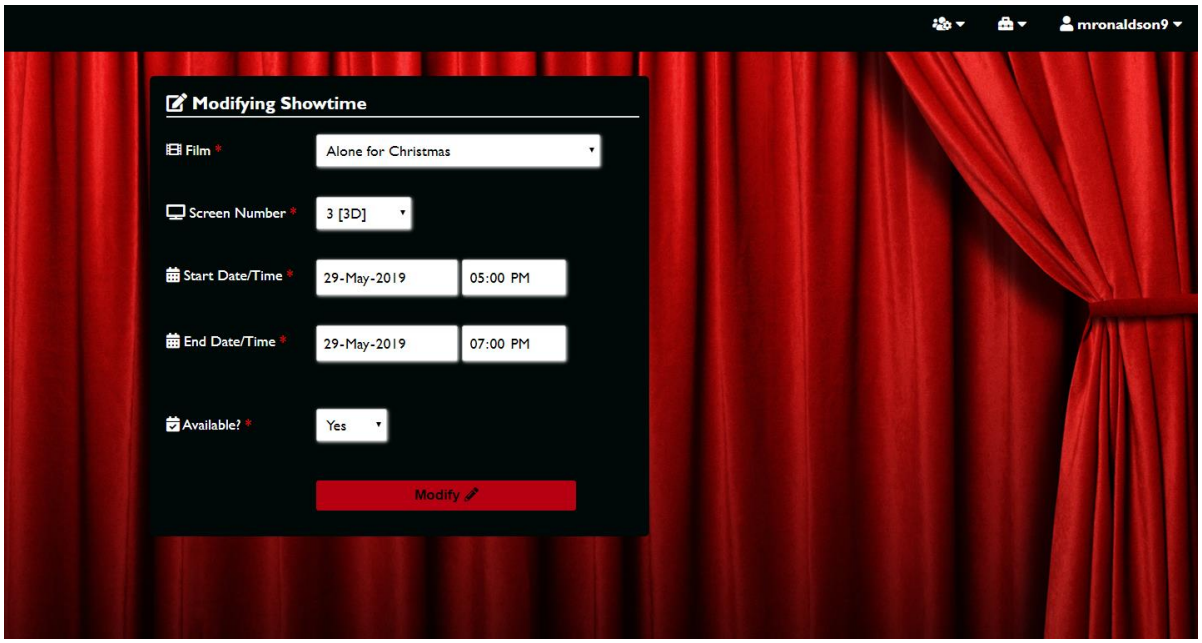
3.10 Κώδικας του αρχείου Panel_User_Modify.jsp, Panel_User_Modify_2.jsp και Panel_User_Modify_Request.jsp

Όταν ένας διαχειριστής μεταβεί σε αυτή την σελίδα τότε θα έχει την επιλογή να διαλέξει πιο χρήστη θέλει να επεξεργαστεί και στην συνέχεια αφού γίνει υποβολή της φόρμας, στο αρχείο **Panel_User_Modify_2.jsp** τα **inputs** θα πάρουν τιμές αντίστοιχα με τα δεδομένα του χρήστη από την βάση και αφού γίνει υποβολή ξανά θα γίνει **UPDATE** στον πίνακα.



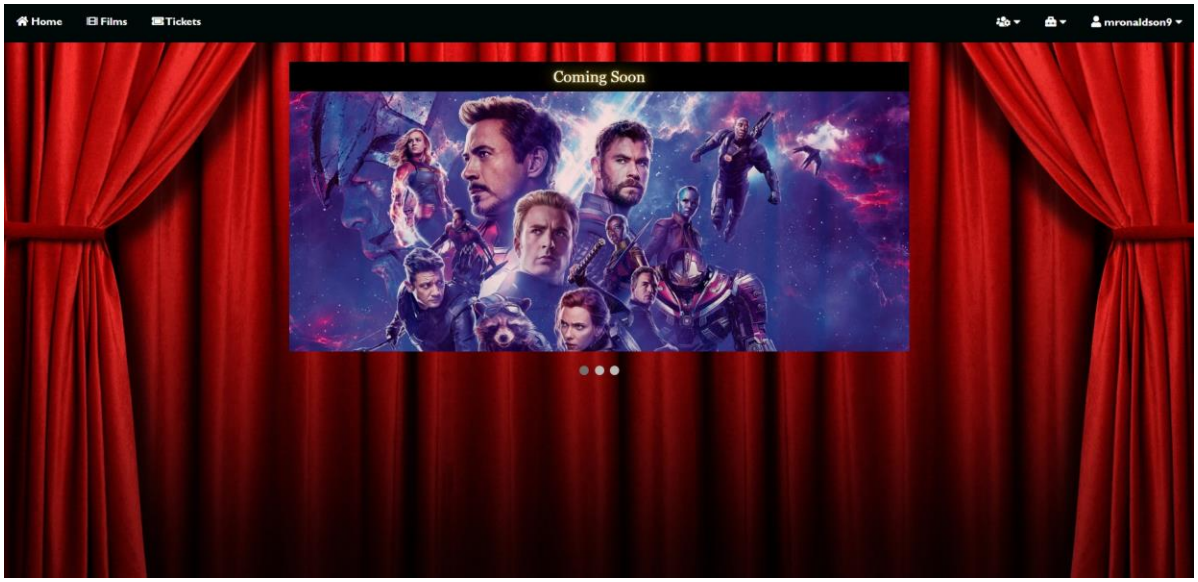
3.11 Κώδικας του αρχείου Showtime_Update.jsp, Showtime_Update_2.jsp και Showtime_Update_Request.jsp

Παρομοίως όπως και στο αρχείο **Panel_User_Modify.jsp** σε αυτή την περίπτωση θα γίνει για την αντίστοιχη προβολή.



3.12 Κώδικας του αρχείου index.jsp

Στο αρχείο **index.jsp** βρίσκεται η αρχική σελίδα του προγράμματος. Στην αρχική σελίδα υπάρχει ένα πλαίσιο με τρείς διαφάνειες που περιέχουν τις τρείς ταινίες που θα είναι διαθέσιμες στο μέλλον για να κάνουν κράτηση οι χρήστες.



4 Βιβλιογραφικές Πηγές

4.1 IntelliJ IDEA Ultimate 2019.1.3

Το IDE που χρησιμοποιήθηκε για την εργασία.

4.2 AdoptOpenJDK with Hotspot 8.212

Το JDK που χρησιμοποιήθηκε για την εργασία (https://adoptopenjdk.net/).

4.3 PGAdmin 4.8

4.4 PostgreSQL 10.5-1

4.5 Tomcat 8.5.39

https://tomcat.apache.org/download-80.cgi

4.6 PostgreSQL-42.2.5 (Driver)

https://jdbc.postgresql.org/download.html

4.7 Mockaroo

Για την παραγωγή **αληθοφανών** δεδομένων.

4.8 SERVLETS και ΣΕΛΙΔΕΣ ΔΙΑΚΟΜΙΣΤΗ JAVA – ΤΕΧΝΟΛΟΓΙΕΣ ΠΥΡΗΝΑ 2Η ΑΜΕΡΙΚΑΝΙΚΗ ΕΚΔΟΣΗ – MARTY HALL & LARRY BROWN

4.9 Συστήματα Βάσεων Δεδομένων 6Η Έκδοση – Abraham Silberschatz, Henry F. Korth & S. Sudarshan

4.10 Slideshow

https://www.w3schools.com/howto/howto_js_slideshow.asp

4.11 Menu

https://www.w3schools.com/css/css_navbar.asp

4.12 Ιδέες για τον τρόπο που θα γίνει το registration/login system

https://www.javatpoint.com/registration-form-in-jsp

4.13 Από που μάθαμε για το Μοντέλο Σχέσεων Οντοτήτων

http://www.cs.uoi.gr/~pitoura/courses/db/db09/slides/er09.pdf

4.14 Βοήθεια σχετικά με τα regex στο JavaScript

https://stackoverflow.com/questions/41975496/regex-for-names-validation-allow-only-letters-and-spaces/41975670

4.15 Βοήθεια σχετικά με την κλάση SessionListener.java

https://stackoverflow.com/questions/24764083/knowing-about-all-sessions-currently-active-in-jsp

4.16 Βοήθεια για να καθαρίσουμε το cache όταν ο χρήστης αποσυνδεθεί

https://stackoverflow.com/questions/43480003/clearing-cache-after-user-logout-in-jsp-no-scriptlet-allowed

4.17 Βοήθεια σχετικά με το πως θα γίνει το Password Encryption/Decryption

https://stackoverflow.com/questions/47537908/how-do-i-hash-and-salt-a-password-into-mysql-database-using-a-servlet

Κώδικας Κλάσης Encryption.java

https://stackoverflow.com/questions/20832008/jsp-simple-password-encryption-decryption

Εργαστηριακό Παράδειγμα: 06b-ExamplePost

4.18 FontAwesome (Library)

Χρησιμοποιήθηκε το library **FontAwesome** για την εισαγωγή εικόνων στο **Menu** και σε άλλα σημεία της εφαρμογής.
(https://fontawesome.com/)

4.19 Σελίδα για την εύρεση εικόνων από ταινίες

https://wall.alphacoders.com/by_category.php?id=20&name=Movie+Wallpapers