ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής

Εργασία Μαθήματος «Προγραμματισμός στο διαδίκτυο και στον παγκόσμιο ιστό»

3	***3η άσκηση - 2023***
Όνομα φοιτητή – Αρ. Μητρώου (όλων σε περίπτωση ομαδικής εργασίας)	Κούκος Κων/νος – Π21070
	Βασιλείου Αλέξιος – Π21009
	Κρόιτορ Καταρτζίου Ιωάν – Π21077
Ημερομηνία παράδοσης	24/06/2023

ΣΗΜΑΝΤΙΚΟ

Για την σωστή εκτέλεση της εργασίας πρέπει να γίνουν τα εξής:

1. **Στο context.xml του tomcat (εμείς χρησιμοποιήσαμε v9.71-75) server πρέπει να μπει το εξής resource με τον κωδικό του mysql server της συσκευής στην οποία θα τρέξει**

```
2.     <Resource name="jdbc/cinema_last" auth="Container"
   type="javax.sql.DataSource"
3.              maxTotal="100" maxIdle="30"
   maxWaitMillis="10000"
4.              username="root" password="toor"
5.              driverClassName="com.mysql.jdbc.Driver"
6.              url="jdbc:mysql://localhost:3306/cinema_last"
```

2. **Στο add_image.java (servlet) στο absolute path string πρέπει το αρχικό path να αλλάξει με το αντίστοιχο της συσκευής στο οποίο θα τρέξει. Αυτό συμβαίνει διότι όταν προσπαθήσαμε να βάλουμε να παίρνει δυναμικά το path, έπαιρνε αυτό του metadata και όχι του project.**

Εκφώνηση της άσκησης

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-b 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.

1 Κλάσεις προγράμματος (αρχικές)

Αρχικά, διατηρήσαμε τις κλάσεις το βασικό πακέτο πακέτο κλάσεων που είχε υλοποιηθεί στην πρώτη εργασία, δηλαδή τις Admin, Cinema, ContentAdmin, Customer, Film, Main, Provoli και User. Παρακάτω παραθέτω μία συνοπτική των επιπλέον λειτουργειών της κάθε μιας, εφόσον η αναλυτική λειτουργία τους περιλαμβάνεται στην εκφώνηση της 1ης άσκησης.

Ως κύρια κλάση των διάφορων χρηστών ορίσαμε την Users και με πεδία το name, username, password, user_type, Users_Array. Επιπλέον, περιλαμβάνει τις login(), logout() μεθόδους καθώς και τον constructor αλλά και τους απαραίτητους getters/setters. Προχωρώντας, κάθε μία από τις κλάσεις Customers, ContentAdmins, Admins κάνει extend την Users αφού στην αφαιρετική τους μορφή αυτές ανήκουν στην κατηγορία χρηστών, άρα οι constructors τους αρχικοποιούνται μέσω του super keyword, ενώ περιέχει και τους απαραίτητους getters/setters αλλά και τις μεθόδους που ορίζονται στην εκφώνηση της άσκησης. Επιπροσθέτως, η κλάση Customers περιέχει το πεδίο dateofBirth, η κλάση ContentAdmin τις μεθόδους searchFilm() η οποία αναζητά βάση id την ταινία της κλάσης Film, και την createProvoli() η οποία δημιουργεί μία προβολή. Συνεχίζοντας, η τρεις αυτές κλάσεις έχουν πρόσβαση στις κλάσεις Film, Cinemas, Provoles, οι οποίες διαχειρίζονται ταινίες, κινηματογράφους και προβολές αντίστοιχα. Αυτές οι κλάσεις περιέχουν constructors, κατάλληλους getters/setters αλλά, τις βασικές μεθόδους που αναγράφονται στην εκφώνηση, όπως και κάποιες πρόσθετες. Πιο συγκεκριμένα, η Film έχει επιπρόσθετα τα πεδία film18 (αν η ταινία είναι άνω των 18), filmDuration (διάρκεια ταινίας), filmDateofPremiere (ημ/νία πρεμιέρας), Films_Array (η λίστα με όλες τις ταινίες), η Cinema το πεδίο Cinemas_Array (λίστα με τα σινεμά) και Provoles το πεδίο Provoles_Array (λίστα με τις προβολές) αλλά και τη μέθοδο checkProvoliIsAvailable() για να ελέγχουμε εάν είναι διαθέσιμη μία προβολή. Επιπλέον, συμπεριλάβαμε την βιβλιοθήκη java.time.LocalTime ώστε να παίρνουμε την τρέχουσα ώρα. Όσον αφορά την MainClass, δημιουργούμε αντικείμενα για κάθε κλάση ενώ επιπλέον τα κάνουμε serialize & deserialize σε .txt αρχεία (με χρήση exceptions). Να σημειώσουμε σε αυτό το σημείο ότι όλες η κλάσεις κάνουν implement το interface Serializable για να μπορεί να εφαρμοστεί το Serialize. Τέλος, όσον αφορά τα exceptions, τα προσθέσαμε μόνο στο Serialize, καθώς δεν είναι απαραίτητο σε κάποιο άλλο σημείο σε αυτό το στάδιο της εφαρμογής.

2 Βάση Δεδομένων & σύνδεση με Tomcat

2.1 Βάση Δεδομένων

Συνεχίζοντας, όπως αναφέρεται και στην εκφώνηση, πήραμε την το δείγμα βάσης που υπήρχε στο gunet και το διαμορφώσαμε σύμφωνα με τις απαιτήσεις του δικού μας προγράμματος (αλλάξαμε το σχεσιακό μοντέλο σε σχέση με τη 2η εργασία). Σημειώνουμε πως όσα πεδία των πινάκων δεν αναλύσαμε παρακάτω, έγινε καθώς είναι αυτοπροσδιορίζονται και είναι προφανής η χρήση τους.

Ξεκινώντας, ο πίνακας user έχει 5 πεδία (username, password, create_time, type, DATE_OF_BIRTH), όπου create_time αντιστοιχεί στην ημερομηνία που δημιουργήθηκε ο χρήστης, type στον τύπο του χρήστη (admin=0, contentAdmin=1 και customer=2) και DATE_OF_BIRTH στην ημερομηνία γέννησης του χρήστη. Ως primary key ορίσαμε το username.

Επιπρόσθετα, ο πίνακας FILMS περιέχει 9 πεδία (ID, TITLE, LENGTH, CATEGORY, DESCRIPTION, DIRECTOR, PREMIERE, POSTER, IMAGELOG) όπου PREMIERE είναι η ημερομηνία της πρεμιέρας, POSTER είναι η εικόνα που θέτουμε στην αντίστοιχη ταινία και είναι τύπου LOGBLOB δηλαδή binary large object που χρησιμοποιείται γενικότερα για την αποθήκευση μεγάλων complex files (εικόνα , βίντεο κλπ.), ενώ IMAGELOG είναι το path της εικόνας που αποθηκεύτηκε. Ως primary key ορίσαμε το ID.

Επίσης ο πίνακας PROVOLES περιέχει 8 πεδία (FILMS_ID, FILMS_TITLE, CINEMAS_ID, ID, DAY, NUMBER_OF_RESERVATIONS, SEATS, SEAT_NUM). Το πεδίο FILMS_ID αναφέρεται στο αντίστοιχο πεδίο ID του πίνακα FILMS, ενώ το πεδίο CINEMAS_ID αναφέρεται στο αντίστοιχο πεδίο ID του πίνακα CINEMAS. Ως primary key ορίσαμε τον συνδυασμό πεδίων (FILMS_ID, CINEMAS_ID, ID).

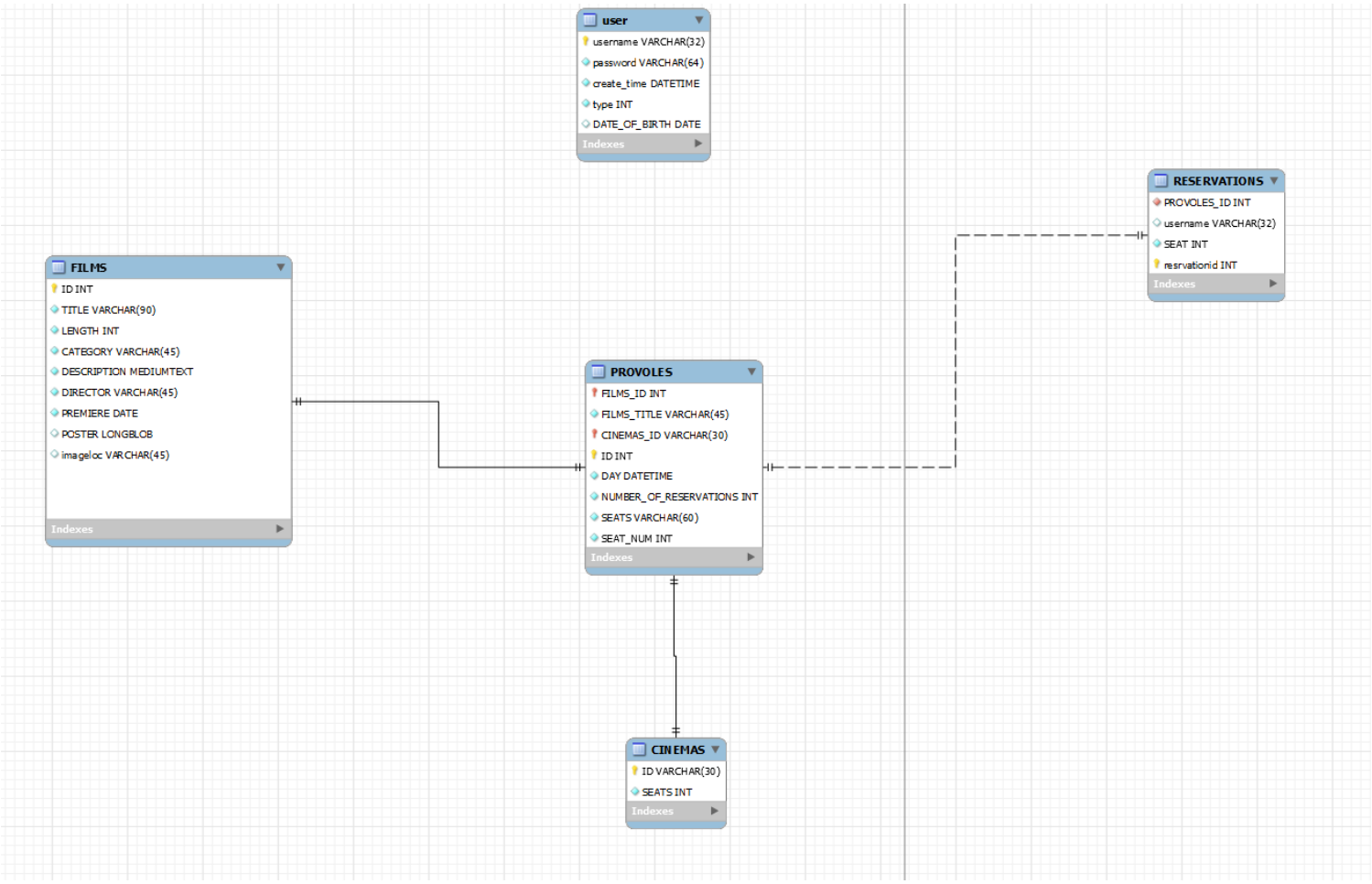
Ο πίνακας CINEMAS αποτελείται από 2 πεδία (ID, SEATS). Ως primary key ορίσαμε το ID.

Τέλος, ο πίνακας RESERVATIONS αποτελείται από 4 πεδία (PROVOLES_ID , username, SEAT, reservationid). Το πεδίο PROVOLES_ID αναφέρεται στο αντίστοιχο πεδίο ID του πίνακα PROVOLES. Ως primary key ορίσαμε το reservationid.

Όσον αφορά την σύνδεση των πινάκων ο user δεν συνδέεται με κάποιον πίνακα. Ύστερα, ο PROVOLES συνδέεται με τον FILMS και CINEMAS, με foreign keys τα FILMS_ID και CINEMAS_ID αντίστοιχα. Τέλος, ο RESERVATIONS έχει ως foreign keys το PROVOLES_ID και συνδέεται με τον PROVOLES.

2.2 Σχεσιακό Μοντέλο

Εικόνα 2. relational model in workbench



2.3 Σύνδεση με Tomcat

Για την σύνδεση με τον Tomcat Server (χρησιμοποιήσαμε τον Tomcat 9.0.71) ακολουθήσαμε τις οδηγίες που αναρτήθηκαν στο gunet προσθέτοντας τον κατάλληλο κώδικα στα αρχεία context.xml και web.xml .

2.4 Ρύθμιση Tomcat

```
<welcome-file-list>
   <welcome-file>log-in.jsp</welcome-file>
</welcome-file-list>
```

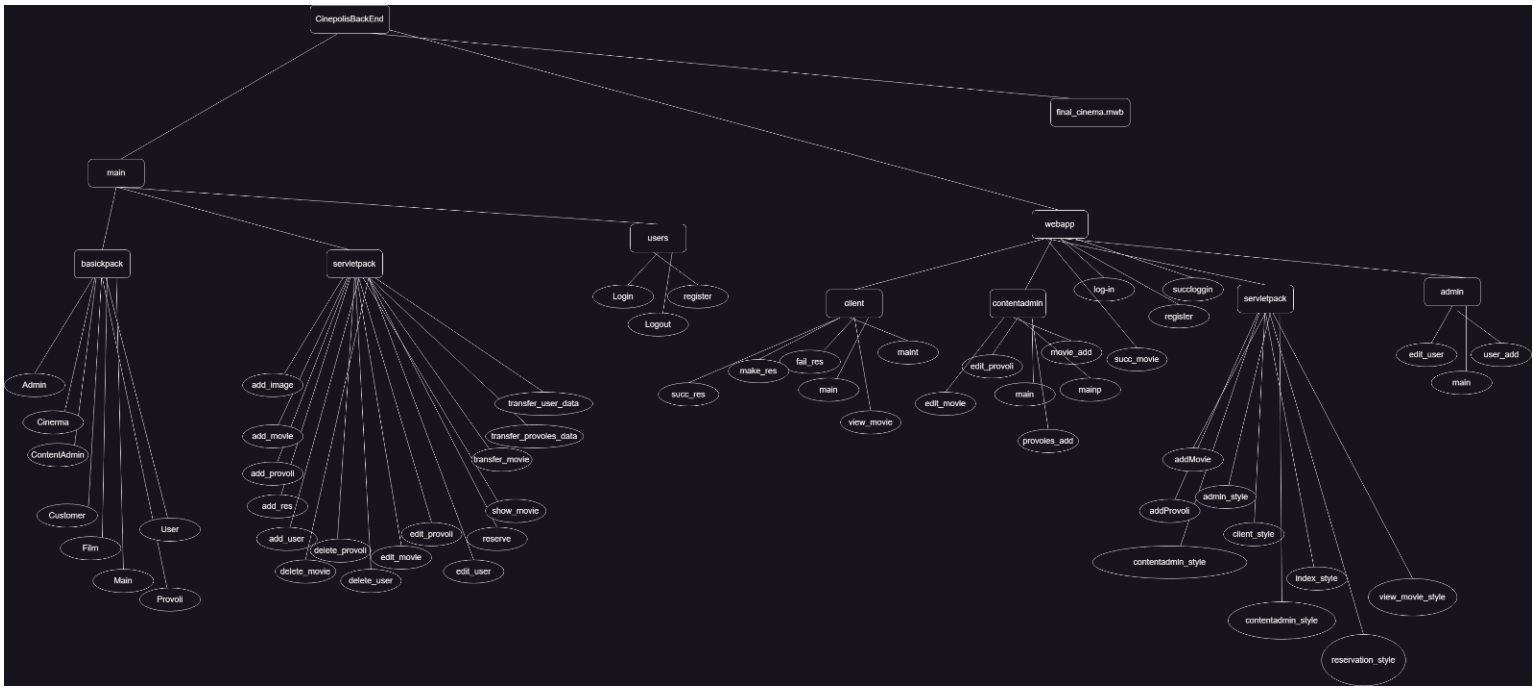
Με αυτή την ρύθμιση ο χρήστης μπορεί να τρέξει κατευθείαν το πρόγραμμα ή να μπει στο http://localhost:8080/CinepolisBackEnd/ για να συνδεθεί.

3 Servlets

3.1 Γενικά στοιχεία

Σε αυτήν την ενότητα θα παραθέσουμε κάποια γενικά δεδομένα και διάφορες τεχνικές οι οποίες είναι κοινές μεταξύ των διαφορετικών servlets που υλοποιήσαμε.

Πρώτα, όμως, θα παρουσιάσουμε το γενικό σχεδιάγραμμα του προγράμματος μας (για καλύτερη ανάλυση μπορείτε να ανατρέξετε στη φωτογραφία final_diagram.png που βρίσκεται στον φάκελο).



Επιπλέον, σημειώνουμε πως σε διάφορα σημεία του κώδικα χρησιμοποιούμε γραμμές System.out.println(“something”). Αυτό συμβαίνει για δική μας διευκόλυνση για να κατανοήσουμε πως συμπεριφέρεται ο κώδικας και που βρισκόμαστε σε κάθε σημείο.

3.1.1 Forms Method

Αρχίζοντας, σε όλες της φόρμες που έχουμε φτιάξει για εξαγωγή παραμέτρων δηλαδή στοιχείων (π.χ. για εγγραφή χρήστη, εισαγωγή ταινίας κλπ.), την doPost() μέθοδο η οποία διαχειρίζεται HTTP POST requests. Αυτή η επιλογή έγινε καθώς η συγκεκριμένη μέθοδο είναι ασφαλέστερη, διαχειρίζεται μεγαλύτερο όγκο δεδομένων ενώ παράλληλα επιτρέπει την τροποποίηση τους (π.χ. change date format, password hashing κλπ.).

```
protected void doPost(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
```

3.1.2 Controlling, Checking and Exception Handling

Σε όλες τις κλάσεις έχουμε θέσει ελέγχους όσον αφορά τα κενά πεδία (εάν δεν έχει συμπληρωθεί κάποιο από τα πεδία τότε η φόρμα δεν μπορεί να γίνει submit), καθώς και ελέγχους όσον την ορθότητα τον password (είτε στο login page αν αντιστοιχεί το password στον συγκεκριμένο χρήστη, είτε όταν κάνει register ελέγχεται αν password==retypedPassword). Όσον αφορά τον έλεγχο στο login page, αυτός γίνεται μεταξύ τον δύο hashed κωδικών (αυτός που βρίσκεται αποθηκευμένος στη βάση και αυτός που πληκτρολόγησε ο χρήστης στη login φόρμα αφού έγινε hashed), hash το οποίο θα περιγράψουμε σε παρακάτω υπό-ενότητα.

Για την διαχείριση των εξαιρέσεων κάναμε χρήση try-catch block καθ' όλη την έκταση του κώδικα και οι κλάσεις εξαιρέσεων που χρησιμοποιήθηκαν (εντός των catch) είναι SQLException (αφορά επικοινωνία με database), ClassNotFoundException (αφορά μη εύρεση κλάσης κατά την εκτέλεση του προγράμματος), NamingException (αφορά την πόσβαση ή χρήση των αντικειμένων που είναι αποθηκευμένα στο namespace του environmnet) και την Exception (που είναι η υπερ-κλάση των εξαιρέσεων και αφορά γενικές εξαιρέσεις. Οι προαναφερθείσες εξαιρέσεις αντιμετωπίζονται με ανακατεύθυνση σε άλλη (κατάλληλη) σελίδα (π.χ. αν δεν επιλέξει καμία θέση στο make_res.jsp και πατήσει "book now" θα τον ανακατευθύνει στη σελίδα fail_res.jsp) και printStackTree. Η ανακατεύθυνση σε άλλης σελίδα γίνεται μέσω της response.sendRedirect("page.jsp").

```
} catch (SQLException e) {
            e.printStackTrace();
            response.sendRedirect("main.jsp");
        } catch (ClassNotFoundException e) {

            e.printStackTrace();
        } catch (NamingException e) {

            e.printStackTrace();
        }
```

3.1.3 Σύνδεση με βάση δεδομένων

Στα περισσότερα από τα servlets ήταν απαραίτητο να συνδεθούμε με τη βάση δεδομένων είτε για να ελέγξουμε ορθότητα στοιχείων του χρήστη, είτε για εμφάνιση δεδομένων στην

οθόνη, είτε για την εισαγωγή/διαγραφή στοιχείων στην βάση (κλπ.). Σε κάθε μία από αυτές τις ενέργειες απαραίτητο ήταν να συνδεθούμε (και να αποσυνδεθούμε) από αυτήν. Για αυτό χρησιμοποιούμε ανάλογο κώδικα.

```
try {
            String RESOURCE_NAME = "jdbc/cinema_last";

              Class.forName("com.mysql.jdbc.Driver");
              Context initContext = new InitialContext();
              Context envContext = (Context)
initContext.lookup("java:/comp/env");

              DataSource ds = (DataSource)
envContext.lookup(RESOURCE_NAME);
              Connection conn = ds.getConnection();

              PreparedStatement statement;
              HttpSession session = request.getSession();
```

3.2 Ανάλυση servlets

Σε αυτήν την ενότητα θα παρουσιάσουμε περιληπτικά (αλλά επαρκώς) την λειτουργία των servlets καθώς και τις διάφορες τεχνικές (εφόσον υπάρχουν) που χρησιμοποιήθηκαν σε καθένα από αυτά. Σημειώνουμε πως δίπλα από κάθε servlet αναγράφουμε εντός παρενθέσεων τον τύπο χρηστών που αφορούν.

3.2.1 add_image (content admin)

Αφού εξάγουμε τις παραμέτρους, δηλαδή δύο φωτογραφίες για μία συγκεκριμένη ταινία ορίζουμε τα ονόματα τους στη βάση ως το id της ταινίας ώστε να τις ξεχωρίζουμε, ενώ αποθηκεύουμε το path των φωτογραφιών αλλά και τις ίδιες τις φωτογραφίες ως BLOB (binary large object)

3.2.2 add_movie (content admin)

Αφού εξάγουμε τις παραμέτρους, δηλαδή τα πεδία title, length, category, description, director, premier, συνδεόμαστε στη βάση και τις εισάγουμε σε αυτήν αφού πρώτα προσαρμόζουμε (τροποποιούμε) κατάλληλα το format του πεδίου premier ώστε να εισαχθεί στη βάση ως date.

```
DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("yyyy-MM-dd"); // Adjusted date
format pattern
                        LocalDate premiereDateTime =
LocalDate.parse(PREMIERE, formatter);
```

```
                        Timestamp premiereTimestamp =
Timestamp.valueOf(premiereDateTime.atStartOfDay());
```

3.2.3 add_provoli (content admin)

Αφού εξάγουμε τις παραμέτρους, δηλαδή τα πεδία του form (title, cinema, premier- εφαρμόζουμε το προαναφερθέν format), ελέγχουμε εάν η ημερομηνία της προβολής είναι πριν από αυτήν της πρεμιέρας (αδύνατο) καθώς και εάν δύο προβολές επικαλύπτονται (συμπίπτουν). Στη συνέχεια, συνδεόμαστε στη βάση και εισάγουμε σε αυτήν την προβολή και ανάλογα το cinema που έχουμε επιλέξει ενημερώνεται και ο αριθμός τον διαθέσιμων θέσεων για την προβολή εκείνη. Η ώρα της προβολής ορίζεται ως η τρέχουσα ώρα (που εισαγάγαμε την προβολή), ενώ για την ώρα λήξης προτίθεται η διάρκεια στην ώρα έναρξης. Για τον έλεγχο των επικαλυπτόμενων προβολών κάνουμε select πρώτα από τη βάση τις ήδη υπάρχουσες προβολές για μία δεδομένη ταινία και συγκρίνονται με αυτή που εισάγεται. Παρόμοια τεχνική ακολουθείται και για τον έλεγχο seats.

```
while(rs.next()) {
                          ID = rs.getInt("ID");
                           day = rs.getTimestamp("DAY");
                         filmid = rs.getString("FILMS_ID");
                         len = rs.getInt("LENGTH");;
                                          }
                    existingEndDate = new
Timestamp(day.getTime() + len * 60 * 1000);
                    if ((premiereTimestamp.after(day) &&
premiereTimestamp.before(existingEndDate)) || (
provoliEndDate.after(day)&&
provoliEndDate.before(existingEndDate))) {

                        request.setAttribute("result",
"Screnings overlap. Start time:"+day+" End
time:"+existingEndDate);
                        flag=false;
                         RequestDispatcher rd =
request.getRequestDispatcher("provoles_add.jsp");


                         rd.forward(request,response);
                         return;
```

3.2.4 add_res (content admin)

Αφού εξάγουμε το id της ταινίας (σύμφωνα με την ταινία που επέλεξε ο χρήστης στη jsp σελίδα), παίρνουμε ως όρισμα το seats (τις θέσεις που επέλεξε ο χρήστης για μία προβολή) από

την make_res.jsp (ο κώδικας για επιλογή θέσεων είναι με javascript). Ύστερα, αφού συνδεθούμε στη βάση, εισάγουμε το username του χρήστη που έκλεισε τις συγκεκριμένες θέσεις, ενώ παράλληλα ανανεώνουμε τον διαθέσιμο αριθμό θέσεων για την προβολή εκείνη καθώς και τον πίνακα RESERVATIONS.

```
char[] charArray = totals.toCharArray();
                    int temp;
                   for (String seat : seats) {

                       System.out.println("Selected seat: " +
seat);



                        query = "INSERT INTO
cinema_last.reservations ( username,PROVOLES_ID,SEAT ) VALUES
(?, ?, ?)";
                        statement =
conn.prepareStatement(query);

                        statement.setString(1,
String.valueOf(session.getAttribute("username")));
                        statement.setInt(2, provoli_id);
                        statement.setInt(3,
Integer.parseInt(seat));

                         statement.executeUpdate();

                         seatn+=1;

                         charArray[Integer.parseInt(seat)-1] =
'1';



                   }
```

3.2.5 add_user (admin)

Αφού εξάγουμε τις παραμέτρους του form (username, password, rpassword, create, type), ελέγχουμε αν type=2 (δηλαδή client) ώστε να εξάγουμε και το πεδίο date. Μετά ελέγχουμε αν password=rpassword, καθώς και επίσης εάν το username έχει ήδη παρθεί από άλλον χρήστη, αφού πρώτα συνδεθήκαμε στη βάση. Ύστερα πραγματοποιούμε το hashing του password με τον αλγόριθμο SHA-256, ο οποίος αναπαριστά σε με ένα array byte προς byte το password, το κατακερματίζει και στη συνέχεια το μετατρέπει σε 16δικό.

```
try {
```

```
                            MessageDigest md =
MessageDigest.getInstance("SHA-256");
                        byte[] hashedBytes =
md.digest(password.getBytes());
                        StringBuilder sb = new
StringBuilder();
                        for (byte b : hashedBytes) {
                            sb.append(String.format("%02x",
b));
                        }
                        hashedpass = sb.toString();

System.out.println(hashedpass.length());
                    }catch (Exception e){
                        System.out.println("nope");
                        System.out.println(e.toString());
                        return;
```

Ύστερα, εισάγουμε τον χρήστη μαζί με όλα του τα χαρακτηριστικά στη βάση. Σημειώνεται πως η παραπάνω σελίδα αφορά τους admin

3.2.6 delete_movie (content admin)

Εξάγουμε το id της ταινίας που επιλέχθηκε στη σελίδα main.jsp (ως content admin) και στη συνέχεια επιλέγουμε τη συγκεκριμένη ταινία από τη βάση και την διαγράφουμε μαζί με τα σχετικά rows από του πίνακες PROVOLES και RESERVATIONS. Μετά την διαγραφή ο χρήστης ανακατευθύνεται στη σελίδα main.jsp .

```
response.sendRedirect("main.jsp");
```

3.2.7 delete_provoli (content admin)

Ομοίως με delete_movie

3.2.8 delete_user (admin)

Ομοίως με delete_movie

3.2.9 edit_movie (content admin)

Εξάγουμε τις παραμέτρους της ταινίας που θέλουμε ένα τροποποιήσουμε (τα ενημερωμένα πεδία) και αφού συνδεθούμε στη βάση κάνουμε UPDATE τα στοιχεία της συγκεκριμένης ταινίας.

3.2.10 edit_provoli (content admin)

Ομοίως με edit_movie

3.2.11 edit_user (admin)

Ομοίως με edit_movie

3.2.12 reserve (client)

Ανακατευθύνει τον χρήστη στην σελίδα make_res.jsp

3.2.13 transfer_movie_data (content admin)

Χρησιμοποιούμε μια ιδιαίτερη τεχνική για να αναγνωρίζουμε αν ο χρήστης θέλει να τροποποιήσει την ταινία ή να εισάγει μία νέα. Αυτό γίνεται με τη χρήση αρνητικού id, δηλαδή τη στιγμή που πατηθεί το + (δηλαδή προσθήκη ταινίας στην main.jsp του content admin), αφού περνιέται ως παράμετρο το αρνητικό id (δηλαδή δεν υπάρχει η ταινία ώστε να ανακατευθύνει τον χρήστη στη σελίδα movie_add.jsp). Ειδάλλως, αν id>0 τότε ανακατευθύνεται στη σελίδα edit_movie.jsp με παράμετρο το id της ταινίας.

```
int ID = Integer.parseInt(request.getParameter("id"));
          HttpSession session = request.getSession();


          if(session.getAttribute("username")==null){
               response.sendRedirect("log-in.jsp");
               return;
          }

          if(ID<0) {
                response.sendRedirect("movie_add.jsp");
          }else {
               String typestring =
session.getAttribute("type").toString();
               int type_int = Integer.valueOf(typestring);
               if(type_int==1) {
                    request.setAttribute("movie_id",
Integer.toString(ID));

     request.getRequestDispatcher("edit_movie.jsp").forward(re
quest, response);

               }
          }
```

3.2.14 transfer_provoli_data (content admin)

Ομοίως με tranfer_movie_data

3.2.15 transfer_user_data (admin)

Ομοίως με tranfer_movie_data

3.2.16 Login (everyone)

Αφού εξάγουμε τα πεδία που συμπλήρωσε ο χρήστης (username, password), συνδεόμαστε στη βάση και ελέγχουμε αρχικά αν υπάρχει το δεδομένο username και ύστερα (εφόσον υπάρχει

το username) μετά το hashing του password που έδωσε ο χρήστης στο συμπλήρωμα του form, ελέγχουμε αν αυτό αντιστοιχεί με το hashed password που υπάρχει ήδη στη βάση (από το register). Στη συνέχεια δημιουργούμε ένα session για αυτόν τον χρήστη.

```
HttpSession session = request.getSession();

session.setAttribute("username",username);

session.setAttribute("type",type);
```

Μετέπειτα, ανάλογα με τον τύπο του χρήστη που συνδέεται, ανακατευθύνεται στην αντίστοιχη jsp σελίδα.

```
if(type==1) {

    response.sendRedirect("contentadmin/main.jsp");
                                        flag=false;
                                    }else if(type==2){

    response.sendRedirect("client/main.jsp");
                                        flag=false;
                                    }else if(type==0){

    response.sendRedirect("admin/main.jsp");
                                        flag=false;
                                    }
```

3.2.17 Logout (everyone)

Όταν ο χρήστης αιτηθεί να αποσυνδεθεί από τον λογαριασμό του (πατώντας το εικονίδιο της εξόδου), αφαιρούνται από το session τα attributes του (name, type) και το session προσδιορίζεται ως invalid. Τέλος, ο χρήστης ανακατευθύνεται στο log-in.jsp .

3.2.18 register (clients)

Αφορά την εγγραφή στη βάση των clients και λειτουργεί όμοια με το add_user.

4 JSP

Σε αυτήν την ενότητα απλά θα παραθέσουμε τη λειτουργία των jsp σελίδων (τι υλοποιεί και που χρησιμεύει η κάθε μία χωρίς να αναλύσουμε ιδιαίτερα, πέρα από κάποια βασικά στοιχεία, αφού αποτελούνται κατά κύριο λόγο από javascript και html). Σημειώνουμε πως δίπλα από κάθε servlet αναγράφουμε εντός παρενθέσεων τον τύπο χρηστών που αφορούν.

4.1 Γενικά στοιχεία

Σε κάθε jsp σελίδα γίνεται ο έλεγχος αν ο χρήστης στο session είναι null δηλαδή αν υπάρχει κάποιος συνδεδεμένος χρήστης εκείνη τη στιγμή (και αν όχι τον ανακατευθύνει στο log-in.jsp). Αν ο έλεγχος αποτύχει, δηλαδή έχει συνδεθεί κάποιος χρήστης, τότε εξάγουμε το type του. Επιπλέον, ορίζουμε την HTTP κεφαλίδα Cache Control στο response που στέλνεται από έναν διακομιστή σε έναν πελάτη. Αναλυτικότερα:

Η HTTP κεφαλίδα" Cache-Control" ελέγχει τη συμπεριφορά της αποθήκευσης προσωρινής μνήμης (cache) στον περιηγητή του πελάτη. Η εντολή response.setHeader("Cache-Control","no-cache,no-store,must-revalidate"); περιλαμβάνει τρεις κατευθυντήριες εντολές για τον περιηγητή:

no-cache: Ο περιηγητής δεν πρέπει να χρησιμοποιήσει την προσωρινή μνήμη για την αποθήκευση της απόκρισης. Κάθε φορά που ο χρήστης ζητά την ίδια πόρτα, ο περιηγητής θα πρέπει να ζητάει από τον διακομιστή να την ανακτήσει εκ νέου.

no-store: Ο περιηγητής δεν πρέπει να αποθηκεύει την απόκριση στην προσωρινή μνήμη καθόλου. Αυτό αποτρέπει τον περιηγητή από το να κρατήσει οποιαδήποτε μορφή της απόκρισης, ακόμη και για την τρέχουσα συνεδρία.

must-revalidate: Αν η αποθηκευμένη απόκριση έχει λήξει, ο περιηγητής πρέπει να ζητήσει επιβεβαίωση από τον διακομιστή πριν τη χρήση της. Αυτό διασφαλίζει ότι ο περιηγητής θα έχει πάντα την πιο ενημερωμένη έκδοση της απόκρισης.

```
if(session.getAttribute("username")==null){
     response.sendRedirect("../log-in.jsp");
}else{
     response.setHeader("Cache-Control","no-cache,no-
store,must-revalidate");
     String typestring =
session.getAttribute("type").toString();
     int type_int = Integer.valueOf(typestring);
     if( type_int!=2 ){
          response.sendRedirect("../log-in.jsp");
     }
}
```

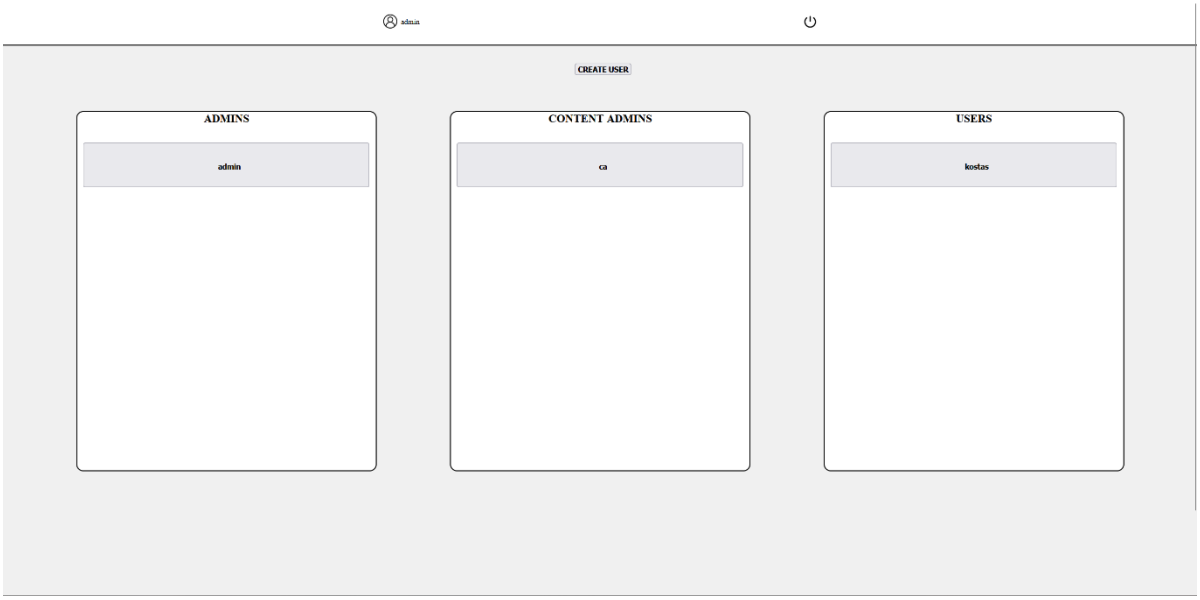
4.2 Περιγραφή JSP

4.2.1 edit_user (admin)

Η σελίδα με την φόρμα που αφορά την τροποποίηση χρηστών. Εξάγουμε τα στοιχεία της φόρμας και τα εισάγουμε στη βάση εφόσον αυτά είναι έγκυρα. Τα errors ανακατευθύνουν στην σελίδα error.jsp .

4.2.2 main (admin)

Η κεντρική σελίδα του admin στην οποία εμφανίζονται 3 πίνακες, οι οποίοι περιέχουν τους χρήστες (κάθε πίνακας περιέχει το σύνολο των αντίστοιχων χρηστών), από τον οποίο μπορεί να επιλέξει έναν οποιοδήποτε χρήστη και να τον τροποποιήσει, αλλά και να δημιουργήσει έναν νέο χρήστη (ανακατευθύνεται στο servlet transfer_user_data).



4.2.3 user_add (admin)

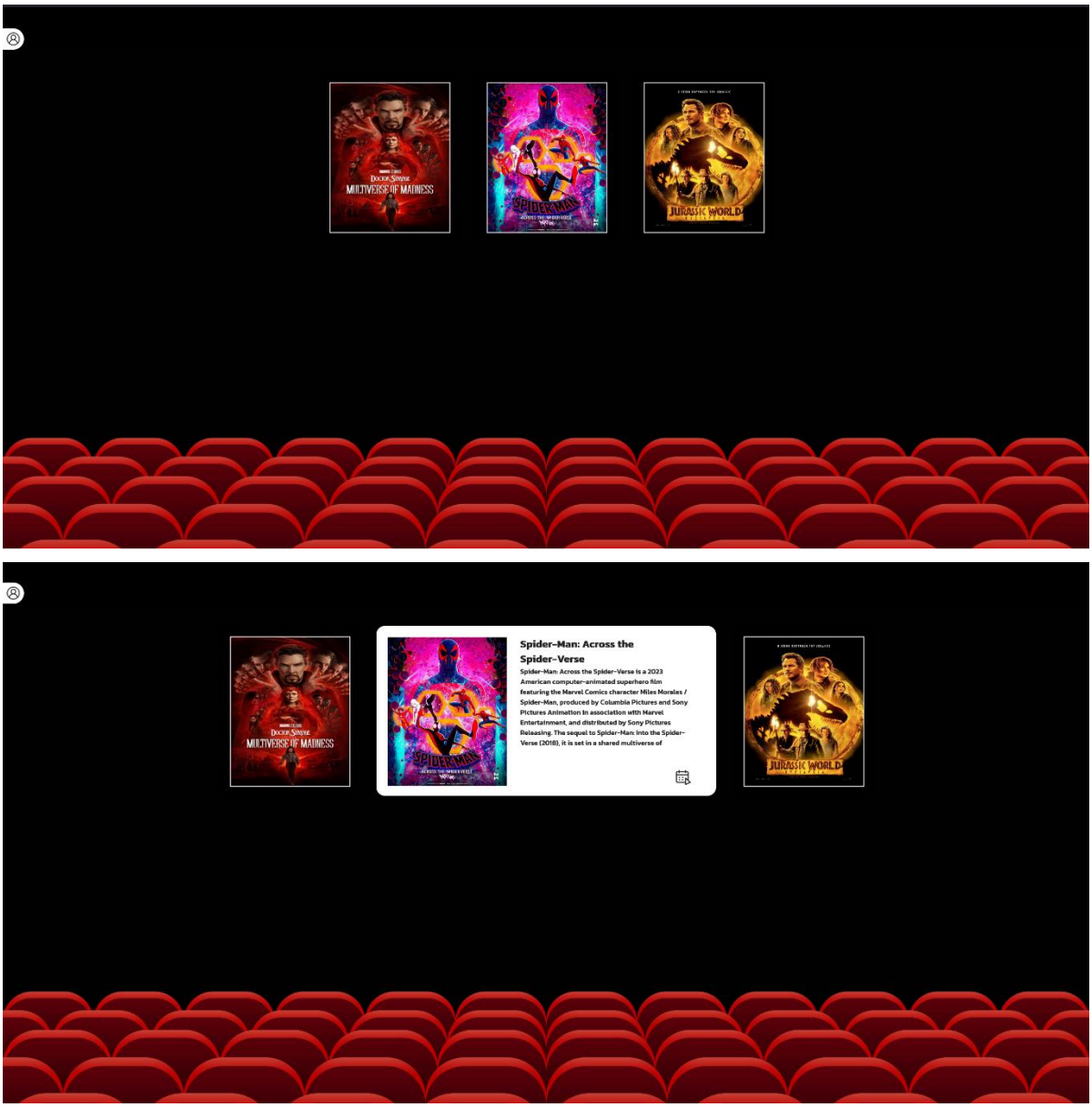
Στη σελίδα αυτή περιέχεται η φόρμα για την προθήκη ενός χρήστη από τον admin.

4.2.4 fail_res (client)

Αυτή είναι η σελίδα στην οποία ανακατευθύνεται ο client εάν αποτύχει η κράτησή του (δεν υλοποιηθεί και η ανακατεύθυνση γίνεται από το make_res.jsp).

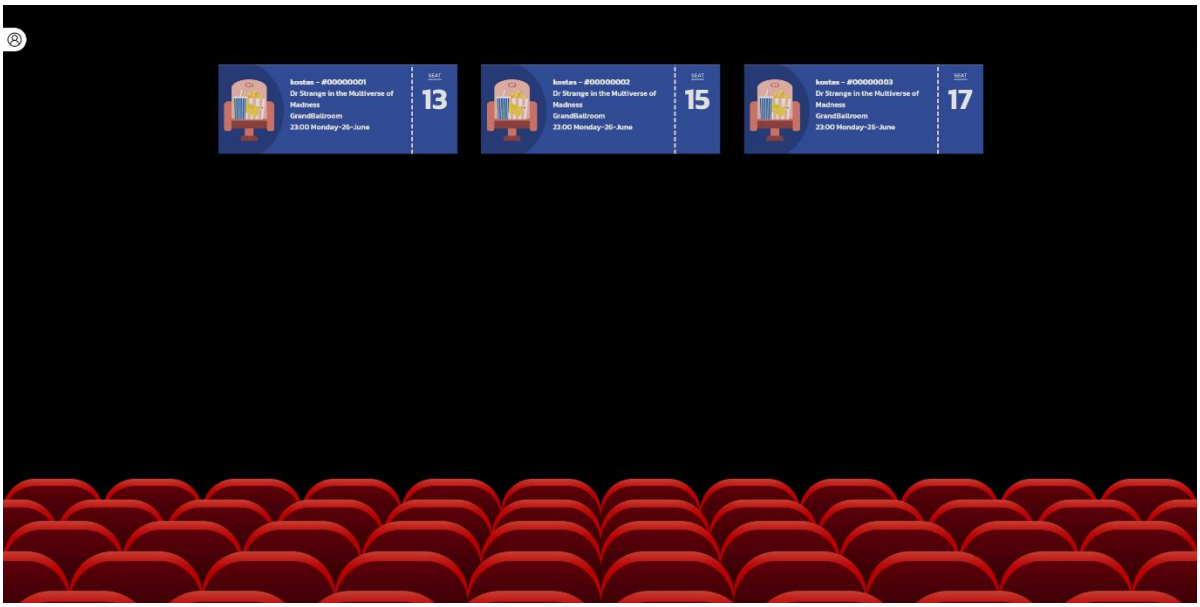
4.2.5 main (client)

Η κεντρική σελίδα του client (ανακατευθύνεται από το log-in.jsp εάν type=2 και από οποιαδήποτε σελίδα εάν πατήσει το αντίστοιχο εικονίδιο) στην οποία ο χρήστης έχει την δυνατότητα να επιλέξει τις ταινίες για τις οποίες επιθυμεί να δει λεπτομέρειες και να κάνει μία κράτηση στη συνέχεια.



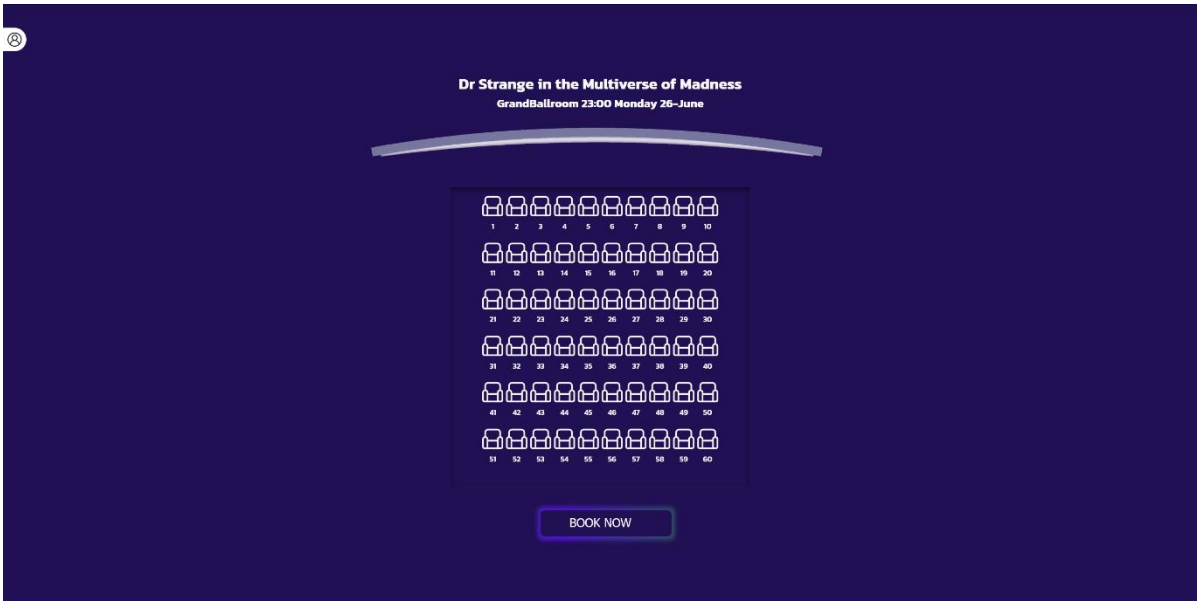
4.2.6 maint (client)

Σε αυτήν την σελίδα ο χρήστης έχει την δυνατότητα να δει τα εισιτήρια που έκλεισε (ανακατευθύνεται εκεί από οποιαδήποτε σελίδα εάν πατήσει το αντίστοιχο εικονίδιο).



4.2.7 make_res (client)

Σε αυτήν την σελίδα εμφανίζονται στον χρήστη οι διαθέσιμες θέσεις για την προβολή της ταινίας που επέλεξε (ανακατευθύνεται εκεί από το view_movie.jsp).

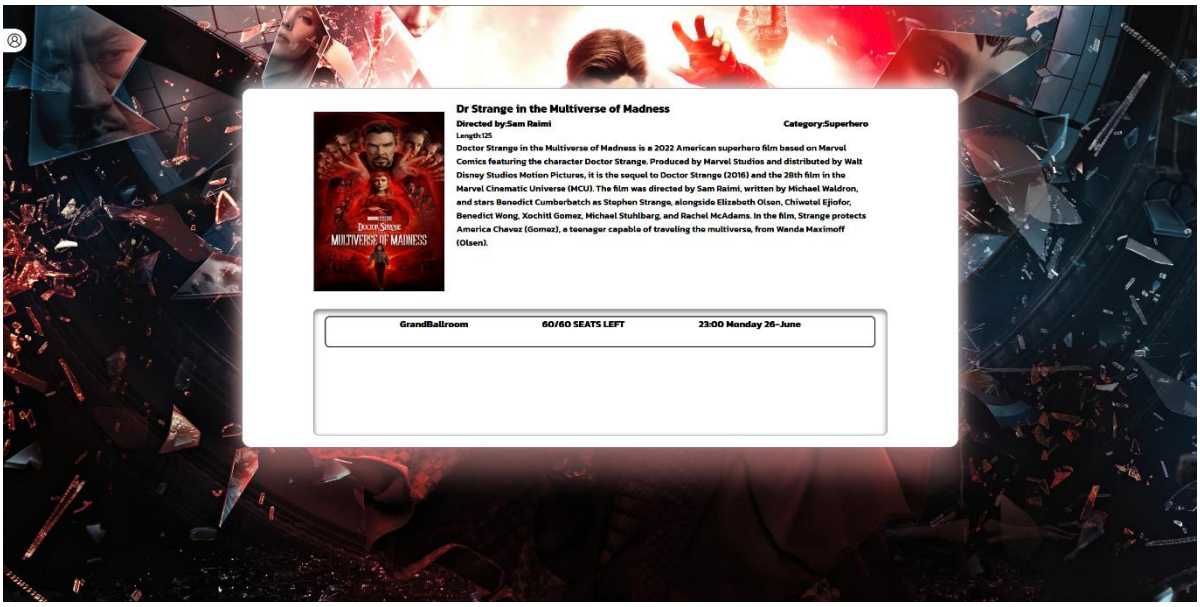


4.2.8 succ_res (client)

Αυτή είναι η σελίδα που εμφανίζεται στον χρήστη εφόσον η κράτηση θέσεων που πραγματοποίησε ήταν επιτυχής (ανακατευθύνεται εδώ από το make_res.jp).

4.2.9 view_movie (client)

Σε αυτήν την σελίδα ο χρήστης μπορεί να τις λεπτομέρειες τις ταινίας που επέλεξε καθώς και τις διαθέσιμες προβολές (ανακατευθύνεται εκεί από το main.jsp).



4.2.10 edit_movie (content admin)

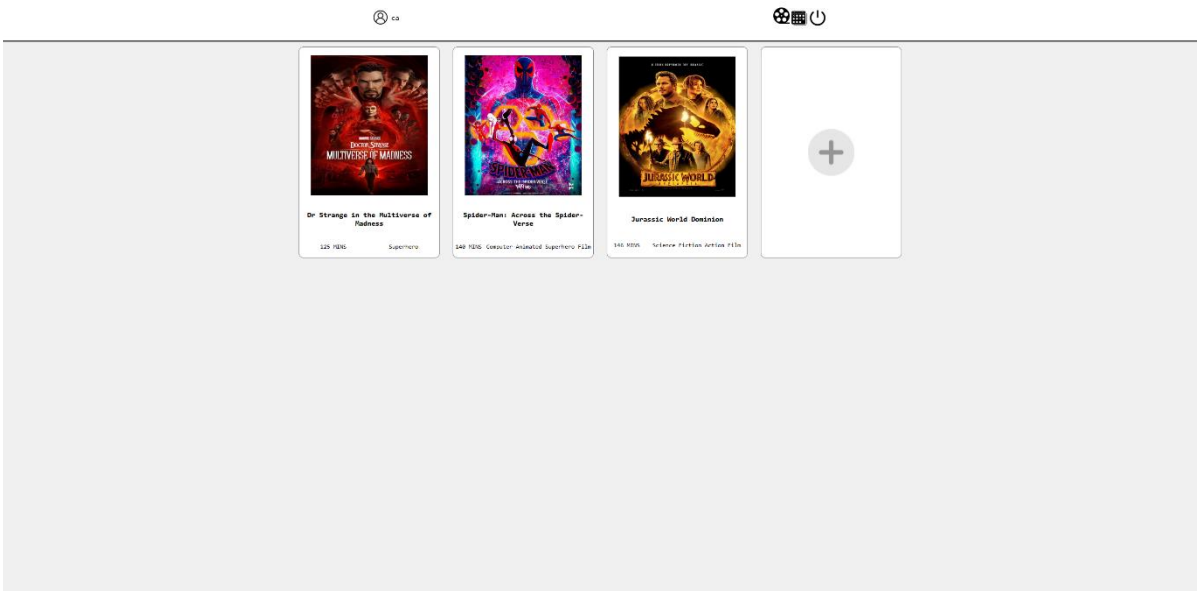
Σε αυτήν την σελίδα ο content admin μπορεί να τροποποιήσει τα στοιχεία μιας ταινίας που επέλεξε (ανακατευθύνεται εκεί από το main.jsp και type =1 και από οπουδήποτε εφόσον πατήσει το αντίστοιχο εικονίδιο).

4.2.11 edit_provoli (content admin)

Σε αυτήν την σελίδα ο content admin μπορεί να τροποποιήσει τα στοιχεία μιας προβολής που επέλεξε (ανακατευθύνεται εκεί από το mainp.jsp και type =1 και από οπουδήποτε εφόσον πατήσει το αντίστοιχο εικονίδιο).

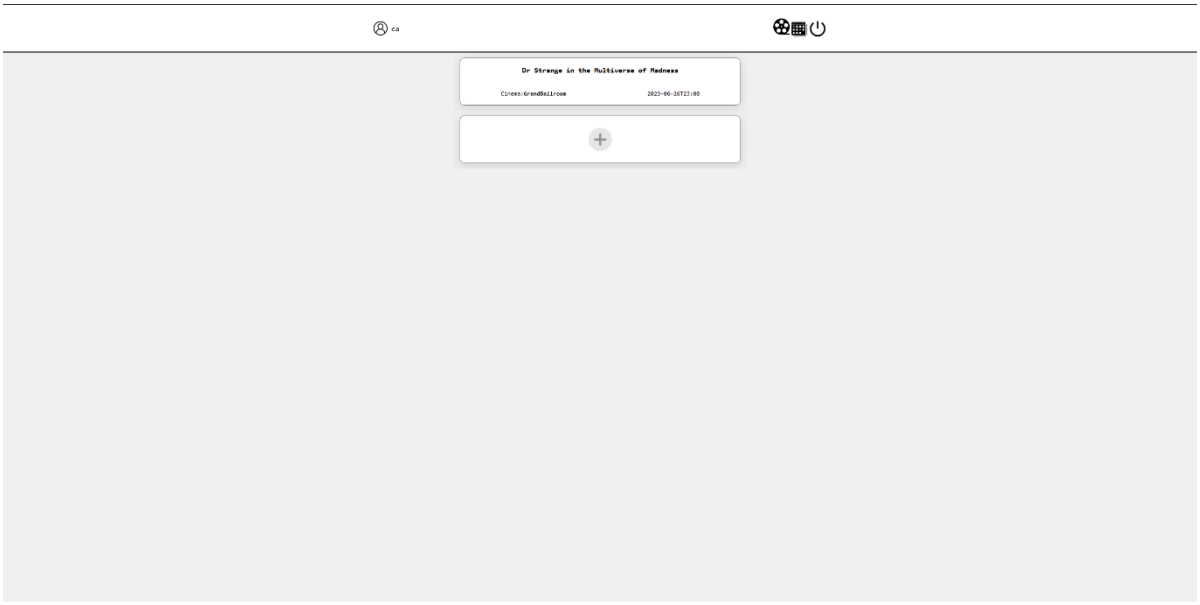
4.2.12 main (content admin)

Η κεντρική σελίδα του content admin από την οποία μπορεί να επιλέξει κάποια ταινία για να τροποποιήσει (edit_movie.jsp) ή να εισάγει (movie_add.jsp), καθώς και να μεταφερθεί και στην main.jsp (ανακατευθύνεται εδώ από log-in.jsp εφόσον type=1).



4.2.13 mainp (content admin)

Σε αυτήν την σελίδα ο content admin μπορεί να τροποποιήσει δει τις προβολές που επέλεξε (ανακατευθύνεται εκεί εφόσον πατήσει το αντίστοιχο εικονίδιο).



4.2.14 movie_add (content admin)

Σε αυτήν την σελίδα ο content admin μπορεί να εισάγει μια ταινία (ανακατευθύνεται εκεί από το mainp.jsp εφόσον πατήσει το αντίστοιχο εικονίδιο).

4.2.15 provoles_add (content admin)

Σε αυτήν την σελίδα ο content admin μπορεί να εισάγει τα μια νέα προβολή (ανακατευθύνεται εκεί από το mainp.jsp εφόσον πατήσει το αντίστοιχο εικονίδιο).

4.2.16 log_in (everyone)

Σε αυτήν την σελίδα οι χρήστες εισάγουν τα credentials ώστε να συνδεθούν.

4.2.17 register (client)

Σε αυτήν την σελίδα οι χρήστες (clients) μπορούν να εγγραφούν.

4.2.18 succ_movie (content_admin)

Η σελίδα υπονοεί επιτυχής προσθήκη ταινίας

4.2.19 succloggin (everyone)

Λειτουργεί ως μεσάζοντας στην ανακατεύθυνση των χρηστών στην κατάλληλη σελίδα ανάλογα το type.

5 CSS

Σε αυτήν την ενότητα απλώς θα παραθέσουμε τα αντίστοιχα αρχεία .css που χρησιμοποιήθηκαν.

5.1.1 addMovie

5.1.2 addProvoli

5.1.3 admin_style

5.1.4 client_style

5.1.5 contentadmin_style

5.1.6 contentadmin_stylep

5.1.7 index_style

5.1.8 reservation_style

5.1.9 view_movie_syle

ΒΙΒΛΙΟΓΡΑΦΙΑ

Για την υλοποίηση της εργασίας χρησιμοποιήσαμε τις διαφάνειες του μαθήματος και για το debugging διάφορα forums αλλά και βίντεο στο youtube.