ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής

Εργασία Μαθήματος «Προγραμματισμός στο διαδίκτυο και στον παγκόσμιο ιστό»

# Τίτλος Εργασίας:
# «Εφαρμογή διαχείρισης ραντεβού τακτικών ιατρικών εξετάσεων»

| | |
|---|---|
| Άσκηση 02 | ***Δημιουργία Βάσης Δεδομένων, Web Project και Υλοποίηση λειτουργιών Ασθενή*** |
| Όνομα φοιτητή – Αρ. Μητρώου | Κωνσταντίνος Αργυρίου – Π19017 |
| | Ιωακείμ Ελ-Χαττάμπ Μπριστογιάννης – Π19048 |
| | Σοφία Μαρκοπούλου – Π19097 |
| | Μυρτώ-Μαρία Σπάθα – Π19156 |
| Ημερομηνία παράδοσης | 19/06/2021 |

## ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ



## 1. Εκφώνηση της άσκησης

ΑΣΚΗΣΗ-2-2021.pdf

## 2. Γενική Περιγραφή της λύσης

Σκοπός της γενικής εργασίας ήταν η ανάπτυξη μιας εφαρμογής η οποία θα δίνει τη δυνατότητα σε ασθενείς και γιατρούς να προγραμματίζουν ραντεβού μεταξύ τους με τη συμβολή κάποιων διαχειριστών οι οποίοι εισάγουν τους γιατρούς στο σύστημα. Πιο συγκεκριμένα, σκοπός της 2ης εργασίας, η οποία αποτελεί και το αντικείμενο της συγκεκριμένης παρουσίασης, ήταν να δημιουργηθεί η βάση δεδομένων της εφαρμογής, να δημιουργηθεί το web project και να υλοποιηθούν κάποιες λειτουργίες του Ασθενή. Συνοπτικά, δημιουργήθηκε η βάση δεδομένων DBKotza, τροποποιήθηκαν οι κλάσεις User, Patient, Appointment, Doctor, Admin, διεγράφη η κλάση CreateUsers και δημιουργήθηκαν πακέτα κλάσεων που υλοποιούν κάποιες λειτουργίες. Αναλυτικά, για τη διαχείριση της βάσης δεδομένων δημιουργήθηκε το πακέτο java->com->database που περιέχει τις κλάσεις Database, QueryManager. Για την κρυπτογράφηση των κωδικών των χρηστών δημιουργήθηκε το πακέτο κλάσεων java->com->core->security που περιέχει την κλάση SecurityManager. Δημιουργήθηκε το πακέτο java->com->web.servlets που περιέχει τα servlet PatientLogin, PatientServlet. Τέλος, δημιουργήθηκαν οι jsp σελίδες patientHome, admin, doctor, patientlogin, index, καθώς και το αρχείο web.xml.

Προτού αναπτυχθούν οι λεπτομέρειες του κώδικα της εφαρμογής, ακολουθεί μια συνοπτική επεξήγηση του κώδικα ώστε να διευκολυνθεί η περιήγηση σε αυτόν αλλά και η κατανόηση των λεπτομερειών. Κατά την εκκίνηση της εφαρμογής, ο επισκέπτης εισάγεται στην αρχική σελίδα (index.jsp) μέσα στην οποία υπάρχουν οι επιλογές *Σύνδεση ως Ασθενής, Σύνδεση ως Γιατρός, Σύνδεση ως Διαχειριστής*. Όταν ο χρήστης πατήσει πάνω στην 1η επιλογή (οι άλλες δύο θα είναι διαθέσιμες μετά την εκπόνηση της 3ης εργασίας) εισάγεται στην σελίδα σύνδεσης ασθενή (patientLogin.jsp, συνδέεται με το PatientLogin servlet), όπου καλείται να πληκτρολογήσει το όνομα χρήστη και τον κωδικό πρόσβασής του στην πλατφόρμα. Σε περίπτωση επιτυχούς σύνδεσης, ο ασθενής εισάγεται στη σελίδα patientHome.jsp, όπου εμφανίζονται τα στοιχεία του και τα προηγούμενα ραντεβού του (μέσω της μεθόδου showPreviousAppointments). Σε αντίθετη περίπτωση, ο ασθενής παραμένει στην τρέχουσα σελίδα, η οποία είναι κενή περιεχομένου.

# 3. Κώδικας προγράμματος

Ακολουθεί η αναλυτική περιγραφή των προσθηκών του προγράμματος. Για να δείτε την παρουσίαση της 1ης εργασίας, στην οποία στηρίχτηκε η οικοδόμηση της 2ης, πατήστε εδώ.

## 3.1 Τροποποίηση κώδικα της κλάσης User (περιγράφονται μόνο οι αλλαγές)

Η κλάση **User** περιέχει επιπλέον των χαρακτηριστικών που έχουν αναλυθεί στην εργασία 1 τα εξής στοιχεία:

- Τον default constructor ο οποίος είναι άδειος και χρησιμεύει για να δημιουργείται ένας προσωρινός εγγεγραμμένος χρήστης -αυτός που συνδέεται στην εφαρμογή),
- Μία μέθοδο getUserDetails η οποία παίρνει ως παραμέτρους το username και του εκάστοτε χρήστη και το αναγνωριστικό για το είδος του table (η μεταβλητή table παίρνει τις τιμές patient, doctor, admin) και επιστρέφει έναν πίνακα κατακερματισμού με τα στοιχεία του.

  Σημείωση: η λειτουργία της μεθόδου θα αναλυθεί περαιτέρω κατά την περιγραφή των περιεχομένων των κλάσεων Patient, Doctor, Admin, όπου και γίνεται υποσκέλισή της(Override)).
- Την τροποποιημένη συνάρτηση login η οποία παίρνει ως παραμέτρους το username,password και το αναγνωριστικό για το είδος του χρήστη table και εισάγει σε έναν πίνακα κατακερματισμού dbCredentials τα username, password του χρήστη τα οποία ανακτά από τη βάση αναζητώντας στον πίνακα table το username που έχει δοθεί ως παράμετρος. Στη συνέχεια, γίνεται έλεγχος αν τα στοιχεία που ανακτήθηκαν από τη βάση είναι τα ίδια με τα στοιχεία που πληκτρολόγησε ο χρήστης (και έχουν δοθεί ως παράμετροι στη συνάρτηση). Σε αντίθετη περίπτωση, γίνεται χρήση της εξαίρεσης LoginFailure.

  Σημείωση: Μέσα στη συνάρτηση login εκτελείται η μέθοδος getFromDatabase της κλάσης QueryManager (βλ. 3.8).
- Την τροποποιημένη συνάρτηση logout η οποία παίρνει ως παράμετρο την τρέχουσα σύνοδο session και την ακυρώνει (invalidate), αφού πρώτα έχει ελέγξει αν η τιμή που έχει αποθηκευτεί στο session έχει συσχετιστεί με το username (μέσω της μεθόδου getAttribute).

## 3.2 Τροποποίηση κώδικα της κλάσης Doctor (περιγράφονται μόνο οι αλλαγές)

Η κλάση **Doctor** περιέχει επιπλέον των χαρακτηριστικών που έχουν αναλυθεί στην εργασία 1 τα εξής στοιχεία:

- Τον default constructor ο οποίος καλεί τον default constructor της υπερκλάσης User μέσω του τελεστή super ώστε να δημιουργήσει αντικείμενο του γιατρού που συνδέεται στην εφαρμογή.

- Τη μέθοδο getUserDetails (υποσκελίζει την συνώνυμη μέθοδο της υπερκλάσης User) η οποία επιστρέφει έναν πίνακα κατακερματισμού στον οποίο εισάγει τα στοιχεία doctorAMKA, username, specialty, surname, name ανακτώντας τα από τη βάση αναζητώντας στον πίνακα table το username που έχει δοθεί ως παράμετρος.

  Σημείωση: Μέσα στη μέθοδο getUserDetails εκτελείται η συνάρτηση getFromDatabase της κλάσης QueryManager (βλ. 3.8).

### 3.3 Τροποποίηση κώδικα της κλάσης Admin (περιγράφονται μόνο οι αλλαγές)

Η κλάση **Admin** περιέχει επιπλέον των χαρακτηριστικών που έχουν αναλυθεί στην εργασία 1 τη μέθοδο getUserDetails (υποσκελίζει την συνώνυμη μέθοδο της υπερκλάσης User) η οποία επιστρέφει έναν πίνακα κατακερματισμού στον οποίο εισάγει το username ανακτώντας το από τη βάση αναζητώντας στον πίνακα table το username που έχει δοθεί ως παράμετρος.

Σημείωση: Μέσα στη μέθοδο getUserDetails εκτελείται η συνάρτηση getFromDatabase της κλάσης QueryManager (βλ. 3.8).

### 3.4 Τροποποίηση κώδικα της κλάσης Patient (περιγράφονται μόνο οι αλλαγές)

Η κλάση **Patient** περιέχει επιπλέον των χαρακτηριστικών που έχουν αναλυθεί στην εργασία 1 τα εξής στοιχεία:

- Τον default constructor ο οποίος καλεί τον default constructor της υπερκλάσης User μέσω του τελεστή super ώστε να δημιουργήσει αντικείμενο του ασθενή που συνδέεται στην εφαρμογή.
- Τη μέθοδο getUserDetails (υποσκελίζει την συνώνυμη μέθοδο της υπερκλάσης User) η οποία επιστρέφει έναν πίνακα κατακερματισμού στον οποίο εισάγει τα στοιχεία patientAMKA, username, name, surname ανακτώντας τα από τη βάση αναζητώντας στον πίνακα table το username που έχει δοθεί ως παράμετρος.

  Σημείωση: Μέσα στη μέθοδο getUserDetails εκτελείται η συνάρτηση getFromDatabase της κλάσης QueryManager (βλ. 3.8).

Επίσης, έχει αφαιρεθεί η συνάρτηση showAppointmentsHistory η λειτουργία της οποίας εκτελείται μέσα στη μέθοδο showPreviousAppointments που περιέχεται στην κλάση Appointment (βλ. 3.5) και καλείται από την ιστοσελίδα patientHome.jsp.

### 3.5 Τροποποίηση κώδικα της κλάσης Appointment (περιγράφονται μόνο οι αλλαγές)

Η κλάση **Appointment** περιέχει επιπλέον των χαρακτηριστικών που έχουν αναλυθεί στην εργασία 1 τα εξής στοιχεία:

- Την τροποποιημένη μέθοδο showPreviousAppointments (στην $1^{η}$ εργασία αναφέρεται ως showPatientHistory) η οποία παίρνει ως παραμέτρους το username

του χρήστη του οποίου το ιστορικό των ραντεβού θα εμφανίσει (δηλαδή του χρήστη που είναι συνδεδεμένος στην εφαρμογή) και έναν JspWriter out μέσω του οποίου εγγράφονται δεδομένα κειμένου στην ιστοσελίδα patientHome.jsp, όπου καλείται η μέθοδος, και τυπώνει έναν πίνακα με τα προηγούμενα ραντεβού του ασθενή.

Σημείωση: Μέσα στη μέθοδο showPreviousAppointments εκτελείται η συνάρτηση getPreviousAppointments της κλάσης QueryManager.

### 3.6 Κώδικας της κλάσης SecurityManager

Η κλάση **SecurityManager** είναι η κλάση η οποία κρυπτογραφεί τους κωδικούς των χρηστών. Πιο συγκεκριμένα, περιέχει τη μέθοδο getHash η οποία παίρνει ως παράμετρο τον κωδικό που έχει εισάγει ο χρήστης στην ιστοσελίδα σύνδεσης και τον μετατρέπει σε bytes τα οποία στη συνέχεια αναπαρίστανται σε δεκαεξαδικό σύστημα (γίνεται χρήση του αλγορίθμου md5).

Σημείωση: Η κλήση της μεθόδου θα γίνει στην 3η εργασία.

### 3.7 Κώδικας της κλάσης Database

Η κλάση **Database** είναι απαραίτητη ώστε να συνδεθεί η εφαρμογή με τη βάση δεδομένων. Πιο συγκεκριμένα, περιέχει τη μέθοδο getConnection η οποία αναζητά το Data Source της βάσης, δημιουργεί μια σύνδεση με αυτό και επιστρέφει αυτή τη σύνδεση.

### 3.8 Κώδικας της κλάσης QueryManager

Η κλάση **QueryManager** χρησιμεύει ώστε να εκτελούνται ερωτήματα ανάκτησης ή εγγραφής δεδομένων στη βάση και να παρουσιάζονται τα αποτελέσματά τους σε μορφή που να μπορούν να χρησιμοποιηθούν από την εφαρμογή. Αναλυτικά, περιέχει τις παρακάτω μεθόδους:

- Την μέθοδο queryExecutor η οποία παίρνει ως παραμέτρους το ερώτημα που θα εκτελεστεί στη βάση, τη σύνδεση με το Data Source, μία λογική μεταβλητή isWrite (true για εγγραφή δεδομένων στη βάση, false για ανάκτηση δεδομένων από τη βάση) και τις παραμέτρους που θα τοποθετηθούν στο ερώτημα (αντιπροσωπεύουν στήλες πινάκων). Σε περίπτωση που η isWrite είναι false εκτελεί το ερώτημα που τέθηκε επιστρέφοντας το σύνολο των αποτελεσμάτων (τα δεδομένα που ανακτήθηκαν από τη βάση). Σε αντίθετη περίπτωση, εισάγει τα δεδομένα στη βάση (εκκρεμεί). Η μέθοδος εκτελείται εσωτερικά κατά την κλήση των μεθόδων getFromDatabase, saveToDatabase (αναλύονται παρακάτω).

  Σημείωση: Η εγγραφή δεδομένων στη βάση θα υλοποιηθεί στην 3η εργασία.

- Τη μέθοδο retrieveData η οποία παίρνει ως παραμέτρους το αποτέλεσμα που έχει εξαχθεί από τη βάση και τα ονόματα των πεδίων που αντιστοιχούν στα ανακτημένα

δεδομένα και τα εισάγει σε έναν πίνακα κατακερματισμού (για παράδειγμα ένα στοιχείο του πίνακα θα μπορούσε να είναι [username, peter12]). Η μέθοδος εκτελείται εσωτερικά κατά την κλήση της μεθόδου getFromDatabase (αναλύεται παρακάτω).

- Τη μέθοδο getFromDatabase η οποία παίρνει ως παραμέτρους το username του συνδεδεμένου χρήστη, το ερώτημα που θα εκτελεστεί στη βάση, τη σύνδεση με το Data Source, τον πίνακα από τον οποίο θα ανακτήσει δεδομένα και τις παραμέτρους που θα τοποθετηθούν στο ερώτημα (αντιπροσωπεύουν στήλες πινάκων). Στο εσωτερικό της μεθόδου γίνεται κλήση των συναρτήσεων queryExecutor και retrieveData και επιστρέφεται ο πίνακας κατακερματισμού που επιστράφηκε κατά την κλήση της τελευταίας συνάρτησης. Η μέθοδος εκτελείται εσωτερικά κατά την κλήση της μεθόδου getUserDetails των κλάσεων Patient, Admin και Doctor και της μεθόδου login της κλάσης User.

- Τη μέθοδο saveToDatabase η οποία παίρνει ως παραμέτρους το ερώτημα που θα εκτελεστεί στη βάση, τη σύνδεση με το Data Source, τον πίνακα στον οποίο θα εισαχθούν τα δεδομένα και τα πεδία που θα τοποθετηθούν στο ερώτημα (αντιπροσωπεύουν στήλες πινάκων) και εκτελεί τη συνάρτηση queryExecutor ώστε να εισάγει δεδομένα στη βάση.

  Σημείωση: Η εισαγωγή δεδομένων στη βάση θα υλοποιηθεί στην 3η εργασία.

- Τη μέθοδο currentDate η οποία παίρνει ως παράμετρο τη σημερινή ημερομηνία και τη μετατρέπει σε μορφή συμβατή με την sql.

- Τη μέθοδο getPreviousAppointments η οποία παίρνει ως παραμέτρους το username του συνδεδεμένου ασθενή, τη σύνδεση με το Data Source και το ερώτημα που θα εκτελεστεί στη βάση και επιστρέφει τα προηγούμενα ραντεβού του ασθενή.

Σημείωση: Τα ερωτήματα (queries) που θα εκτελεστούν τη βάση έχουν συνταχθεί στο enum Queries (βλ. 3.9).

### 3.9 Κώδικας του enum Queries

Το enum **Queries** περιέχει τα παρακάτω queries:

- RETRIEVE_CREDENTIALS("SELECT username,password FROM {0} WHERE username = ?") (Χρησιμοποιείται ως παράμετρος στη μέθοδο getFromDatabase η οποία καλείται στη μέθοδο login)
- RETRIEVE_DETAILS("SELECT * FROM {0} WHERE username = ?") (Χρησιμοποιείται ως παράμετρος στη μέθοδο getFromDatabase η οποία καλείται στη μέθοδο getUserDetails)
- PREVIOUS_APPOINTMENTS("select date, time, doctor_name, doctor_surname, doctor_specialty from appointment where date< ? and patient_username= ? order by date") (Χρησιμοποιείται ως παράμετρος στη μέθοδο getPreviousAppointments)

### 3.10 Κώδικας του servlet PatientLogin

Το Servlet **PatientLogin** είναι απαραίτητο ώστε να παίρνει τα δεδομένα που έχει εισάγει ο ασθενής στη σελίδα patientLogin.jsp, να ελέγχει αν υπάρχει ασθενής με αυτά τα δεδομένα και να εισάγει τον ασθενή στην πλατφόρμα (στη σελίδα patientHome.jsp). Περιέχει τις μεθόδους init η οποία αρχικοποιεί το servlet και doPost η οποία παίρνει ως παραμέτρους τις request (περιέχει τα δεδομένα που εισήγαγε ο ασθενής στη φόρμα) και response και παίρνει το username και το password από τη σελίδα patientLogin.jsp (μέσω του post request στη σελίδα patientLogin.jsp), εκτελεί τις μεθόδους login και getUserDetails (βλ. παραπάνω), ορίζει τα δεδομένα που θα μεταφερθούν στη σελίδα patientHome.jsp μέσω του αιτήματος και στέλνει τα στοιχεία του ασθενή στη σελίδα patientHome.jsp. Σε περίπτωση ανεπιτυχούς σύνδεσης (για λόγους μη εγκυρότητας δεδομένων, κωλύματος στη βάση, κλπ) δεν στέλνονται τα δεδομένα και ο ασθενής παραμένει στη σελίδα patientLogin.jsp.

### 3.11 Περιγραφή του περιεχομένου των ιστοσελίδων

- index.jsp: Η αρχική σελίδα της εφαρμογής. Περιέχει συνδέσμους ανακατεύθυνσης προς τις ιστοσελίδες σύνδεσης των ασθενών (patientLogin.jsp), των γιατρών (doctorLogin.jsp) και των διαχειριστών (adminLogin.jsp) αντίστοιχα.

  Σημείωση: Οι δύο τελευταίες ιστοσελίδες θα αναπτυχθούν στην 3η εργασία.

- patientLogin.jsp: Η σελίδα όπου ο ασθενής εισάγει το όνομα χρήστη και τον κωδικό πρόσβασής του. Καλεί το PatientLogin servlet μέσω της μεθόδου post. Σε περίπτωση επιτυχούς σύνδεσης, εμφανίζεται η σελίδα patientHome.jsp. Σε αντίθετη περίπτωση, ο ασθενής παραμένει στην τρέχουσα σελίδα η οποία πλέον είναι κενή περιεχομένου.

- patientHome.jsp: Η σελίδα όπου εμφανίζονται στο χρήστη τα στοιχεία του και τα προηγούμενα ραντεβού του.

- web.xml: Συνδέει την εφαρμογή με το αρχείο context.xml του apache Tomcat (όπου εμφανίζονται οι πόροι που θα χρησιμοποιηθούν από την εφαρμογή).

### 3.12 Περιγραφή των πινάκων της βάσης δεδομένων

Οι πίνακες της βάσης δεδομένων DBKotza είναι οι εξής:

- Patient: Περιέχει τα στοιχεία των ασθενών (patient_AMKA, username, password, name, surname)

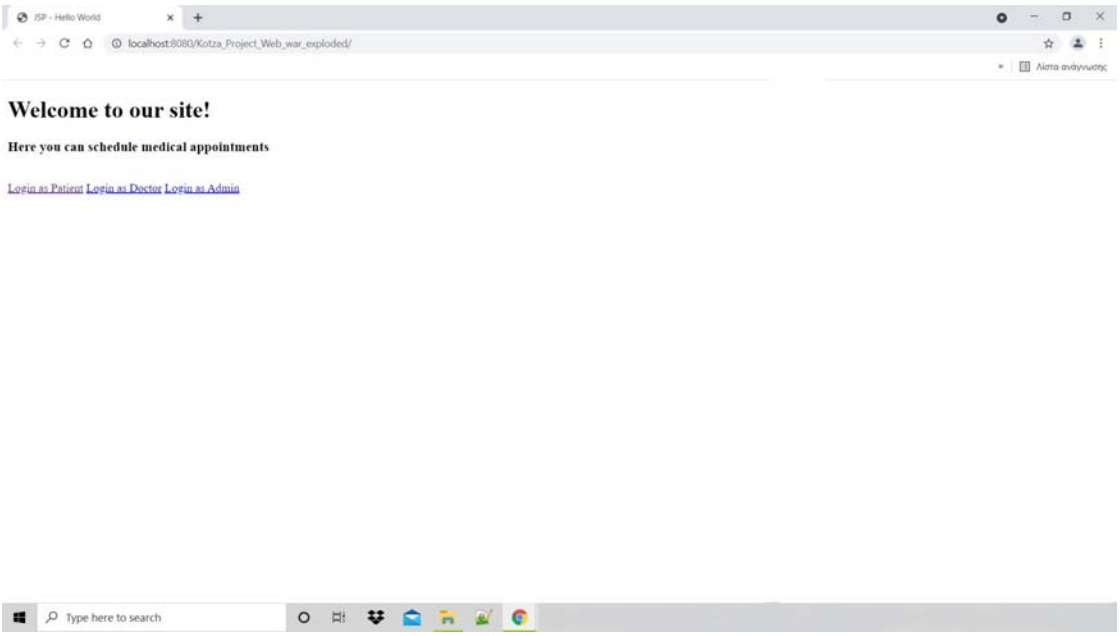
- Doctor: Περιέχει τα στοιχεία των γιατρών (doctor_AMKA, ADMIN_userid, username, password, specialty, name, surname)

- Admin: Περιέχει τα στοιχεία των διαχειριστών (ADMIN_userid, username, password)
- Appointment: Περιέχει τα στοιχεία των ραντεβού (appointmentID, date, time, doctor_AMKA, doctor_name, doctor_surname, doctor_specialty, patient_username)

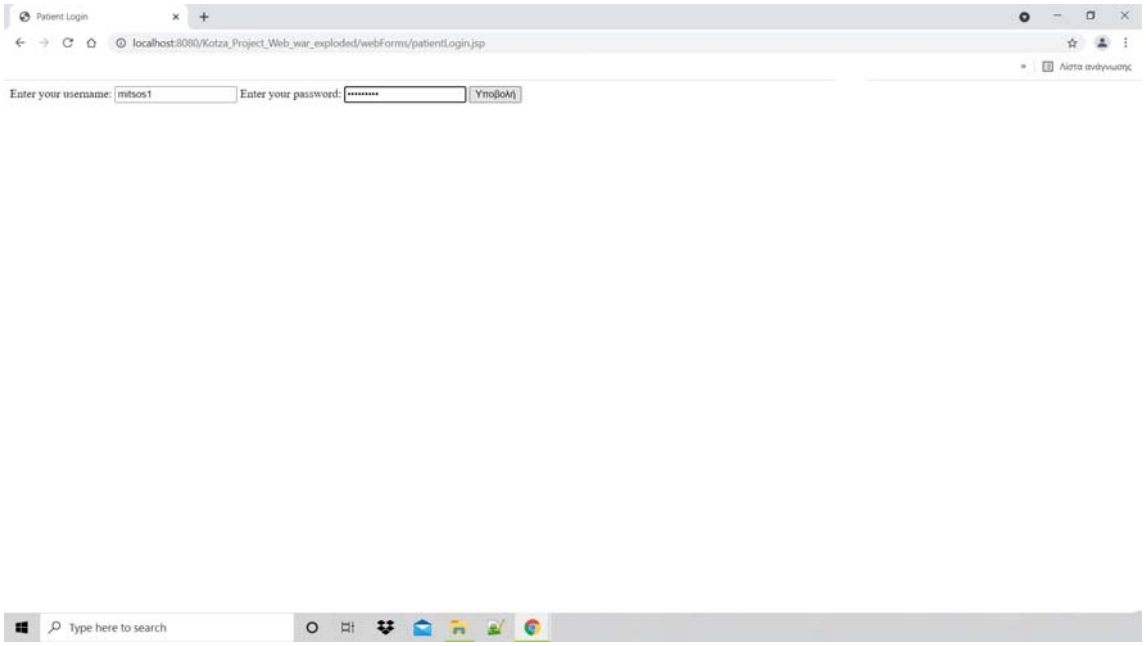
## 4. Βάση Δεδομένων

exercise_2.pgsql

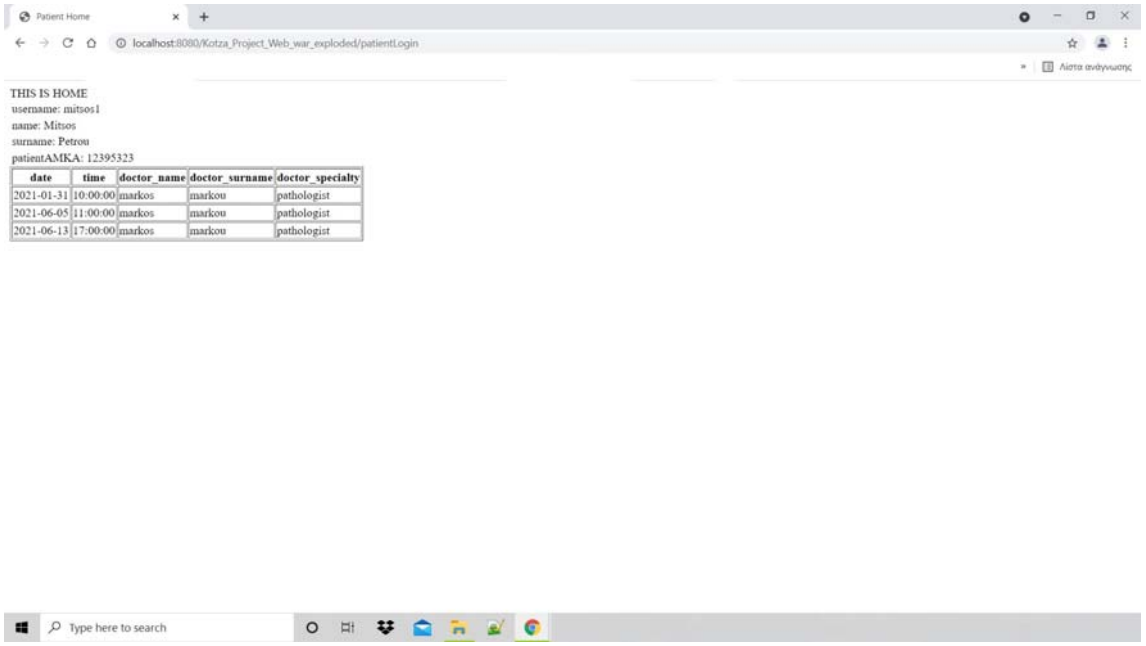
## 5. Στιγμιότυπα από την εκτέλεση του προγράμματος



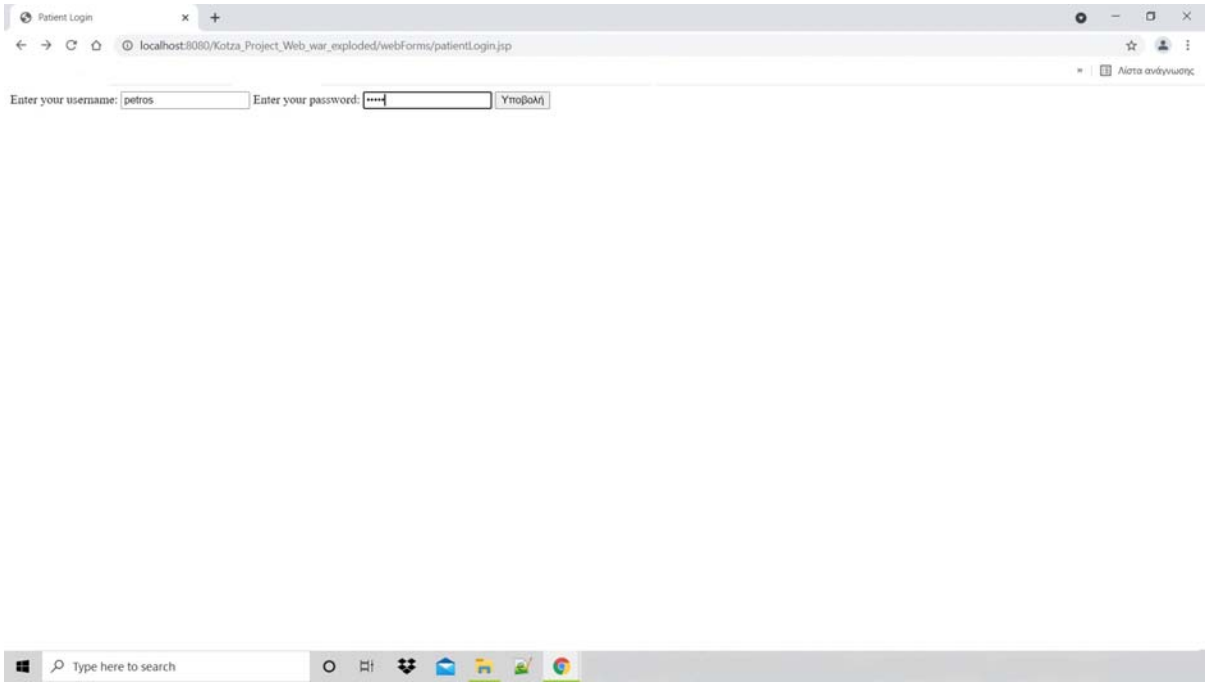

**Εικόνα 5.1 Αρχική σελίδα index.jsp**



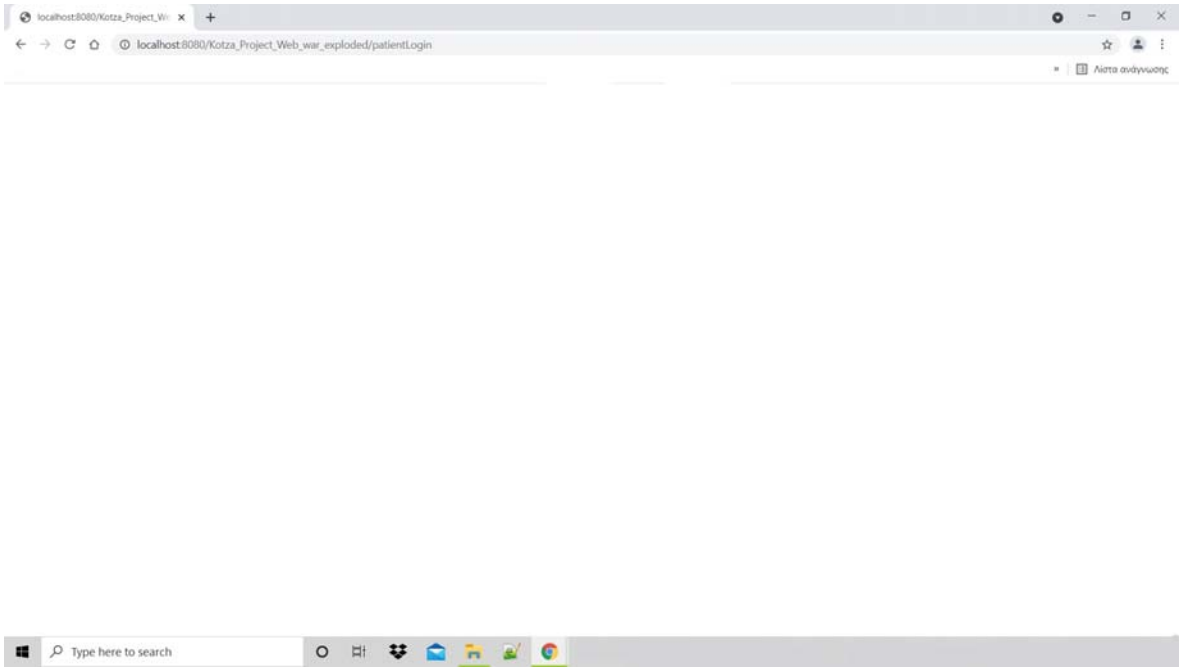

**Εικόνα 5.2 Σελίδα PatientLogin.jsp (επιτυχής σύνδεση)**




**Εικόνα 5.3 Σελίδα PatientHome.jsp (επιτυχής σύνδεση)**




**Εικόνα 5.4 Σελίδα PatientLogin.jsp (ανεπιτυχής σύνδεση)**




**Εικόνα 5.5 Σελίδα PatientLogin.jsp (ανεπιτυχής σύνδεση)**

## 6. Βιβλιογραφικές Πηγές

1. *MD5 hash in Java*. (2020). Ανακτήθηκε από: https://www.geeksforgeeks.org/md5-hash-in-java/
2. Hall, M. and Brown, L. (2006). *Servlets και σελίδες διακομιστή Java* [2η Αμερικανική Έκδοση] Αθήνα: Εκδόσεις Κλειδάριθμος