ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Τμήμα Πληροφορικής



Εργασία Μαθήματος «Προγραμματισμός στο διαδίκτυο και στον παγκόσμιο ιστό»

|  |  |
| --- | --- |
|  | ***Τελική εργασία*** |
| Όνομα φοιτητή – Αρ. Μητρώου  (όλων σε περίπτωση ομαδικής εργασίας) | ΜΙΧΑΛΗΣ ΣΤΥΛΙΑΝΙΔΗΣ – Π19165 |
| ΚΩΝΣΤΑΝΤΙΝΟΣ ΚΑΛΟΓΕΡΟΠΟΥΛΟΣ – Π19057 |
| ΕΛΕΥΘΕΡΙΟΣ ΚΟΝΤΟΥΡΗΣ - Π19077 |
|  |
| Ημερομηνία παράδοσης |  |

**Εκφώνηση Εργασίας**

ΤΕΛΙΚΟ ΠΑΡΑΔΟΤΕΟ ΜΑΘΗΜΑΤΟΣ

Δημιουργία 3-tier εφαρμογής για την διαχείριση ραντεβού ιατρικών εξετάσεων

Στόχοι εργασίας: Ολοκλήρωση λειτουργικότητας *3-tier εφαρμογής, ολοκλήρωση server-side τεχνολογιών (servlets και προαιρετικά jsp), επικοινωνία με βάση δεδομένων, ολοκλήρωση λειτουργιών.*

Στην τελική εργασία του μαθήματος θα επεκτείνετε την 2η Άσκηση ώστε να ολοκληρώσετε την εφαρμογή τριών επιπέδων (3-tier), η οποία θα υλοποιεί όλες τις λειτουργίες (μεθόδους) που ορίσατε στην 1η Άσκηση (με τις πιθανές αλλαγές που έγιναν).

Αναλυτικά Βήματα:

1. Επέκταση web project προηγούμενης άσκησης
   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. Δημιουργία διαδικτυακής διεπαφής
   1. Για την είσοδο των χρηστών στο σύστημα θα υλοποιείστε μηχανισμό login με username και password. Το password θα αποθηκεύεται σε κρυπτογραφημένη (hashed+salted) μορφή. Από την αρχική σελίδα οι διάφοροι χρήστες θα μπορούν να έχουν πρόσβαση στις λειτουργίες τους.
   2. Σε αυτό το βήμα, θα υλοποιήσετε τις διαδικτυακές διεπαφές (html ή jsp σελίδες) που θα χρησιμοποιούν οι χρήστες όλων των κατηγοριών (Ασθενείς, Ιατροί, Διαχειριστές) για να αλληλεπιδρούν με την εφαρμογή και να χρησιμοποιούν τις αντίστοιχες μεθόδους που απαιτούνται.
      1. Θα υπάρχει ένα κεντρικό μενού σε μία index.html (ή index.jsp) σελίδα, η οποία θα είναι η αρχική σελίδα για όλους τους χρήστες. Μετά τo login θα προβάλλεται το μενού λειτουργιών κάθε χρήστη ανάλογα με την κατηγορία στην οποία ανήκει.
      2. Λειτουργίες Ασθενών (Patient): Οι Ασθενείς θα μπορούν να εκτελούν κατ ελάχιστο τις λειτουργίες: προβολή ιστορικού προηγούμενων ραντεβού, προβολή διαθέσιμων κενών για κλείσιμο ραντεβού με έναν γιατρό κάποιας ειδικότητας, κλείσιμο ραντεβού ακύρωση ραντεβού (σε περίπτωση που το ραντεβού είναι προγραμματισμένο τουλάχιστον 3 ημέρες μετά).
      3. Λειτουργίες Ιατρών (Doctor): Οι Ιατροί θα μπορούν να εκτελούν κατ ελάχιστο τις λειτουργίες: καταχώρηση διαθεσιμότητας για ραντεβού (ανά μήνα), προβολή πίνακα ραντεβού, ακύρωση ραντεβού (σε περίπτωση που είναι τουλάχιστον 3 ημέρες μετά).
      4. Λειτουργίες Διαχειριστή (Administrator). Οι Διαχειριστές θα μπορούν να εκτελούν κατ ελάχιστο τις λειτουργίες: εισαγωγή νέου Ιατρού και χρήστη, διαγραφή Ιατρού.
   3. Η εφαρμογή θα υποστηρίζει διαχείριση συνόδου (session management) από τη στιγμή που ο χρήστης συνδέεται, μέχρι την αποσύνδεσή του από την εφαρμογή. Κατά την αποσύνδεση του χρήστη θα πρέπει να διαγράφεται το session.
3. Υλοποίηση επιπέδου Δεδομένων και σύνδεση εφαρμογής με τη βάση
   1. Όλα τα δεδομένα θα αποθηκεύονται σε βάση δεδομένων, την οποία έχετε σχεδιάσει από την 2η Άσκηση (π.χ. μέσω mysql + mysql workbench ή postgres;λ ή άλλης αντίστοιχης τεχνολογίας). Μπορείτε να προβείτε σε όποιες τροποποιήσεις θεωρείτε απαραίτητες. Προσθέσετε δοκιμαστικά δεδομένα στη βάση.
   2. Διαμορφώστε κατάλληλα το project σας ώστε να συνδέσετε τη Βάση Δεδομένων που έχετε δημιουργήσει με τον application server σας, ως μία 3-tier εφαρμογή (σύνδεση του application server με τη Βάση Δεδομένων και της εφαρμογής σας μέσω του applicaton server – μπορείτε να βρείτε αντίστοιχο παράδειγμα στα παραδείγματα κώδικα που περιλαμβάνονται στη σελίδα του μαθήματος).
4. Υλοποίηση επιπέδου επεξεργασίας (servlet)
   1. Διαμορφώστε κατάλληλα το project σας ώστε να επικοινωνεί με τον application server της επιλογής σας (στα java παραδείγματα έχουμε χρησιμοποιήσει apache tomcat).
   2. Υλοποιήσετε όλες τις λειτουργίες που προσφέρει η εφαρμογή σας χρησιμοποιώντας τεχνολογία servlet. Δημιουργήστε ένα ή περισσότερα servlet τα οποία θα δέχονται είσοδο από το επίπεδο διεπαφής (html ή jsp σελίδες και φόρμες), θα αναζητούν στη βάση δεδομένων τα στοιχεία που απαιτούνται ότι απαιτείται και θα επιστρέφουν το αποτέλεσμα στον εκάστοτε χρήστη ως δυναμική html σελίδα.
   3. Προαιρετικά, μπορείτε να χρησιμοποιήσετε τεχνολογία jsp για τη δημιουργία και την διαμόρφωση των ιστοσελίδων.

Πίνακας Περιεχομένων

[1 Εισαγωγή 4](#_Toc77090186)

[2 Servers 4](#_Toc77090187)

[2.1 Application Server 4](#_Toc77090188)

[2.2 Database Server 5](#_Toc77090189)

[3 Βάση Δεδομένων – Μοντέλο Οντοτήτων-Σχέσεων 7](#_Toc77090190)

[4 Εκκίνηση και Παραμετροποίηση του Web Project 8](#_Toc77090191)

[5 Λειτουργίες 8](#_Toc77090192)

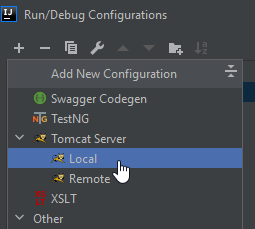
[6 Βιβλιογραφικές Πηγές 10](#_Toc77090193)

1. Εισαγωγή

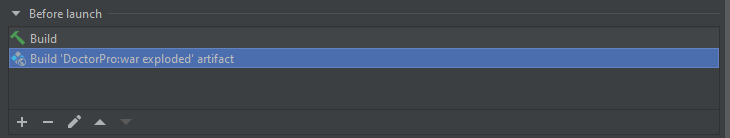
Για την δημιουργία της εφαρμογής χρησιμοποιήθηκε το IntelliJ IDEA 2021.1 (Ultimate Edition). Σε κάθε ενότητα δίνονται ακριβής οδηγίες για το πως να ρυθμιστούν σωστά τα εργαλεία που χρησιμοποιούνται.

1. Servers
   1. Application Server

Για Application Server χρησιμοποιήθηκε ο Tomcat 9. Αρχικά εγκαταστάθηκε και έπειτα με τη χρήση του IntelliJ έγινε παραμετροποίηση του. Το path στο οποίο εγκαταστάθηκε ο Tomcat έπρεπε να προστεθεί στο παράθυρο Settings -> Build, Execution, Deployment -> Application Servers κάνοντας κλικ στο ‘+’ πάνω αριστερά στο παράθυρο. Η σύνδεση με την εφαρμογή έγινε μέσο του παραθύρου ‘Run/Debug Configurations’, προσθέτοντας στη λίστα με τα configurations το Tomcat Local Server .



Όμως, για να τρέξει τελικά η εφαρμογή μέσα από το IDEA πρέπει να δημιουργηθεί ένα artifact. Στην προκείμενη περίπτωση δημιουργήθηκε το ‘DoctorPro:war exploded’ artifact μέσω του πεδίου Before Launch Tasks στο ίδιο παράθυρο.



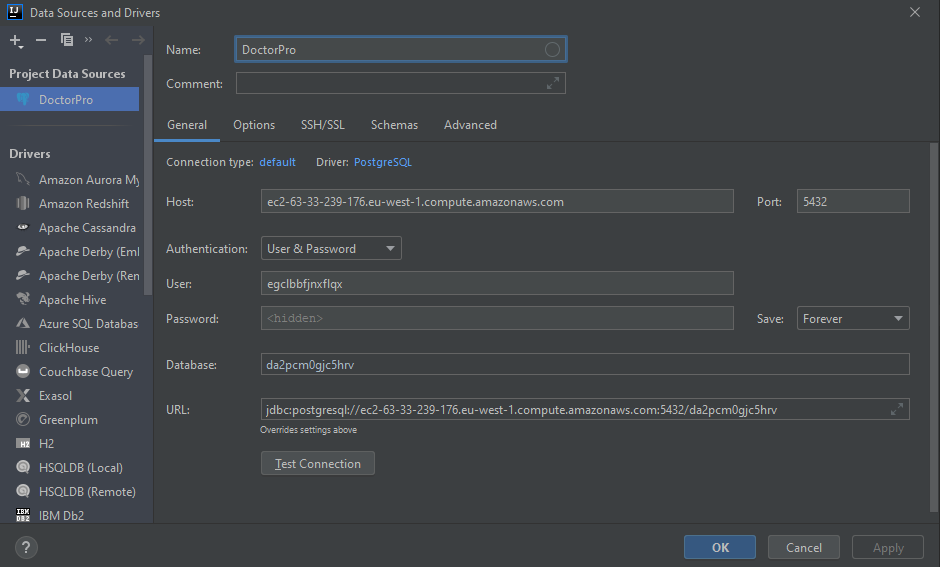
* 1. Database Server

Για Database Server έγινε χρήση του PostgresSQL. Η σύνδεση του με τον Tomcat 9 έγινε με τη χρήση του JDBC driver, μετακινώντας το αρχείο ‘postgresql-42.2.20.jar’, που κατεβάσαμε από την ιστοσελίδα ‘jdbc.postgresql.org’, στον κατάλογο ./apache-tomcat-9.0.46/lib.

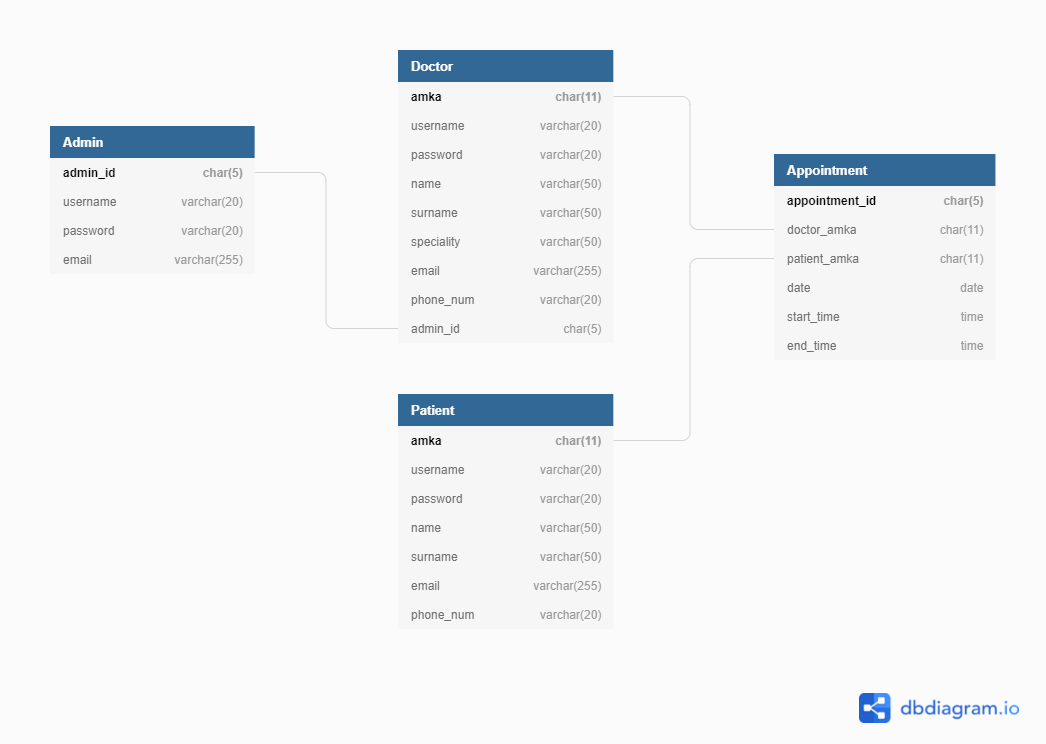
Εξαιτίας της ανάγκης να χρησιμοποιείται η ίδια βάση και λόγω ευκολίας στη διαχείριση, προχωρήσαμε σε hosting τη βάσης σε πραγματικό server. Για να την επίτευξη αυτής της ενέργειας χρησιμοποιήθηκε η υπηρεσία Heroku. Η πρόσβαση και απόκτηση των βασικών πακέτων έγινε μέσο του GitHub Student Developer Pack, όπου το Heroku αποτελεί ένα από τα προνόμια του.

Η σύνδεση με το IntelliJ έγινε μέσο του παραθύρου View/Tool-Windows/Databases, όπου προστέθηκε ως Data Source η PostgreSQL και έπειτα χρησιμοποιώντας τα πιστοποιητικά που εκδόθηκαν από το Heroku, ρυθμίστηκε η βάση. Μπορείτε να βρείτε τα στοιχεία αυτά στο αρχείο ‘database\_creds.txt’.

Παρακάτω παρουσιάζεται το παράθυρο όπου ρυθμίζεται η Βάση στο IntelliJ.



1. Βάση Δεδομένων – Μοντέλο Οντοτήτων-Σχέσεων



Το Μοντέλο Οντοτήτων-Σχέσεων περιέχει τους βασικούς πίνακες της εφαρμογής. Πιο συγκεκριμένα:

* **Doctor:** Ο πίνακας των γιατρών περιέχει τον ΑΜΚΑ του γιατρού, ο οποίος αποτελεί πρωτεύον κλειδί του πίνακα και είναι ένα σύνολο από 11 χαρακτήρες σταθερού μήκους. Τα υπόλοιπα πεδία, εκτός του admin\_id, είναι πεδία χαρακτήρων μεταβλητού μήκους, το καθένα με διαφορετικό ανώτατο όριο. Τέλος, έχουμε το πεδίο admin\_id, το οποίο σχετίζεται με τον πίνακα Admin και δηλώνει το ID του διαχειριστή που πρόσθεσε τον εκάστοτε γιατρό.
* **Patient:** Ο πίνακας των ασθενών φέρει πολλές ομοιότητες με τον πίνακα των Γιατρών. Η μόνη διαφορά έγκειται στην μη ύπαρξη των πεδίων της ειδικότητας (specialty) και του admin\_id, καθώς ο κάθε ασθενής είναι ο ίδιος υπεύθυνος για την εγγραφή του στο σύστημα.
* **Available\_Appointment:** Ο πίνακας αυτός αποθηκεύει τους διαθέσιμους χρόνους των γιατρών για ραντεβού. Κάθε ραντεβού έχει έναν μοναδικό 6ψήφιο ID για αναγνωριστικό, τους ΑΜΚΑ του γιατρού, την ημέρα (χωρίς ώρα) που θα διεξαχθεί το ραντεβού καθώς και την ώρα (χωρίς ζώνη ) που θα ξεκινήσει και θα τελειώσει. Σχετίζεται με τον πίνακα των Γιατρών καθώς από εκεί παίρνει τον ΑΜΚΑ του γιατρού με τον οποίο θα λαμβάνει χώρα το ραντεβού.
* **Scheduled\_Appointment:** Ο πίνακας αυτός αποθηκεύει τα ραντεβού που έχουν κλείσει οι ασθενείς με έναν γιατρό. Κάθε ραντεβού έχει έναν μοναδικό 6ψήφιο ID για αναγνωριστικό, τους ΑΜΚΑ του γιατρού και του ασθενή, την ημέρα (χωρίς ώρα) που θα διεξαχθεί το ραντεβού καθώς και την ώρα (χωρίς ζώνη ) που θα ξεκινήσει και θα τελειώσει. Σχετίζεται με τους πίνακες των Γιατρών και Ασθενών καθώς από εκεί παίρνει τον ΑΜΚΑ του γιατρού με τον οποίο λαμβάνει χώρα το ραντεβού και το ΑΜΚΑ του ασθένη, ώστε έτσι να υλοποιείται μια σύνδεση γιατρού-ασθενή για ένα ραντεβόυ.
* **Admin:** Ο πίνακας των διαχειριστών είναι υπεύθυνος για την αποθήκευση των διαχειριστών. Για κάθε διαχειριστή αποθηκεύει έναν μοναδικό 6ψήφιο αριθμό που λειτουργεί ως πρωτεύον κλειδί, ένα username, έναν κωδικό, το email του και το χαρακτηριστικό SuperUser. Οι admin χωρίζονται σε δύο κατηγορίες: αυτούς που έχουν ανεβασμενό επίπεδο δικαιωμάτων και μπορούν να προσθέσουν καινούργιους admin και σε αυτούς που μπορούν να προσθέσουν μόνο καινούργιους γιατρούς. Αυτό το χαρακτηριστικό ελέγχεται με το πεδίο SuperUser που είναι τύπου Boolean. Tέλος, σχετίζεται με τον πίνακα των Γιατρών καθώς οι διαχειριστές είναι υπεύθυνοι για την προσθήκη, διαγραφή και επεξεργασία των γιατρών.

Η Βάση δημιουργήθηκε από την αρχή στον database server και χρησιμοποιώντας το Mockaroo, μια δωρεάν εφαρμογή που τρέχει στον φυλλομετρητή, δημιουργήθηκαν εικονικά δεδομένα για την βάση. Στον πίνακα των ραντεβού χρησιμοποιήθηκαν στοιχεία που υπήρχαν ήδη από τους υπόλοιπους πίνακες. Για λόγους ευκολίας, στο αρχείο ‘test\_users.txt’ θα βρείτε ασθενείς που έχουν περασμένα ραντεβού.

1. Εκκίνηση και Παραμετροποίηση του Web Project

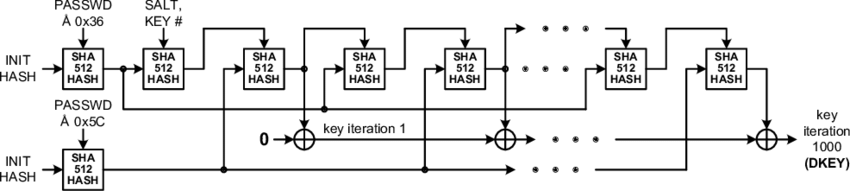
Η δημιουργία του DoctorProSystemCare έγινε μέσα από το παράθυρο ‘New Project’, επιλέγοντας το ‘Java Enterprise’ από τη λίστα. Ως template επιλέχθηκε το ‘Web Application’ ώστε να γνωρίζει το IntelliJ το είδος της εφαρμογής και να δημιουργήσει αυτόματα τους κατάλληλους καταλόγους και αρχεία.

Από την Εργασία 1 έγινε εισαγωγή των Model τάξεων Patient, Doctor, Appointment, Admin και User, οι οποίες τοποθετήθηκαν στο πακέτο ‘com.WebFlexers.models’.

Στο πακέτο ‘com.WebFlexers.servlets’ δημιουργήθηκαν τα servlets που θα χρησιμοποιηθούν για την εργασία. Ενδεικτικά τα PatientServlet, DoctorServlet και AdminServlet.

1. Υλοποίηση Μηχανισμού Login
2. Κρυπτογράφηση κωδικού(password)

Για την κρυπτογράφηση των κωδικών στην βάση δεδομένων έγινε χρήση του αλγορίθμου **PBKDF2(**Password-Based Key Derivation Function 2**)**.Ο παραπάνω αλγόριθμος κρίθηκε ικανοποιητικός λόγω της καλής αναλογίας ασφάλειας και απόδοσης που παρέχει απέναντι σε επιθέσεις τύπου brute-force και dictionary σε μία βάση δεδομένων. Όσο ανεβαίνει η ισχύς των υπολογιστών, τόσο πιο ευάλωτο είναι ένα είδος κρυπτογράφησης. Το πλεονέκτημα που παρέχει αυτός ο αλγόριθμος είναι η δυνατότητα να γίνει πιο αργός με βάση των αριθμό των επαναλήψεων και ως αποτέλεσμα να παράγεται ένα πιο δυνατό hash, καθιστώντας τον με αυτό τον τρόπο σχετικά πιο ασφαλή στον παραπάνω κίνδυνο που αναφέρθηκε. Για την παραγωγή του hash λαμβάνει ως παραμέτρους έναν αλγόριθμο SHA(Secure Hash Algorithms), τον αριθμό των επαναλήψεων(iteration number) που θα πρέπει να γίνουν για την παραγωγή του και το salt, το οποίο είναι ένας αριθμός από τυχαία παραγμένα bytes.



Περιγραφή της λειτουργίας του PBKDF2WithHmacSHA512

1. Λειτουργίες Διαχειριστή
2. Λειτουργίες Γιατρού
3. Λειτουργίες Ασθενή

Στην συγκεκριμένη εργασία έγινε υλοποίηση κάποιων σημαντικών λειτουργιών του Ασθενή. Πιο συγκεκριμένα:

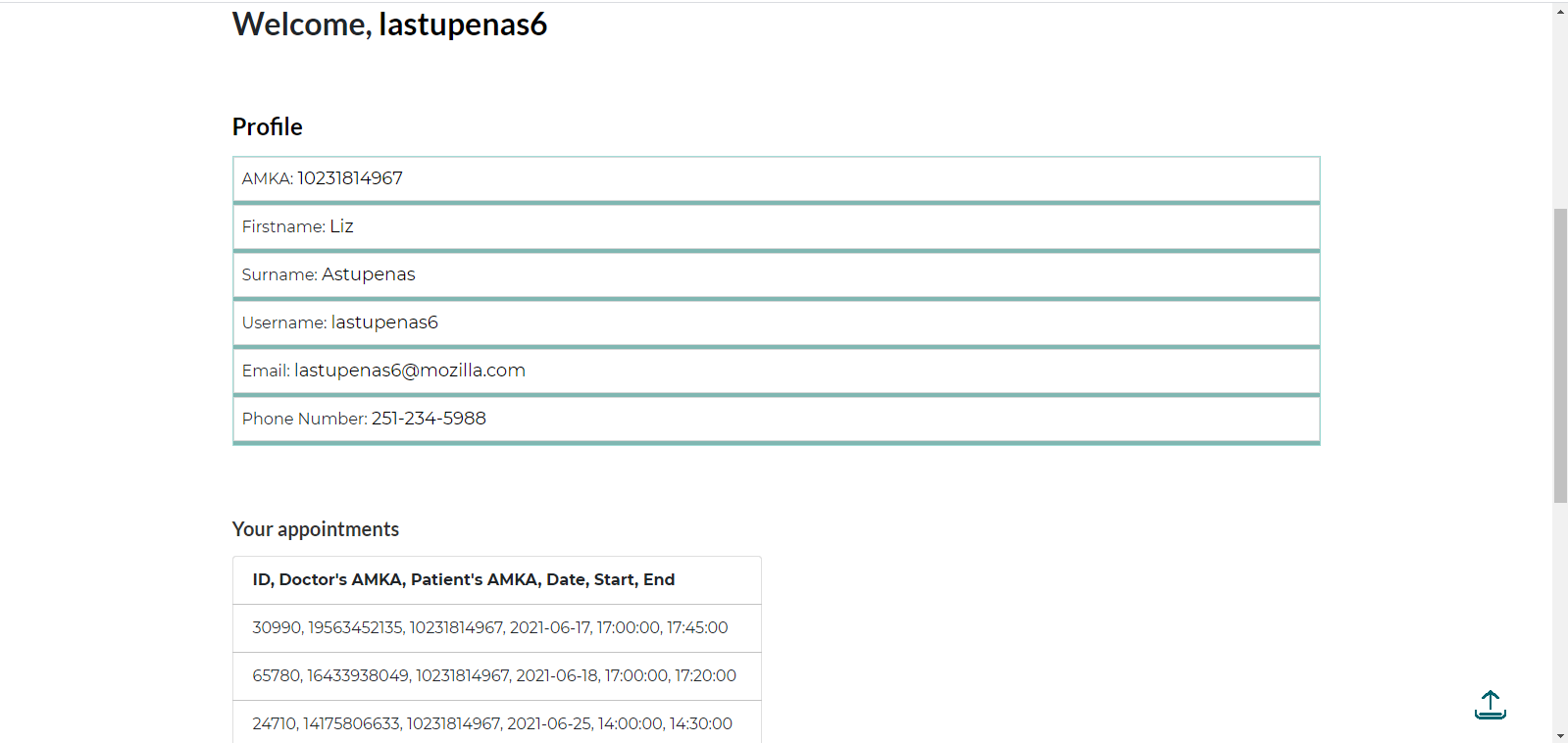
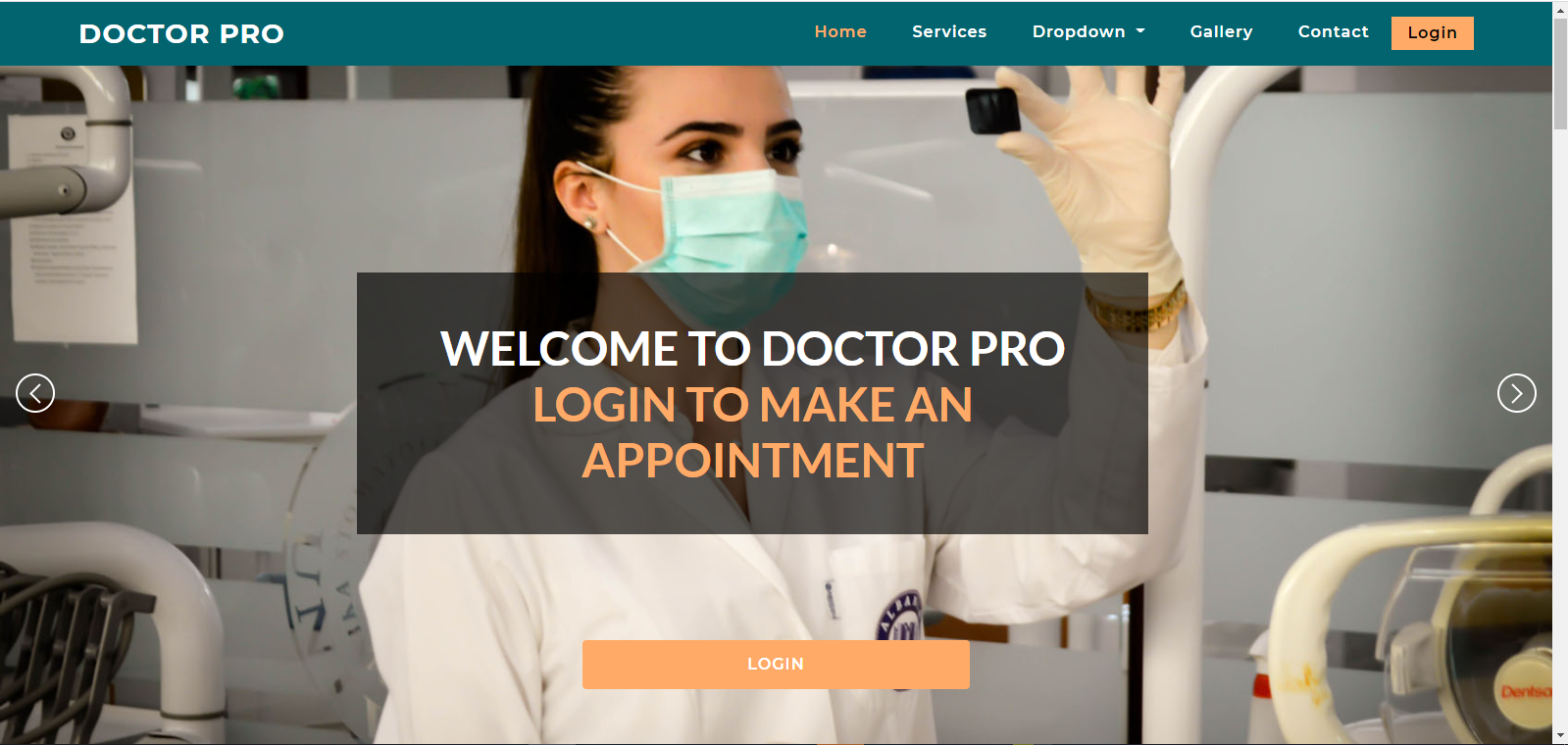
* **~~Login:~~** ~~Πατώντας το κουμπί ‘Login’ στη φόρμα σύνδεσης της κεντρικής σελίδας, στέλνεται ένα μήνυμα post στο PatientServlet. Αυτό επιτυγχάνεται με τα χαρακτηριστικά method και action της ετικέτας form. Στο action ορίζουμε το πως θα σταλούν τα δεδομένα και στο method τον προορισμό.~~

~~Έπειτα, το PatientServlet λαμβάνει το μήνυμα της φόρμας και αναλαμβάνει αρχικά να πιστοποιήσει τον χρήστη και έπειτα να τον συνδέσει στην σελίδα.~~

~~Η πιστοποίηση γίνεται μέσω της στατικής τάξης validatePatient(), η οποία βρίσκεται στην τάξη του μοντέλου Patient. Αυτό επρόκειτο να αλλάξει στην τρίτη εργασία εξαιτίας του ότι περιπλέκει πολύ τον κώδικα και τις λειτουργίες.~~

~~Ο κώδικας του Login ξεκινάει στην γραμμή 563 στην σελίδα ‘index.jsp’~~

* **~~Εμφάνιση στοιχείων:~~** ~~Αφού εκτελεστεί το query στην βάση και πιστοποιηθεί ο χρήστης τότε μέσω της, επίσης στατικής, τάξης getPatientDetails() λαμβάνονται σε έναν πίνακα τα στοιχεία του χρήστη. Έπειτα δημιουργείται ένα καινούργιο session και στέλνονται στην σελίδα ‘profile.jsp’ μέσω της setAttribute() οι πληροφορίες που λήφθηκαν, ώστε να εμφανιστούν. Ο κώδικας που εμφανίζει τα στοιχεία ξεκινάει από την γραμμή 131 στην σελίδα ‘profile.jsp’.~~
* **~~Προβολή ιστορικού ραντεβού:~~** ~~Για την λήψη των ραντεβού του ασθενή που συνδέθηκε, χρησιμοποιήθηκε η, επίσης, στατική συνάρτηση getAppointmentsHistory() από το μοντέλο του Patient, όπου χρησιμοποιώντας τον ΑΜΚΑ γίνεται ένα query στον πίνακα των ραντεβού και αν βρεθούν ραντεβού εμφανίζονται σε σχετικό πεδίο στην σελίδα ‘profile.jsp’. Η προβολή γίνεται με τη χρήση μιας επανάληψης foreach στην λίστα τον ραντεβού που έστειλε η setAttribute() η οποία με τη σειρά της έλαβε τα δεδομένα από την προαναφερόμενη συνάρτηση. Τον κώδικα θα τον βρείτε από την γραμμή 219 και έπειτα στην σελίδα ‘profile.jsp’.~~



1. Βιβλιογραφικές Πηγές

Template της σελίδας: ‘https://w3layouts.com/template/alleviating-medical-category-bootstrap-responsive-web-template/’

Ρυθμιση του intellij ultimate για Web Applications με χρήση Glassfish(Η ίδια ακριβως διαδικασία γίνεται και για τον Tomcat Application Server). https://www.jetbrains.com/help/idea/creating-and-running-your-first-java-ee-application.html