ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής

Εργασία Μαθήματος «Προγραμματισμός στο διαδίκτυο και στον παγκόσμιο ιστό»

ΑΣΚΗΣΗ 03	***ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΤΟ ΔΙΑΔΙΚΤΥΟ ΚΑΙ ΣΤΟΝ ΠΑΓΚΟΣΜΙΟ ΙΣΤΟ***
Φοιτητές στην εργασία:	Π19204 – Γεώργιος Σεϊμένης
	Π19064 – Ευστράτιος Καρκάνης
	Π19032 – Νικόλαος Γεωργιάδης
Ημερομηνία παράδοσης:	14/07/2021

Εκφώνηση τηςάσκησης

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

1.1. 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

2.1. 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.

2.2. 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.

2.2.1. 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 το login θα προβάλλεται το μενού λειτουργιών κάθε χρήστη ανάλογα με την κατηγορία στην οποία ανήκει.

2.2.2. Λειτουργίες Ασθενών (Patient): Οι Ασθενείς θα μπορούν να εκτελούν κατ' ελάχιστο τι 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).

2.2.3. 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).

2.2.4. Λειτουργίες Διαχειριστή (Administrator). Οι Διαχειριστές θα μπορούν να εκτελούν κατ' ελάχιστο τις λειτουργίες: εισαγωγή νέου Ιατρού και χρήστη, διαγραφή Ιατρού.

2.3. 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

3.1. Όλα τα δεδομένα θα αποθηκεύονται σε βάση δεδομένων, την οποία έχετε σχεδιάσει από την 2η Άσκηση (π.χ. μέσω mysql + mysql workbench ή postgres ή άλλης αντίστοιχης τεχνολογίας). Μπορείτε να προβείτε σε όποιες τροποποιήσεις θεωρείτε απαραίτητες. Προσθέσετε δοκιμαστικά δεδομένα στη βάση.

3.2. 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 application server – μπορείτε να βρείτε αντίστοιχο παράδειγμα στα παραδείγματα κώδικα που περιλαμβάνονται στη σελίδα του μαθήματος).

4. Υλοποίηση επιπέδου επεξεργασίας (servlet)

4.1. Διαμορφώστε κατάλληλα το project σας ώστε να επικοινωνεί με τον application server της επιλογής σας (στα java παραδείγματα έχουμε χρησιμοποιήσει apache tomcat).

4.2. 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.

4.3. Προαιρετικά, μπορείτε να χρησιμοποιήσετε τεχνολογία jsp για τη δημιουργία και την διαμόρφωση των ιστοσελίδων.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1 Γενική περιγραφή της εργασίας

Η παρούσα εργασία πρόκειται για ένα ολοκληρωμένο Web Project. Στην ουσία, η σύνθεση του περιλαμβάνει τόσο τα προγραμματιστικά στοιχεία της προηγούμενης εργασίας (κλάσεις και κώδικας), όσο και την εγκατάσταση ενός Web Server και μίας βάσης δεδομένων. Πιο συγκεκριμένα, στο συγκεκριμένο Project έχουν χρησιμοποιηθεί ο Web Server **Apache Tomcat 8.5** και η βάση δεδομένων **MySQL**. Επιπρόσθετα, έχουν προστεθεί πολλά καινούρια αρχεία (.java, .jsp, .css, .png, .html) τα οποία είναι απαραίτητα για την ορθή εκτέλεση του project.

Αξίζει να σημειωθεί ότι ιδιαίτερη έμφαση έχει δοθεί τόσο στο GUI (Graphical User Interface) της εφαρμογής (χρησιμοποιώντας αρχεία JSP και CSS) όσο και στην υλοποίηση των λειτουργιών που εκτελούνται από την πλευρά του εξυπηρετητή. Οι λεπτομέρειες αυτές περιγράφονται αναλυτικότερα παρακάτω.

2 Επαναχρησιμοποίηση του κώδικα

Αρκετά χρήσιμες φάνηκαν οι δύο προηγούμενες εργασίες σε οτιδήποτε αφορά την επαναχρησιμοποίηση των ήδη υπαρχουσών κλάσεων και κώδικα. Όλα τα στοιχεία τους ήταν ήδη φτιαγμένα, κι αφού έγιναν οι κατάλληλες τροποποιήσεις, χρησιμοποιήθηκαν παρέα με τα servlets και το γραφικό περιβάλλον της εφαρμογής.

3 Κώδικας Προγράμματος

3.1 Back End

3.1.1 Τύποι χρηστών

Για να γίνει πιο εύκολη η κατανόηση και η υλοποίηση των λειτουργιών της ιστοσελίδας, έγινε μια πολύ απλή παραδοχή: υπάρχουν τρεις τύποι χρηστών, που μπορούν χρησιμοποιήσουν την ιστοσελίδα και θα υλοποιήσουμε τις λειτουργίες τους σε δικές τους τάξεις (classes).

ΓΕΝΙΚΑ

Ξεκινώντας από ένα αφηρημένο σχέδιο θα φτάσουμε στα ειδικότερα. Για αυτόν τον λόγο όλες οι γενικευμένες λειτουργίες, που δεν χρειάζονται κάποια ειδίκευση, έχουν καταγραφεί σε μία κλάση, την οποία ονομάζουμε **Users**. Αυτή η κλάση περιλαμβάνει λειτουργίες σαν: την είσοδο/έξοδο (login/logout), την εγγραφή (register), την δυνατότητα ενημέρωσης για λανθασμένη ενέργεια (fail), όπως και λοιπές άλλες.

Εκτός, όμως, από τις λειτουργίες, υπάρχουν κάποια πολύ συγκεκριμένα χαρακτηριστικά, που κάθε χρήστης πρέπει να έχει. Τα πέντε από αυτά είναι για κάθε χρήστη και είναι το **όνομα**, το **επώνυμο**, το **username**, ο **κωδικός** και η **ηλικία**.

ΑΣΘΕΝΗΣ

Ο ασθενής είναι ο μόνος χρήστης της ιστοσελίδας που μπορεί να εγγραφεί από μόνος του στο σύστημα, πατώντας την σελίδα της *Register*. Η σελίδα είναι ειδικά φτιαγμένη για αυτόν, και περιέχει πεδία για κάθε χαρακτηριστικό ενός γενικού χρήστη, μαζί με έναν αριθμό ΑΜΚΑ (υπενθυμίζεται ότι αυτός ο αριθμός πρέπει να περιέχει ακριβώς 11 ψηφία).

Μία από τις βασικές λειτουργίες του ασθενούς είναι να ψάχνει, μέσα σε ένα συγκεκριμένο διάστημα που θα ορίζει αυτός, διαθέσιμους ιατρούς, ώστε να κλείσει ένα ραντεβού. Ο ασθενής δεν έχει τόσο μεγάλο έλεγχο σε αυτό, καθώς, για να κρατηθεί ένα ραντεβού, θα πρέπει, πρωτίστως, να είναι ο ιατρός διαθέσιμος. Μάλιστα, ο ασθενής δεν μπορεί να κλείσει όποια ημέρα επιθυμεί αυτός, αλλά αυτές που έχει ορίσει ο ιατρός ως «διαθέσιμες». Αλλά, ο ασθενής μπορεί να διαλέξει όποιον παράγοντα αναζήτησης επιθυμεί, ώστε να βρει το ραντεβού που θέλει. Μπορεί να επιλέξει ανάμεσα σε ΑΜΚΑ ιατρού, σε ειδικότητα ιατρού ή ονοματεπώνυμο.

Όταν, τελικά, ο ασθενής κλείσει το ραντεβού του, μπορεί να το δει αναλυτικότερα σε μια άλλη σελίδα. Μάλιστα, αν επιθυμεί ο ασθενής, μπορεί να ακυρώσει οποιοδήποτε ραντεβού εκκρεμεί.

Σε περίπτωση που περάσει η ημέρα του προγραμματισμένου ραντεβού, ο ασθενής έχει τη δυνατότητα να δει, όχι μόνο το συγκεκριμένο, αλλά και όλα τα προηγούμενα ραντεβού που έχουν πραγματοποιηθεί.

ΙΑΤΡΟΣ

Η εισαγωγή ενός ιατρού, δεν είναι όσο απλή όσο του ασθενούς. Ο ιατρός θα πρέπει να εισαχθεί στο σύστημα με τη βοήθεια κάπου διαχειριστή (που θα εξηγηθεί παρακάτω). Για να εισαχθεί ο ιατρός, όπως και ο ασθενής, θα πρέπει να έχει δηλωθεί στον λογαριασμό του κάποιο ΑΜΚΑ, συν την ειδικότητά του. Η

ειδικότητα δεν μπορεί να είναι οτιδήποτε, παρά μόνον τρεις από τις εξής επιλογές: οφθαλμίατρος, παθολόγος, ορθοπεδικός.

Ο ιατρός έχει περισσότερο έλεγχο με τα ραντεβού, καθώς θα πρέπει να δηλώσει ότι είναι διαθέσιμος, σε μία συγκεκριμένη ημέρα. Για να επιτευχθεί αυτό, εισάγουμε ένα ραντεβού στη βάση δεδομένων, δηλώνοντας ως ΑΜΚΑ ασθενούς τον αριθμό 0. Επισημαίνεται, επίσης, ότι κάθε ραντεβού διαρκεί, υποχρεωτικά, 30 λεπτά. Έχοντας αυτήν την παραδοχή υπ' όψη, μπορούμε να αποτρέψουμε τον Γιατρό να δηλώσει διαθεσιμότητα δύο φορές στην ίδια ώρα και στην ίδια ημέρα.

Όταν ο ιατρός δηλώσει διαθεσιμότητα, και κάποιος ασθενής, κλείσει ραντεβού με αυτόν τον, θα μπορεί ο ιατρός να δει το προγραμματισμένο του ραντεβού σε μία ειδικά προσαρμοσμένη σελίδα για αυτόν. Υπάρχει, επιπλέον, και η δυνατότητα της ακύρωσης του ραντεβού αυτού.

ΔΙΑΧΕΙΡΙΣΤΗΣ

Ο διαχειριστής μπορεί να ελέγχει, σχεδόν, τα πάντα μέσα στο σύστημα. Υπάρχει, βέβαια ο περιορισμός, που ο διαχειριστής μπορεί να δημιουργηθεί μόνο από άλλους διαχειριστές, δηλαδή ο ίδιος περιορισμός που ισχύει και με τον Γιατρό. Ο διαχειριστής δεν έχει κάποια επιπλέον στοιχεία π.χ. ΑΜΚΑ, παρά μόνον τα βασικά που έχει κάθε χρήστης.

Ο διαχειριστής έχει τη δυνατότητα να προσθέσει στο σύστημα οποιονδήποτε χρήστη. Αρκεί να συμπληρώσει όλα τα απαραίτητα στοιχεία που αναγράφονται για κάθε χρήστη. Βεβαίως, υπάρχει το πρόβλημα της διπλοεγγραφής. Οπότε, αν ο διαχειριστής επιχειρήσει να εισάγει έναν ασθενή με ένα ΑΜΚΑ που ήδη υπάρχει (είτε σε άλλο ιατρό, είτε σε άλλον ασθενή), τότε το σύστημα θα τον αποτρέψει και θα του εμφανίσει ανάλογο μήνυμα.

Με την δυνατότητα της προσθήκης έρχεται και η δυνατότητα διαγραφής από το σύστημα. Εισάγοντας το αντίστοιχο «κλειδί» για κάθε χρήστη, π.χ. για τους Γιατρούς και τους ασθενείς, το κλειδί είναι το ΑΜΚΑ και για τους διαχειριστές είναι το username, ο διαχειριστής μπορεί να διαγράψει χρήστες. Εφ' όσον έχει εξασφαλιστεί η λύση του προβλήματος της διπλοεγγραφής παραπάνω, δεν χρειάζονται έλεγχοι στον ιατρό και στον ασθενή. Ο μόνος έλεγχος που χρειάζεται κατά την διαγραφή, είναι ο διαχειριστής να επιχειρήσει να σβήσει τον εαυτό του. Μόνο σε αυτήν την περίπτωση χρειάζεται να αποτρέψουμε την ενέργεια της διαγραφής, καθώς το σύστημα δεν πρέπει να μείνει χωρίς διαχειριστές. Με αυτόν τον τρόπο, εξασφαλίζουμε ότι θα έχει μείνει στο σύστημα **τουλάχιστον ένας διαχειριστής**.

3.1.2 Servlets

Υπάρχουν τρία servlets, ένα για κάθε χρήστη. Τα servlets θα μας χρησιμεύσουν στο να πραγματοποιούνται οι ενέργειες που θα επιλέγει ο κάθε χρήστης. Όλες οι μέθοδοι είναι γραμμένες στις κλάσεις των αντίστοιχων χρηστών.

ΓΕΝΙΚΗ ΜΕΘΟΔΟΛΟΓΙΑ

Εντός **όλων** των ιστοσελίδων υπάρχουν μερικές «κρυμμένες» μεταβλητές που έχουν γραφεί με την βοήθεια της γλώσσας **JavaScript**. Αυτές οι μεταβλητές κρατούν ενίοτε αριθμούς, ενίοτε Strings. Το περιεχόμενό τους, δηλαδή, αντιστοιχεί **στην ενέργεια που έχει επιλέξει ο χρήστης**. Κάθε κουμπί υποβολής καλεί το αντίστοιχο Servlet που, ύστερα, θα αντιστοιχιστεί και στην κατάλληλη μέθοδο. Μία απλή υπόδειξη είναι αυτή της login.

```
<input type="hidden" name="admin_action" value="login">
<input type="hidden" name="patient_action" value="6">
<input type="hidden" name="doctor_action"  value="login">
```

ΚΡΥΜΜΕΝΕΣ ΜΕΤΑΒΛΗΤΕΣ ΣΤΗΝ LOGIN.

Όπως φαίνεται παραπάνω, το περιεχόμενο των μεταβλητών αντιστοιχεί σε μία ενέργεια. Για τα Servlets του διαχειριστή και του ιατρού, το περιεχόμενο, πρέπει να αντιστοιχεί στην login με το συνθηματικό «login». Για το Servlet του ασθενούς, πρέπει να ορίσουμε τον αριθμό 6.

Συγκεκριμένα για την σελίδα της login, θα πρέπει να προσέξουμε, καθώς θα πρέπει να καλέσουμε το σωστό Servlet, ανάλογα με το ποιος χρήστης επιχειρεί να μπει στην ιστοσελίδα. Πάλι, χρησιμοποιώντας κώδικα JavaScript, έχουμε υλοποιήσει να αλλάζει το Servlet που θα καλείται, κάθε φορά που ο χρήστης πατάει ένα από τα radio buttons.

ΚΛΗΣΗ ΜΕΘΟΔΩΝ ΜΕΣΩ ΤΩΝ SERVLETS

Όπως προαναφέραμε, τα Servlets είναι μονάχα, τα εργαλεία που θα μας οδηγούν στην σωστή κλήση των μεθόδων. Όλα τα άλλα, τα αναλαμβάνουν οι μέθοδοι των κλάσεων. Οπότε σε κάθε Servlet, η μεθοδολογία, είναι σχεδόν η ίδια. Σε όλα τα Servlets, για επαυξημένη ασφάλεια, έχουμε χρησιμοποιήσει την **do Post**, ώστε να μην φαίνονται τα περιεχόμενα στο URL. Σε όλες τις μεθόδους do Post, έχουμε πάντοτε μία **switch-case** που ελέγχει το περιεχόμενο των προαναφερθεισών μεταβλητών. Έκτοτε, σε κάθε case καλούμε και την ανάλογη μέθοδο από κάθε κλάση (όλες οι μέθοδοι είναι στατικές), αφού πρώτα πάρουμε τα δεδομένα από την αντίστοιχη σελίδα και τα περάσουμε ως ορίσματα στις μεθόδους που καλούμε.

Θα πρέπει να προσέξουμε στο κομμάτι της βάσης δεδομένων, καθώς τα πάντα θα μπορούσαν να πάνε στραβά στον τομέα αυτόν. Οπότε, την πρώτη φορά που καλείται ένα Servlet (όποιο κι αν είναι αυτό), **αρχικοποιούμε το Data Source,** στο οποίο θα βρίσκεται η βάση δεδομένων μας. Φροντίζουμε, επίσης, η μεταβλητή αυτή να είναι στατική, καθώς θα χρησιμοποιείται πολύ συχνά εντός του Servlet.

3.1.3 Sessions

Το θέμα της ασφάλειας της ιστοσελίδας, το καλύπτουν τα Sessions. **Όλες οι ιστοσελίδες που έχουν προσωπικά δεδομένα είναι υλοποιημένες σε JSP**, πράγμα που σημαίνει ότι μπορούμε να πιστοποιήσουμε με Java αν είναι ασφαλές να εκτελέσουμε την εντολή του χρήστη. Επίσης, αν ένας χρήστης κάνει έξοδο από την πλατφόρμα, θα πρέπει να μην τον αφήσουμε να πατήσει το κουμπί back και να πάει πίσω στην σελίδα, με τα στοιχεία που έκανε login.

LOGIN & LOGOUT

Κατά την είσοδο, θα πρέπει να δημιουργήσουμε attributes, τα οποία θα έχει κάθε χρήστης που κάνε επιτυχή είσοδο στο σύστημα. Κάθε φορά, που ένας χρήστης κάνει επιτυχή είσοδο στο σύστημα, πρέπει πρώτα να «καταστρέψουμε» το προηγούμενο session και να φτιάξουμε ένα καινούργιο. Κάθε φορά που γίνεται αυτό, εξασφαλίζουμε ότι δεν θα υπάρχουν ήδη δεδομένα στο session, άρα κι το ότι δεν θα είναι δύο χρήστες ταυτόχρονα συνδεδεμένοι. Στην έξοδο, θα πρέπει να καταστρέψουμε τα attributes από το session που κάναμε στην είσοδο του χρήστη, αλλά και να κάνουμε invalidate αυτό το session.

SESSIONS ΜΕΣΑ ΣΤΙΣ ΣΕΛΙΔΕΣ

Μέσα σε όλες τις σελίδες, που αντιστοιχούν σε κάποια ενέργεια χρήστη, πρέπει να ελέγχουμε τα attributes από το session που έκανε είσοδο ο χρήστης. Επίσης, πρέπει να απενεργοποιήσουμε την μνήμη cache, ώστε να μην μπορεί να πάει πίσω στην ίδια σελίδα ο χρήστης, αφ' ότου κάνει logout.

3.1.4 Αξιοποίηση πηγών στο κομμάτι του back end

Όσον αφορά τον προγραμματισμό της εφαρμογής στο κομμάτι του back end, υπήρχαν αρκετά κομμάτια, τα οποία συμβουλευτήκαμε από online πηγές. Οι πηγές που χρησιμοποιήθηκαν, αναφέρονταν στην επίλυση των ακόλουθων ζητημάτων:

- ποια μέθοδος θα εκτελεστεί όταν ο χρήστης πατήσει κάποιο κουμπί (onclick event), [1].

- για τον έλεγχο των ημερομηνιών που εισάγει ο χρήστης (μορφή του πεδίου – format [2], το εύρος min και max των ημερομηνιών που ο χρήστης μπορεί να εισάγει [3] και [4]).

- πως να λάβουμε όλα τα attributes ενός session, [5].

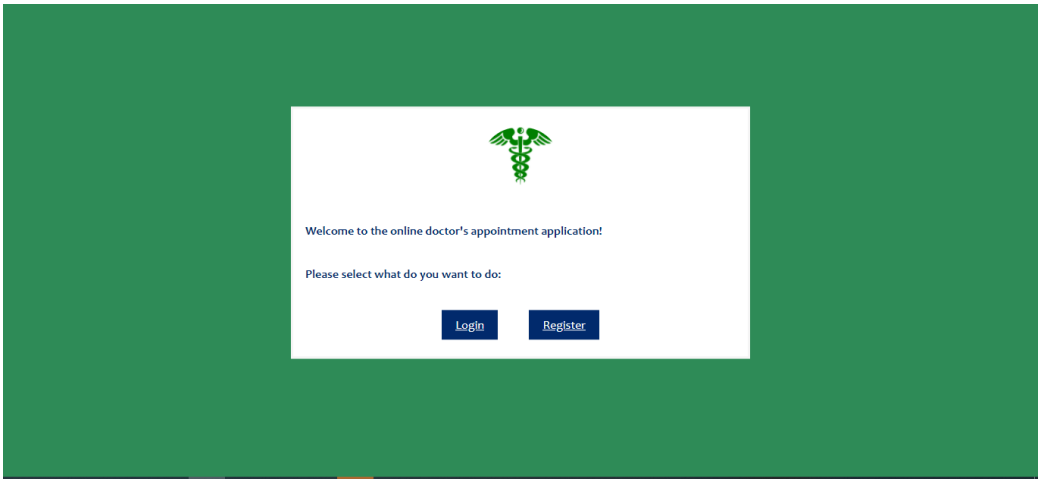
- μετατροπή ενός πίνακα που περιέχει χαρακτήρες ASCII σε string,[6].

- επιλογή των εβδομαδιαίων ραντεβού ενός χρήστη από την βάση δεδομένων, [7].

3.2 Front End

3.2.1 Το μενού της εφαρμογής

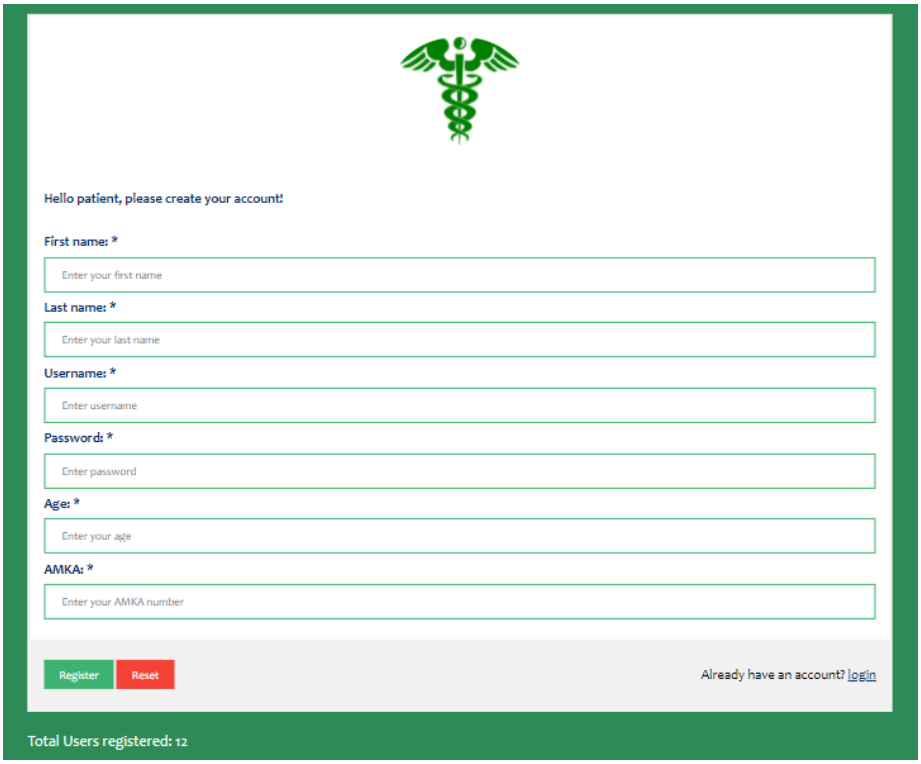
Το πρόγραμμα όταν φορτώσει εμφανίζει σε κάθε χρήστη (ιατρό, διαχειριστή ή ασθενή) την σελίδα *index.html*. Πρόκειται για το κεντρικό μενού της εφαρμογής. Ο χρήστης μπορεί να επιλέξει τι θέλει να κάνει (εγγραφή ή σύνδεση) και να συνεχίσει αναλόγως. Ακολουθεί σχετική εικόνα της αρχικής σελίδας:



Η σελίδα index.html

3.2.2 Η σελίδα εγγραφής (register)

Η σελίδα της εγγραφής (*register.jsp*) επιτρέπει σε έναν ασθενή να δημιουργήσει έναν νέο λογαριασμό στο σύστημα. Πρόκειται για μία φόρμα εισαγωγής στοιχείων (**First name, Last name, Username, Password, Age, AMKA**) του ασθενή. Τα στοιχεία αυτά αποθηκεύονται στη συνέχεια σε μία βάση δεδομένων, όταν ο ασθενής πατήσει το κουμπί “Register” της φόρμας. Ακολουθεί σχετική εικόνα της σελίδας:

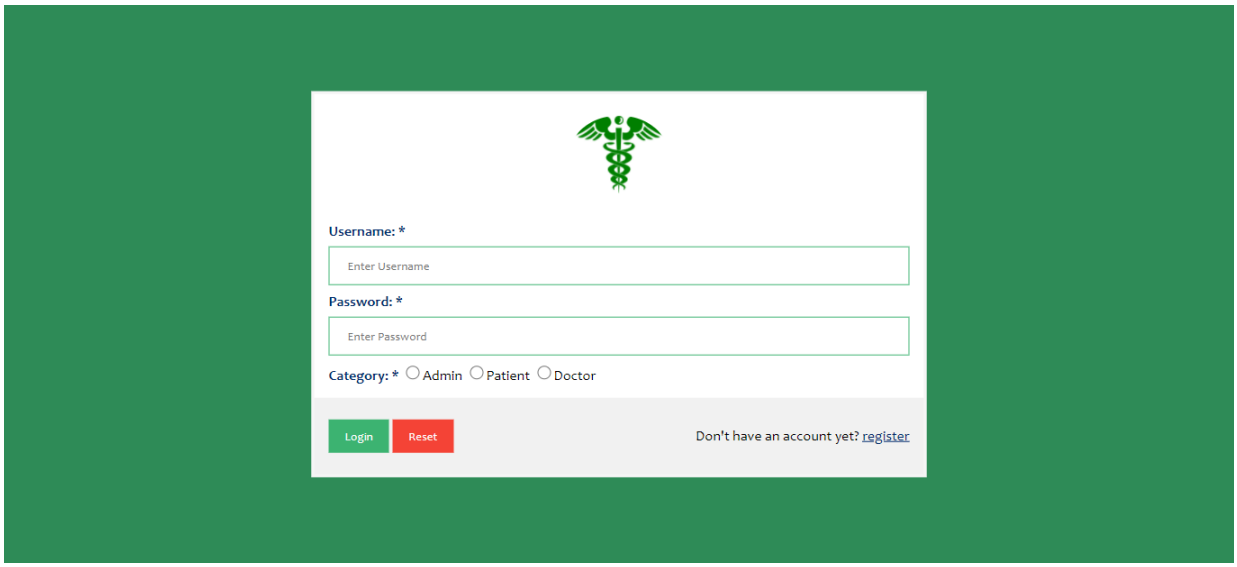


Η σελίδα register.jsp

Σημειώνεται ότι γίνεται έλεγχος εγκυρότητας των δεδομένων που εισάγονται τόσο από την πλευρά του πελάτη (χρήση JavaScript) όσο και από την πλευρά του server. Οι λεπτομέρειες αυτές έχουν αναλυθεί περεταίρω στο παραδοτέο της **δεύτερης** εργασίας.

3.2.3 Η σελίδα σύνδεσης (login)

Η σελίδα της εγγραφής (*login.jsp*) επιτρέπει σε έναν χρήστη (ιατρό, διαχειριστή ή ασθενή) να συνδεθεί στο σύστημα. Πρόκειται για μία φόρμα εισαγωγής στοιχείων (**Username, Password**) του χρήστη. Επιπλέον, ο χρήστης καλείται να συμπληρώσει και την κατηγορία στην οποία ανήκει, δηλαδή αν είναι ιατρός, ασθενής ή διαχειριστής. Αυτό είναι εφικτό με τα radio buttons που υπάρχουν στην φόρμα login. Τα κουμπάκια αυτά (radio buttons) βοηθούν την διαδικασία της σύνδεσης, έτσι ώστε να ενεργοποιείται κάθε φορά το αντίστοιχο servlet της εφαρμογής. Μόλις ο χρήστης πατήσει το κουμπί "*Login*" εμφανίζεται αν γίνεται ή όχι επιτυχής η σύνδεσή του στο σύστημα. Ακολουθεί σχετική εικόνα της σελίδας:



Η σελίδα login.jsp

Σε περίπτωση λανθασμένης εισαγωγής στοιχείων, εμφανίζεται στην οθόνη του χρήστη η σελίδα *fail.html*. Ακολουθεί σχετική εικόνα της σελίδας:

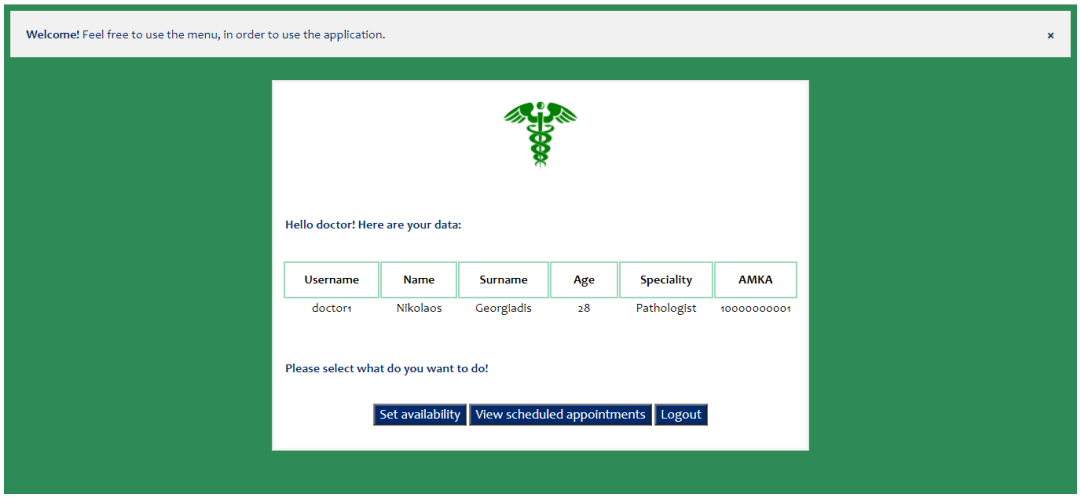
Η σελίδα fail.html

Σε περίπτωση επιτυχούς σύνδεσης εμφανίζεται το προσωπικό μενού κάθε χρήστη (ανάλογα την κατηγορία του χρήστη, εμφανίζεται διαφορετικό μενού με διαφορετικές λειτουργίες).

3.2.4 Τα κύρια μενού των χρηστών

3.2.4.1 Ιατρός

Στην παρακάτω φωτογραφία βλέπουμε το κύριο μενού ενός χρήστη κατηγορίας ιατρού (doctor):



Η σελίδα doctor_main_environment.jsp

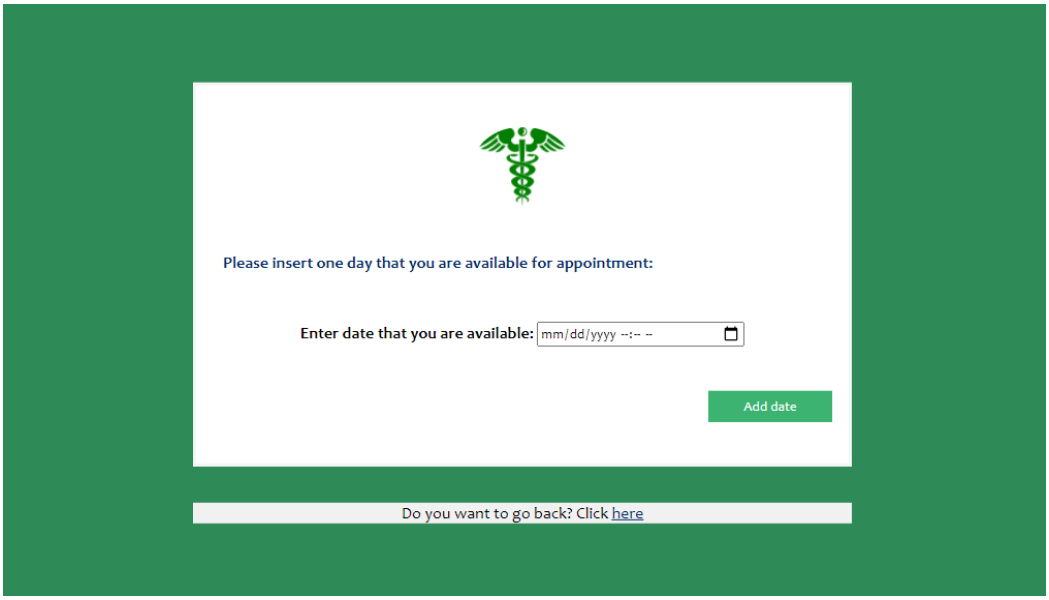
Η συγκεκριμένη σελίδα (*doctor_main_environment.jsp*) εμφανίζεται στην οθόνη μετά από μία επιτυχή σύνδεση (login) του ιατρού στο σύστημα. Η σελίδα περιλαμβάνει τα ακόλουθα στοιχεία για κάθε ιατρό που συνδέεται:

1) **τα στοιχεία του ιατρού** (όπως αυτά βρίσκονται αποθηκευμένα στη βάση δεδομένων). Συγκεκριμένα εμφανίζονται τα ακόλουθα στοιχεία σε μορφή πίνακα:

Username	Name	Surname	Age	Speciality	AMKA

2) **τρία κουμπιά ενεργειών** που εκτελούν έκαστο διαφορετική λειτουργία. Το πρώτο κουμπί (*Set availability*) επιτρέπει σε έναν ιατρό να ορίσει το πότε είναι διαθέσιμος για ραντεβού. Ακολούθως, το δεύτερο κουμπί (*View scheduled appointments*) επιτρέπει στον χρήστη να δει όλα τα προγραμματισμένα ραντεβού που έχει (ανά μήνα ή ανά εβδομάδα) και το τελευταίο κουμπί (*Logout*) κάνει αποσύνδεση του ιατρού από το σύστημα.

Όταν ένας ιατρός που έχει συνδεθεί στο σύστημα, πατήσει το πρώτο κουμπί (*Set availability*) για να ορίσει ο ίδιος την διαθεσιμότητά του για ραντεβού, εμφανίζεται η σελίδα *doctor_set_availability.jsp*. Η σελίδα αυτή εμφανίζεται στην επόμενη φωτογραφία:



Η σελίδα doctor_set_availability.jsp

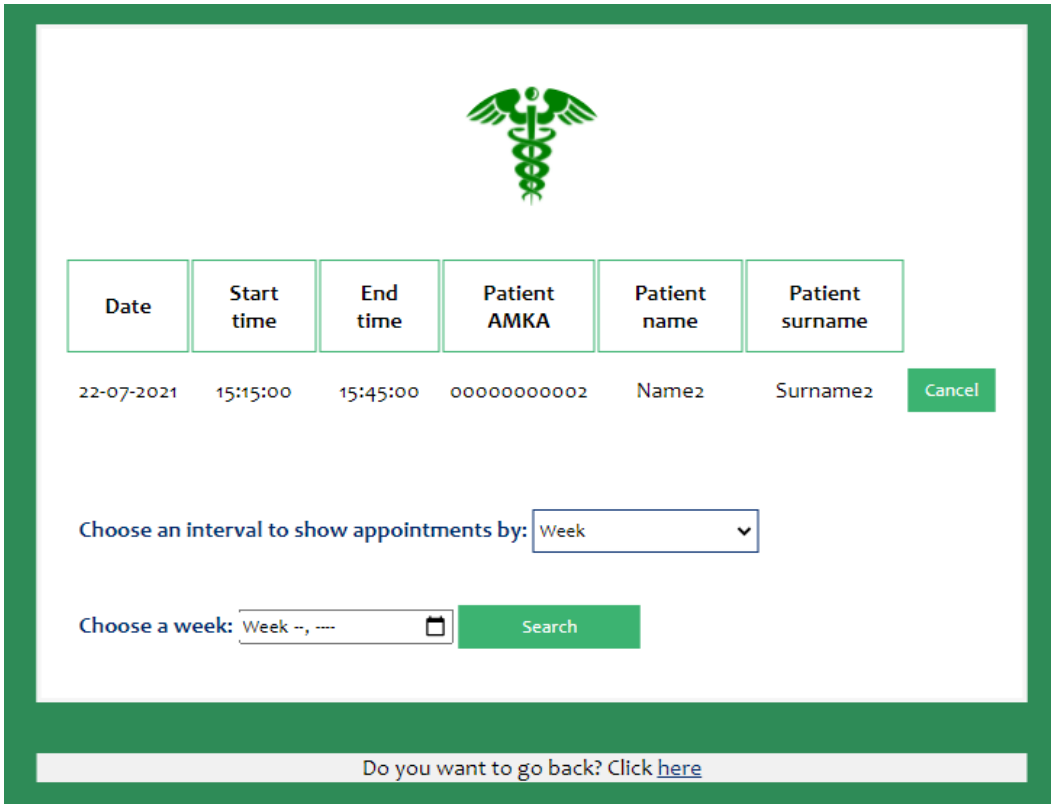
Εδώ ο ιατρός απλώς εισάγει μία ημερομηνία και πατάει το κουμπί "*Add date*" για να δηλωθεί η διαθέσιμη αυτή ημερομηνία στο σύστημα. Αν η δήλωση της νέας διαθέσιμης ημερομηνίας του ιατρού αποθηκευτεί στη βάση σωστά, εμφανίζεται η ακόλουθη σελίδα:

Επιτυχής εισαγωγή διαθέσιμης ημερομηνίας

Όταν ο ιατρός πατήσει το δεύτερο κουμπί (*View scheduled appointments*) εμφανίζεται στην οθόνη η σελίδα *doctor_view_appointments.jsp,* όπως φαίνεται και στην επόμενη φωτογραφία:

Η σελίδα doctor_view_appointments.jsp

Ο ιατρός λοιπόν, μπορεί να αναζητήσει όλα τα ραντεβού που έχει είτε ανά μήνα, είτε ανά εβδομάδα. Αυτό γίνεται με την βοήθεια της drop down λίστας. Για παράδειγμα, στην παρακάτω φωτογραφία βλέπουμε τα διαθέσιμα ραντεβού (τα ήδη προγραμματισμένα) για τον ιατρό με username “doctor1”:



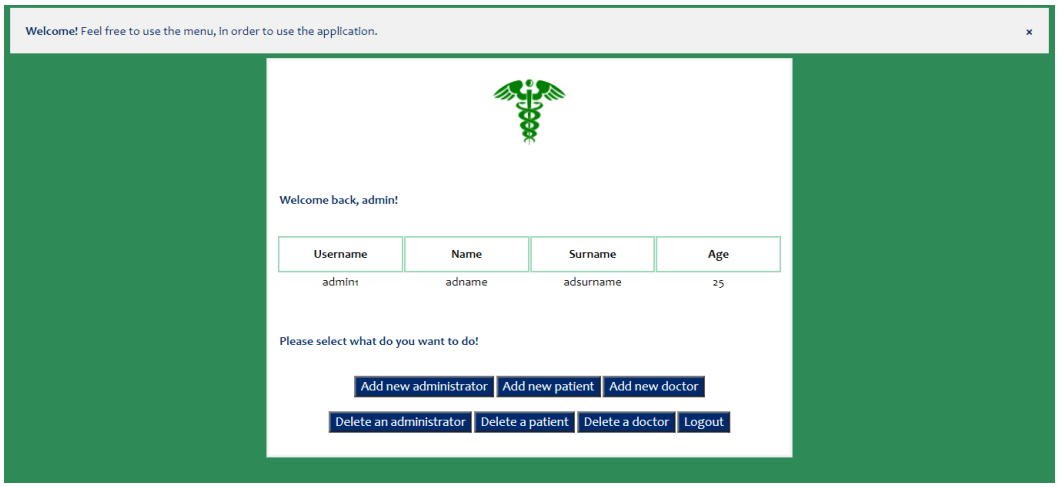
Διαθέσιμα ραντεβού του ιατρού doctor1

Ανά πάσα στιγμή, ο ιατρός μπορεί να ακυρώσει ένα ραντεβού που ήδη έχει. Αυτό γίνεται πατώντας το κουμπί «*Cancel*» (κάθε διαθέσιμο ραντεβού που έχει ένας ιατρός, εμφανίζεται στην σελίδα doctor_view_appointments.jsp και από δίπλα από κάθε ραντεβού υπάρχει ένα κουμπί «*Cancel*» για την ακύρωση του εκάστοτε ραντεβού).

Τέλος, το κουμπί "*Logout*" ΄κλείνει την σύνδεση (session) και μεταφέρεται ο έλεγχος του προγράμματος στην σελίδα *login.jsp.*

3.2.4.2 Διαχειριστής

Στην παρακάτω φωτογραφία βλέπουμε το κύριο μενού ενός χρήστη κατηγορίας διαχειριστή (admin):



Η σελίδα admin_main_environment.jsp

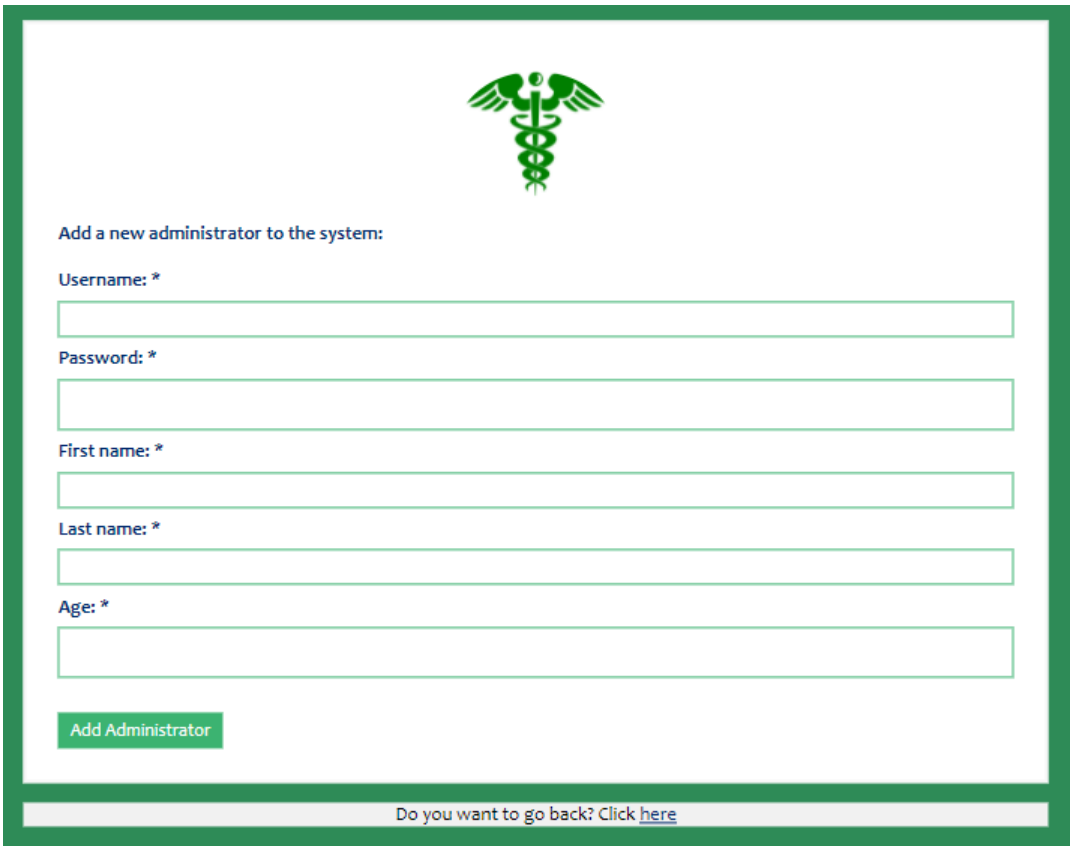
Η συγκεκριμένη σελίδα (*admin_main_environment.jsp*) εμφανίζεται στην οθόνη μετά από μία επιτυχή σύνδεση (login) του διαχειριστή στο σύστημα. Η σελίδα περιλαμβάνει τα ακόλουθα στοιχεία για κάθε admin που συνδέεται:

1) **τα στοιχεία του διαχειριστή** (όπως αυτά βρίσκονται αποθηκευμένα στη βάση δεδομένων). Συγκεκριμένα εμφανίζονται τα ακόλουθα στοιχεία σε μορφή πίνακα:

Username	Name	Surname	Age

2) **επτά κουμπιά ενεργειών** που εκτελούν έκαστο διαφορετική λειτουργία. Οι λειτουργίες κάθε κουμπιού αναλύονται τώρα:

   - **κουμπί Add new administrator:** Όταν ένας διαχειριστής πατήσει το συγκεκριμένο κουμπί, στην ουσία μεταφέρεται στην σελίδα *add_new_admin.jsp*, για να εισάγει στο σύστημα έναν νέο διαχειριστή (admin). Η σελίδα αυτή φαίνεται στο επόμενο screenshot:



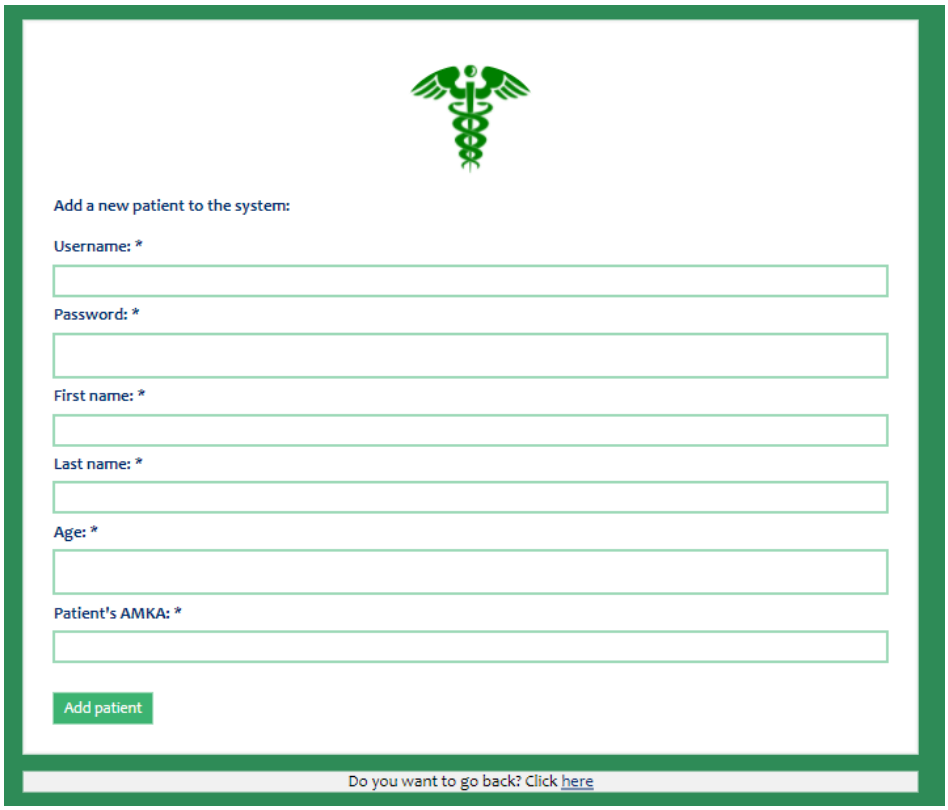
Εισαγωγή νέου διαχειριστή

Ο διαχειριστής έπειτα εισάγει τα στοιχεία (του νέου διαχειριστή που θα προστεθεί στην εφαρμογή) στη φόρμα και πατάει το κουμπί *"Add Administrator"* για την εκτέλεση αυτής της ενέργειας. Έπειτα από την επιτυχή εισαγωγή, εμφανίζεται η ακόλουθη σελίδα:

Επιτυχής εισαγωγή διαχειριστή

Σημειώνεται ότι κατά την υποβολή των στοιχείων, γίνεται έλεγχος εγκυρότητας των δεδομένων της φόρμας από την πλευρά του εξυπηρετητή. Οι έλεγχοι αυτοί έχουν αναλυθεί σε προηγούμενη εργασία.

- **κουμπί Add new patient:** Όταν ένας διαχειριστής πατήσει το συγκεκριμένο κουμπί, στην ουσία μεταφέρεται στην σελίδα *add_new_patient.jsp*, για να εισάγει στο σύστημα έναν νέο ασθενή (patient). Η σελίδα αυτή φαίνεται στο επόμενο screenshot:



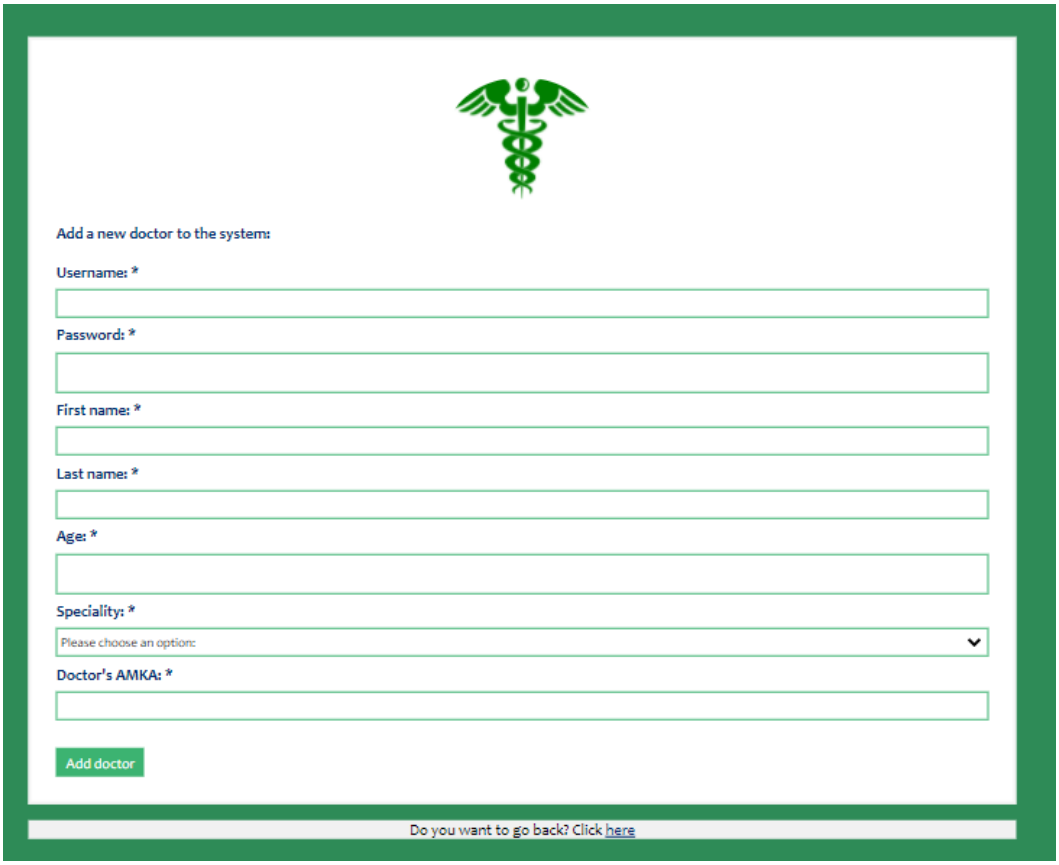
Εισαγωγή νέου ασθενή

Ο διαχειριστής έπειτα εισάγει τα στοιχεία (του νέου ασθενή που θα προστεθεί στην εφαρμογή) στη φόρμα και πατάει το κουμπί *"Add patient"* για την εκτέλεση αυτής της ενέργειας. Έπειτα από την επιτυχή εισαγωγή, εμφανίζεται η ακόλουθη σελίδα:

Επιτυχής εισαγωγή ασθενή

Σημειώνεται ότι κατά την υποβολή των στοιχείων, γίνεται έλεγχος εγκυρότητας των δεδομένων της φόρμας από την πλευρά του εξυπηρετητή. Οι έλεγχοι αυτοί έχουν αναλυθεί σε προηγούμενη εργασία.

- **κουμπί Add new doctor:** Όταν ένας διαχειριστής πατήσει το συγκεκριμένο κουμπί, στην ουσία μεταφέρεται στην σελίδα *add_new_doctor.jsp*, για να εισάγει στο σύστημα έναν νέο ιατρό (doctor). Η σελίδα αυτή φαίνεται στο επόμενο screenshot:



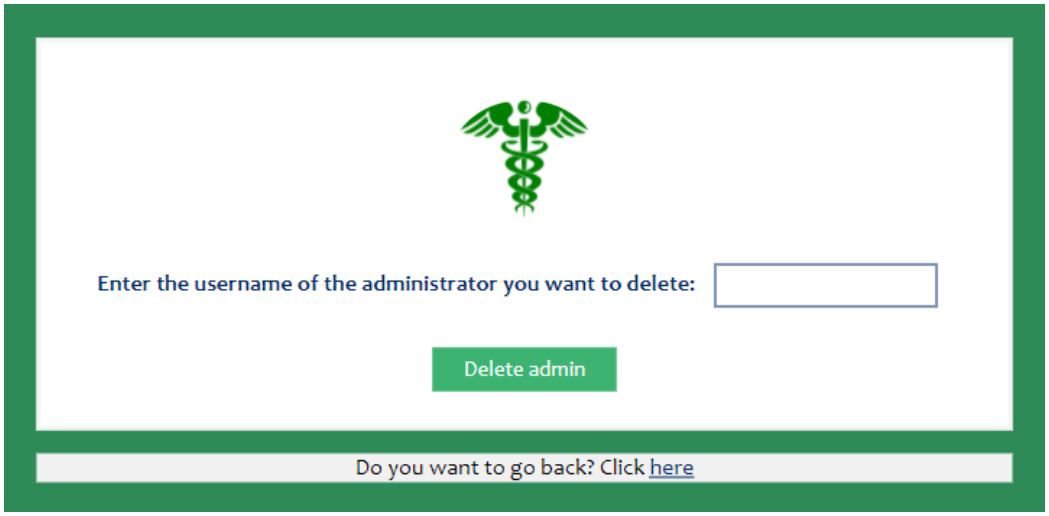
Εισαγωγή νέου ιατρού

Ο διαχειριστής έπειτα εισάγει τα στοιχεία (του νέου ιατρού που θα προστεθεί στην εφαρμογή) στη φόρμα και πατάει το κουμπί "*Add patient*" για την εκτέλεση αυτής της ενέργειας. Έπειτα από την επιτυχή εισαγωγή, εμφανίζεται η ακόλουθη σελίδα:

Επιτυχής εισαγωγή ιατρού

Σημειώνεται ότι κατά την υποβολή των στοιχείων, γίνεται έλεγχος εγκυρότητας των δεδομένων της φόρμας από την πλευρά του εξυπηρετητή. Οι έλεγχοι αυτοί έχουν αναλυθεί σε προηγούμενη εργασία. Παρόλα αυτά, σε αυτή την εργασία έχει προστεθεί και ο έλεγχος για το αν ένας ιατρός έχει μία ΜΟΝΟ από τις ακόλουθες ειδικότητες: Οφθαλμίατρος, Παθολόγος ή Ορθοπαιδικός.

- **κουμπί Delete an administrator:** Όταν ένας διαχειριστής πατήσει το συγκεκριμένο κουμπί, στην ουσία μεταφέρεται στην σελίδα *delete_admin.jsp*, για να διαγράψει από το σύστημα έναν διαχειριστή (admin). Η σελίδα αυτή φαίνεται στο επόμενο screenshot:

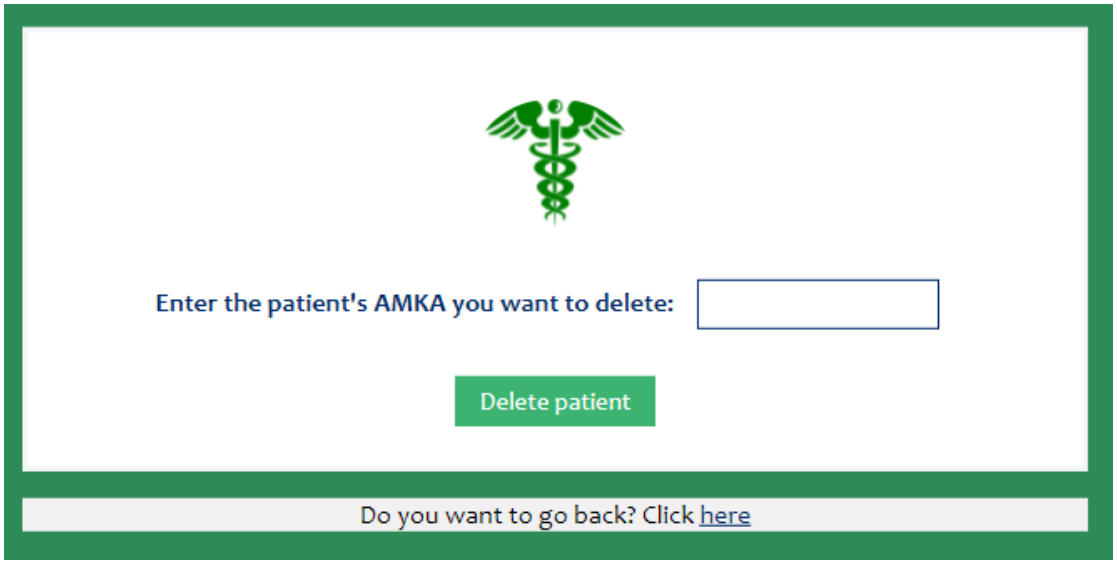


Διαγραφή ενός διαχειριστή

Ο διαχειριστής εισάγει το username του διαχειριστή που θέλει να διαγράψει. Η επιτυχής διαγραφή δηλώνεται με την εμφάνιση της ακόλουθης σελίδας:

Επιτυχής διαγραφή διαχειριστή

- **κουμπί Delete a patient:** Όταν ένας διαχειριστής πατήσει το συγκεκριμένο κουμπί, στην ουσία μεταφέρεται στην σελίδα *delete_patient.jsp*, για να διαγράψει από το σύστημα έναν ασθενή (patient). Η σελίδα αυτή φαίνεται στο επόμενο screenshot:

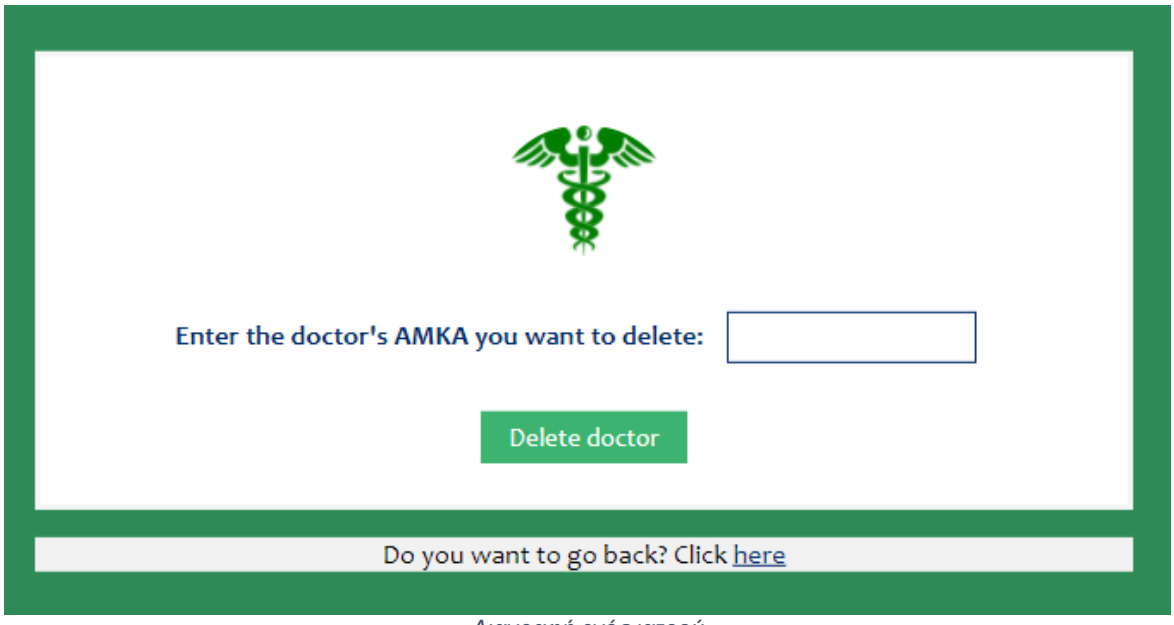


Διαγραφή ενός ασθενή

Ο διαχειριστής εισάγει το ΑΜΚΑ του ασθενή που θέλει να διαγράψει. Η επιτυχής διαγραφή δηλώνεται με την εμφάνιση της ακόλουθης σελίδας:

Επιτυχής διαγραφή ασθενή

- **κουμπί Delete a doctor:** Όταν ένας διαχειριστής πατήσει το συγκεκριμένο κουμπί, στην ουσία μεταφέρεται στην σελίδα *delete_doctor.jsp*, για να διαγράψει από το σύστημα έναν ιατρό (doctor). Η σελίδα αυτή φαίνεται στο επόμενο screenshot:



Διαγραφή ενός ιατρού

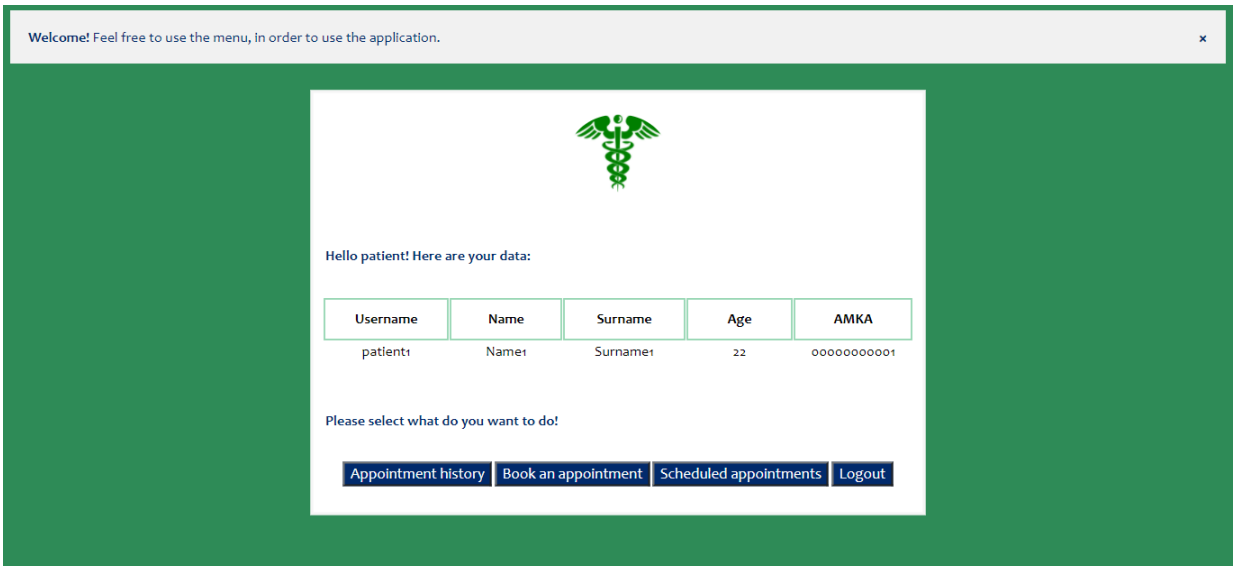
Ο διαχειριστής εισάγει το ΑΜΚΑ του ιατρού που θέλει να διαγράψει. Η επιτυχής διαγραφή δηλώνεται με την εμφάνιση της ακόλουθης σελίδας:

Επιτυχής διαγραφή ιατρού

- **κουμπί Logout:** όταν ο διαχειριστής πατήσει αυτό το κουμπί, κλείνει την σύνδεση (session) και μεταφέρεται ο έλεγχος του προγράμματος στην σελίδα *login.jsp.*

3.2.4.3 Ασθενής

Στην παρακάτω φωτογραφία βλέπουμε το κύριο μενού ενός χρήστη κατηγορίας ασθενή (patient):



Η σελίδα patient_main_environment.jsp

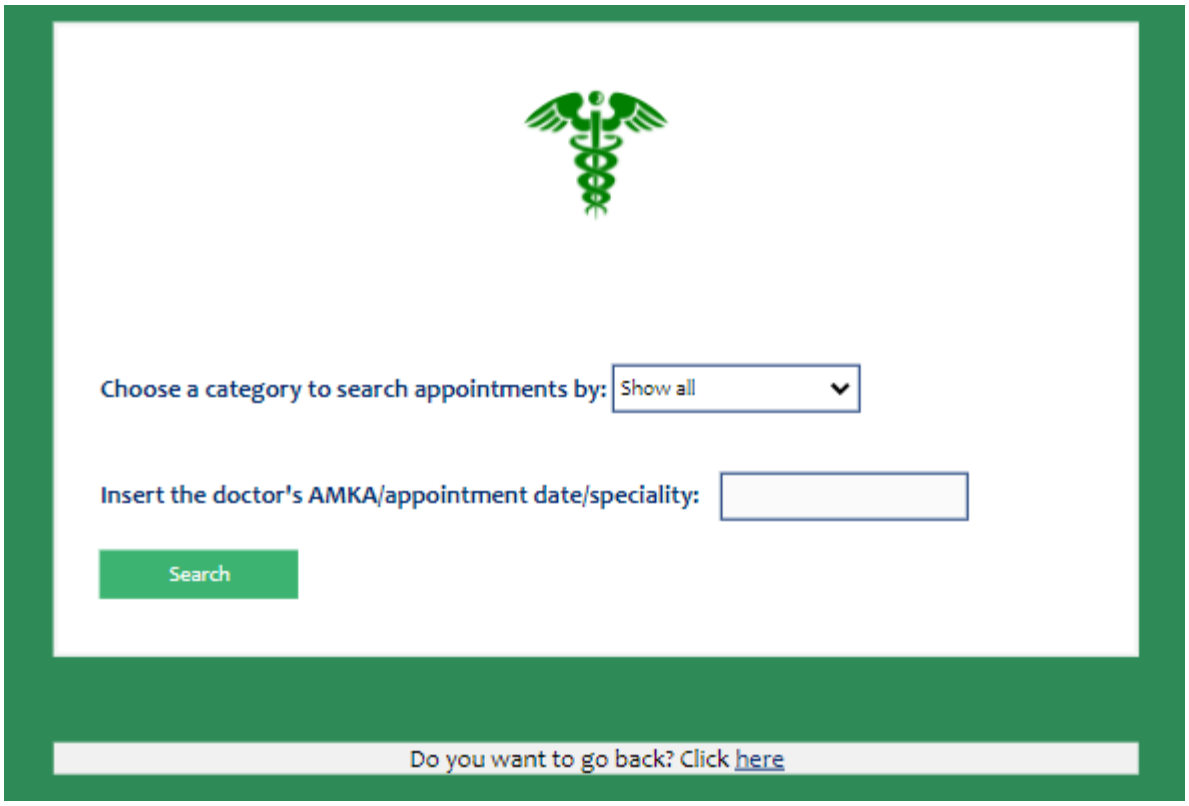
Η συγκεκριμένη σελίδα (*patient_main_environment.jsp*) εμφανίζεται στην οθόνη μετά από μία επιτυχή σύνδεση (login) του ασθενή στο σύστημα. Η σελίδα περιλαμβάνει τα ακόλουθα στοιχεία για κάθε ασθενή που συνδέεται:

1) **τα στοιχεία του ασθενή** (όπως αυτά βρίσκονται αποθηκευμένα στη βάση δεδομένων). Συγκεκριμένα εμφανίζονται τα ακόλουθα στοιχεία σε μορφή πίνακα:

Username	Name	Surname	Age	AMKA

2) **τέσσερα κουμπιά ενεργειών** που εκτελούν έκαστο διαφορετική λειτουργία. Το πρώτο κουμπί (*Appointment history*) επιτρέπει σε έναν ασθενή να δει το ιστορικό των ραντεβού που είχε. Το επόμενο κουμπί (*Book an appointment*) επιτρέπει στον ασθενή να κλείσει ένα νέο ραντεβού. Το τρίτο κουμπί (*Scheduled appointments*) επιτρέπει στον χρήστη να δει όλα τα προγραμματισμένα ραντεβού που έχει και, τέλος, το τελευταίο κουμπί (*Logout*) αποσυνδέει τον ασθενή από το σύστημα (κλείνοντας το session).

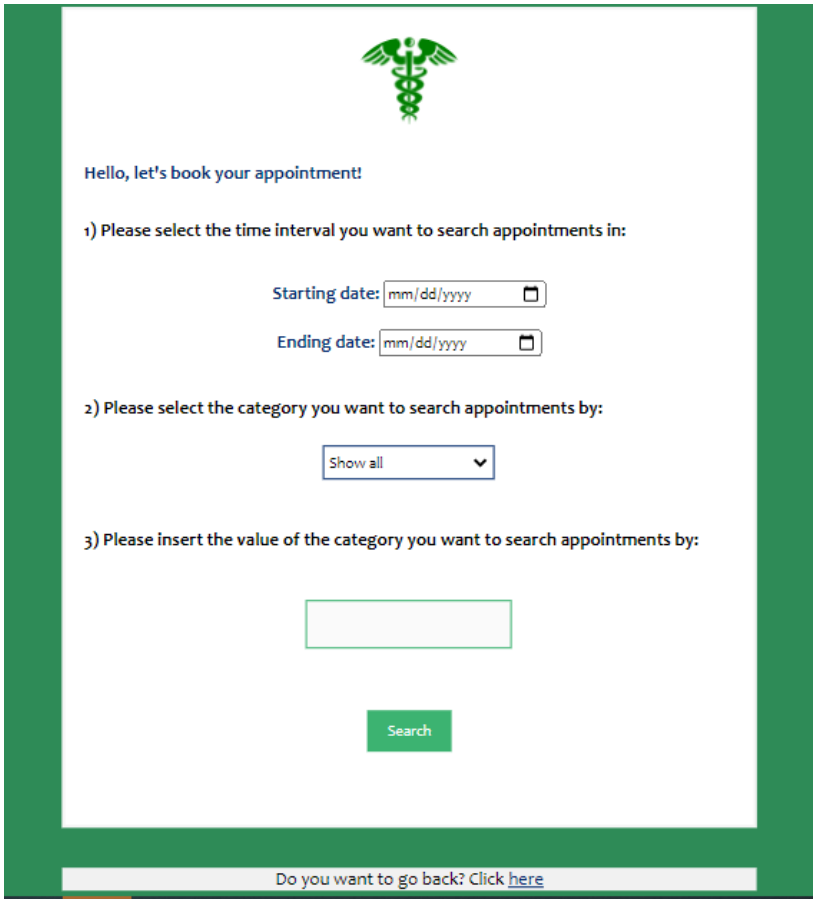
Όταν ένας ασθενής που έχει συνδεθεί στο σύστημα, πατήσει το πρώτο κουμπί (*Appointment history*) θα μεταφερθεί στην σελίδα *appointmenthistory.jsp* για να δει όλα τα ραντεβού που είχε στο παρελθόν. Η σελίδα φαίνεται στην επόμενη φωτογραφία:



Αναζήτηση παλαιών ραντεβού του ασθενή

Πατώντας το κουμπί “Search”, ο ασθενής μπορεί να δει **όλα** τα παλαιά ραντεβού του. Επιπλέον, με τη βοήθεια της drop down λίστας, ο ασθενής μπορεί να περιορίσει την αναζήτηση των ραντεβού του κατά **ημερομηνία ραντεβού**, **ειδικότητα του ιατρού** και κατά **ΑΜΚΑ του ιατρού**.

Όταν ο ασθενής πατήσει το δεύτερο κουμπί (*Book an appointment*) εμφανίζεται στην οθόνη η σελίδα *AvailableDoctorAppointments.jsp,* όπως φαίνεται και στην επόμενη φωτογραφία:



Προγραμματισμός νέου ραντεβού από τον ασθενή

Εδώ ο ασθενής επιλέγει τις ημερομηνίες **Starting_date** και **Ending_date**, για να προβληθούν στην οθόνη όλα τα διαθέσιμα ραντεβού που υπάρχουν στο χρονικό διάστημα **[Starting_date , Ending_date]**. Επιπλέον, ο ασθενής μπορεί να περιορίσει την αναζήτηση των ραντεβού που πρόκειται να εμφανιστούν στην οθόνη με την βοήθεια της drop down λίστας. Έτσι, ο ασθενής μπορεί να επιλέξει να εμφανίζονται ραντεβού με ιατρούς συγκεκριμένης ειδικότητας (*Speciality*), είτε με ιατρούς που φέρουν συγκεκριμένο ονοματεπώνυμο (*Full name*) και ΑΜΚΑ (*Doctor AMKA*). Όταν ο ασθενής πατήσει το κουμπί “Search”, εμφανίζονται τα διαθέσιμα ραντεβού με βάση τις ρυθμίσεις που δηλώθηκαν προηγουμένως.

Για παράδειγμα, στην επόμενη φωτογραφία εμφανίζονται τα διαθέσιμα ραντεβού που προκύπτουν με βάση ορισμένες παραμέτρους που εισήγαγε ο χρήστης:

Date	Start time	End time	Doctor AMKA	Doctor name	Doctor surname	Doctor specialty	
22-07-2021	15:15:00	15:45:00	10000000001	Nikolaos	Georgiadis	Pathologist	Book

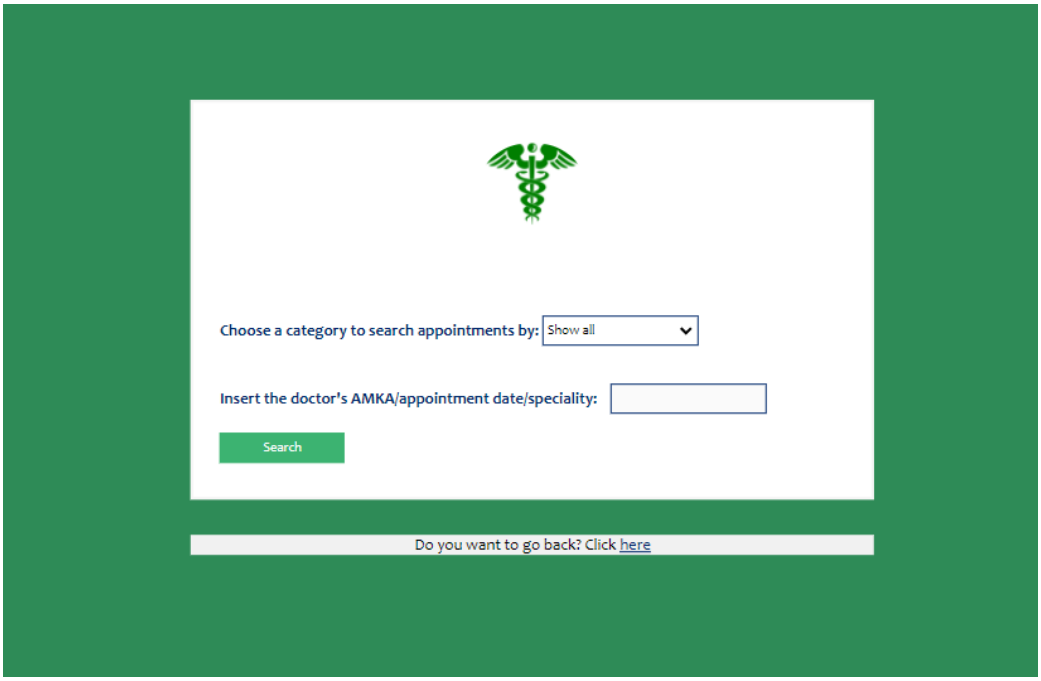
Διαθέσιμα ραντεβού

Αφού ο ασθενής πατήσει το κουμπί book, ουσιαστικά κλείνει το συγκεκριμένο ραντεβού (τα στοιχεία του ραντεβού φαίνονται στην προηγούμενη φωτογραφία). Τότε εμφανίζεται το ακόλουθο μήνυμα:

Thank you for using our web application to book your appointment Name2! Your appointment has been booked on 22-07-2021 at 15:15:00 until 15:45:00(Doctor's AMKA: 10000000001)

Επιτυχές κλείσιμο ενός ραντεβού μεταξύ ασθενούς και ιατρού

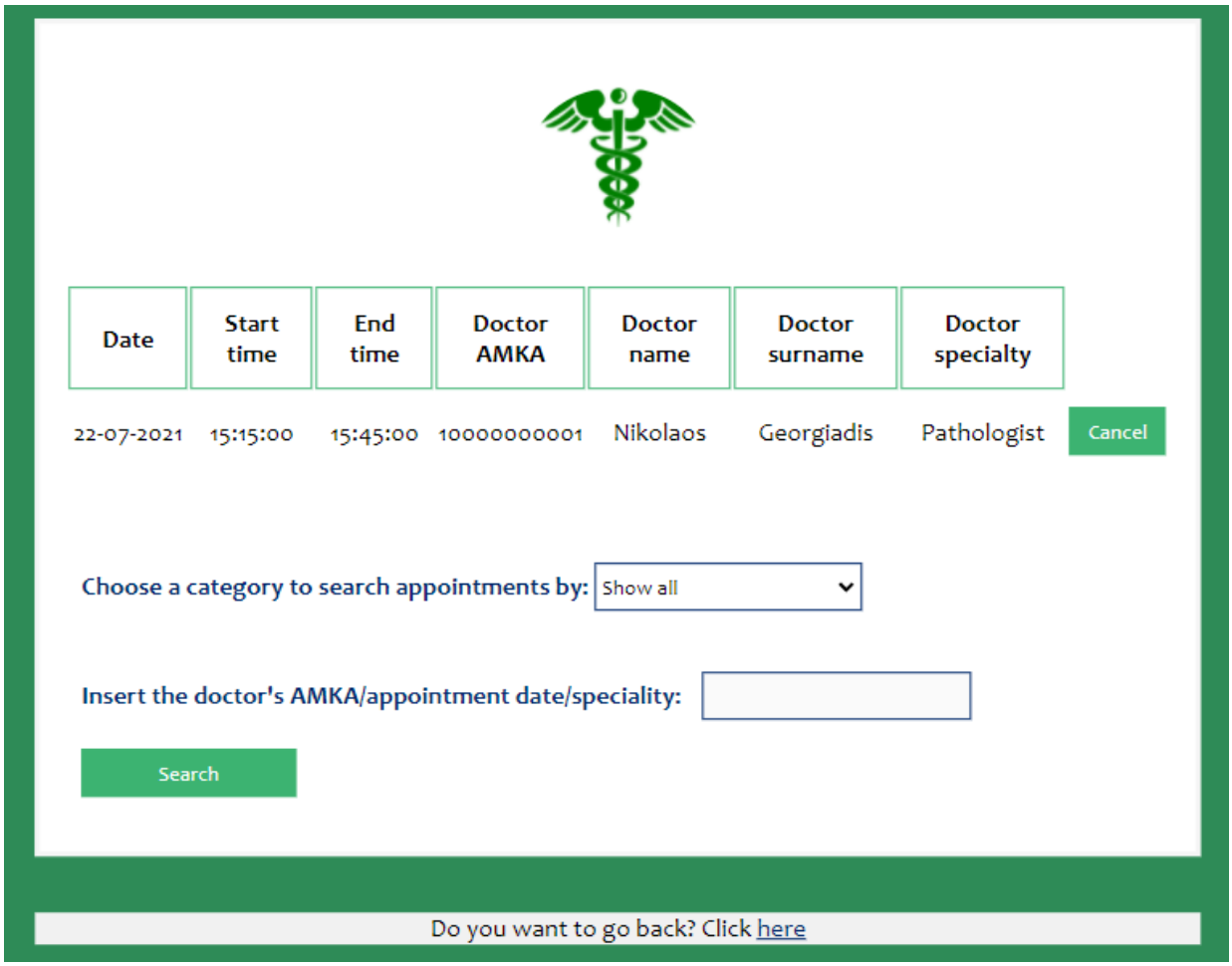
Όταν ο ασθενής πατήσει το κουμπί “Scheduled appointments”, θα μεταφερθεί αυτομάτως στην σελίδα *ScheduledAppointments.jsp*, η οποία εμφανίζεται στην ακόλουθη φωτογραφία:



Προβολή μελλοντικών (προγραμματισμένων) ραντεβού ενός ασθενή

Έτσι, κάθε ασθενής που συνδέεται στο σύστημα, μπορεί να δει όλα τα προγραμματισμένα (μελλοντικά) ραντεβού που έχει κλείσει. Πατώντας το κουμπί “Search”, εμφανίζονται όλα τα ραντεβού του ασθενή που υπάρχουν δηλωμένα στη βάση. Φυσικά, όπως και προηγουμένως, η αναζήτηση αυτή μπορεί να περιοριστεί ως προς τα αποτελέσματά της με την βοήθεια της Drop down λίστας.

Για παράδειγμα, στην επόμενη φωτογραφία φαίνονται διαθέσιμα ραντεβού που έχει ήδη ο ασθενής με username “patient2”:



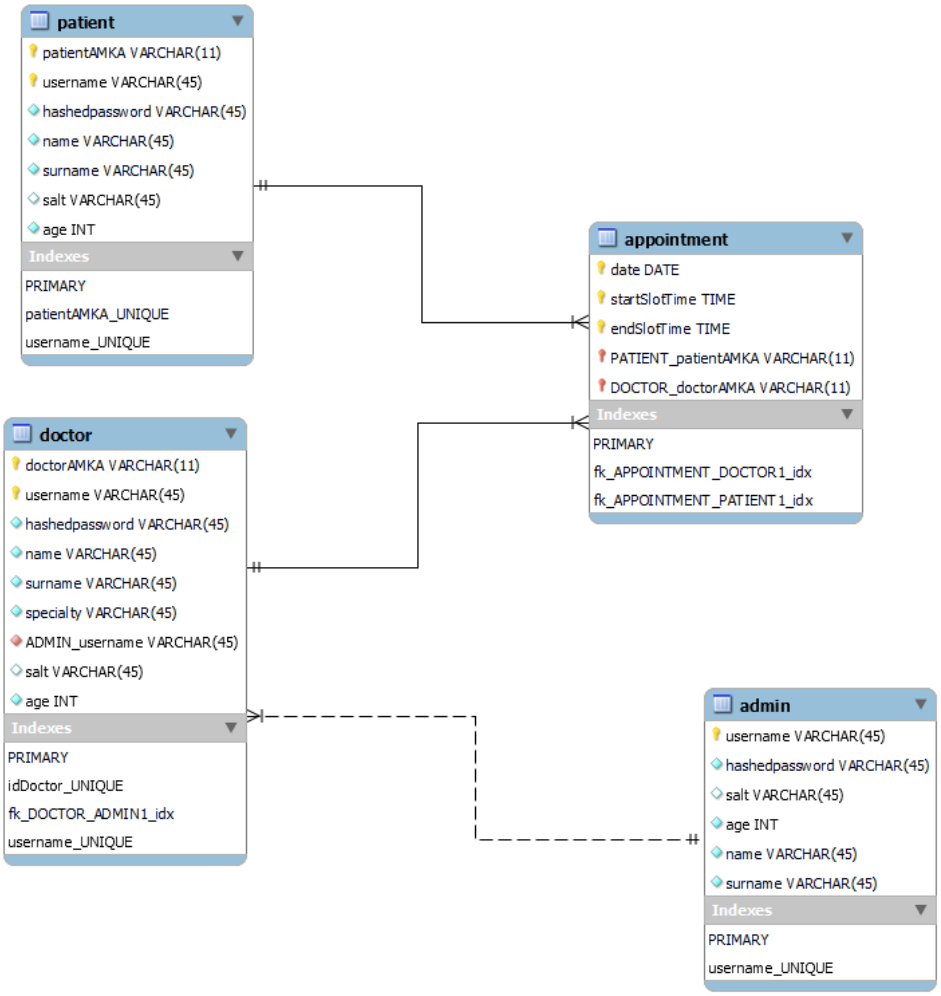
Προγραμματισμένα ραντεβού του ασθενούς patient2

Εάν ο ασθενής το επιθυμεί, μπορεί να πατήσει το κουμπί “Cancel”, για να ακυρώσει ένα συγκεκριμένο ραντεβού που έχει.

Τέλος, το κουμπί “*Logout*” κλείνει την σύνδεση (session) και μεταφέρεται ο έλεγχος του προγράμματος στην σελίδα *login.jsp.*

4 Η βάση δεδομένων

Το σχήμα της βάσης δεδομένων είναι το ακόλουθο:



Σχήμα βάσης δεδομένων στη MySQL(αρχείο app_db.mwb)

Αφού γίνει **forward engineer** το παραπάνω αρχείο, πρέπει να εισαχθεί και ο φάκελος “sql statements” στη MySQL. Οδηγούμαστε στο μενού > Server > Data Import > Import From Dump Project Folder > επιλογή “sql statements” > Start Import

Γενικά

Σχετικά με τα χαρακτηριστικά κάθε πίνακα, μπορούμε να παρατηρήσουμε πως όλοι οι χρήστες έχουν ένα **username**,**hashedpassword**,**name**,**surname**,**age** και **salt**. Το salt, είναι μία σειρά από τυχαίους χαρακτήρες που ορίζουν μοναδικά το hash ενός password. Ένα κοινό password μεταξύ δύο χρηστών, δεν έχει το ίδιο hash καθώς τα δύο αυτά passwords θα έχουν αναμειχθεί το καθένα με διαφορετικό salt πριν κρυπτογραφηθούν. Αυτό αυξάνει την ασφάλεια της εφαρμογής μας (για την δυνατότητα δοκιμής της εφαρμογής έχουμε συμπεριλάβει το αρχείο “passwords memo.txt” που περιέχει τους κωδικούς των χρηστών).

Ο ασθενής και ο ιατρός ξεχωρίζουν από τους admin ως προς τα χαρακτηριστικά τους, καθώς έχουν ένα αποκλειστικό ΑΜΚΑ. Ο ιατρός, επίσης, κρατά και μία πληροφορία για το ποιος admin τον δημιούργησε.

Ο πίνακας των ραντεβού, περιέχει την πληροφορία της ημερομηνίας έναρξης, της ώρας έναρξης και της ώρας λήξης του, όπως επίσης και τα ΑΜΚΑ του ιατρού και του ασθενή που συμμετέχουν στο ραντεβού.

Να σημειωθεί πως έχει εισαχθεί μία dummy εγγραφή ενός ασθενή 0, του οποίου τα χαρακτηριστικά είναι όλα 0. Αυτή η εγγραφή υπάρχει, για να προσδιορίζει στον πίνακα των ραντεβού ότι αν PATIENT_patientAMKA=0, ο ιατρός είναι διαθέσιμος(αλλιώς θα περιέχει το ΑΜΚΑ του ασθενή που έχει κλείσει το ραντεβού).

Τα κλειδιά

Κάθε βάση δεδομένων περιέχει σε κάθε πίνακά της τουλάχιστον ένα πρωτεύων/υποψήφιο κλειδί, το οποίο είναι το χαρακτηριστικό (ή το σύνολο των χαρακτηριστικών) που προσδιορίζουν μοναδικά τις εγγραφές. Στη δική μας βάση δεδομένων, έχουμε τα εξής πρωτεύοντα/υποψήφια κλειδιά:

Στον πίνακα patient έχουμε υποψήφιο κλειδί το username και το patientAMKA. Αυτά τα δύο, προσδιορίζουν μοναδικά έναν ασθενή.

Στον πίνακα doctor έχουμε υποψήφιο κλειδί το username και το doctorAMKA. Αυτά τα δύο, προσδιορίζουν μοναδικά έναν ιατρό.

Στον πίνακα admin έχουμε πρωτεύων κλειδί το username το οποίο προσδιορίζει μοναδικά έναν διαχειριστή.

Στον πίνακα appointment τα πράγματα είναι πιο περίπλοκα. Σκεφτόμαστε ότι, ένας ιατρός ή ένας ασθενής, δεν μπορεί να έχει κλείσει ένα ραντεβού την ίδια μέρα στο ίδιο διάστημα καθώς δεν μπορεί να βρίσκεται σε δύο ραντεβού ταυτόχρονα. Οπότε εδώ, έχουμε δύο υποψήφια κλειδιά, την τετράδα (date,startSlotTime,endSlotTime,PATIENT_patientAMKA) και την τετράδα (date,startSlotTime,endSlotTime,DOCTOR_doctorAMKA) οι οποίες προσδιορίζουν μοναδικά ένα ραντεβού.

Υπάρχουν επίσης και τα λεγόμενα ξένα κλειδιά. Ξένο κλειδί σε έναν πίνακα είναι ένα χαρακτηριστικό που αποτελεί κλειδί σε άλλο πίνακα. Ένα ξένο κλειδί, δεν μπορεί να

παίρνει τιμές που δεν υπάρχουν στο σχετιζόμενο με αυτό πίνακα. Το χαρακτηριστικό ADMIN_username στον πίνακα doctor αποτελεί ξένο κλειδί, καθώς δεν μπορεί να αναφέρεται σε έναν admin που δεν υπάρχει. Τέλος, τα PATIENT_patientAMKA και DOCTOR_doctorAMKA στον πίνακα appointment, αποτελούν και αυτά ξένα κλειδιά που αναφέρονται στον πίνακα patient και doctor αντίστοιχα.

Άλλοι περιορισμοί

Αρχικά, λόγω των ξένων κλειδιών, προκύπτουν μερικοί περιορισμοί. Όταν ένας admin σβήσει έναν ιατρό ή ασθενή, πρέπει να σβηστούν και τα αντίστοιχα ραντεβού που τους περιέχουν. Αυτό επιτυγχάνεται με μία εντολή “on delete cascade” μετά τη προσθήκη της ιδιότητας του ξένου κλειδιού.

Επίσης, όταν σβήνεται ένας admin, μέσω της εντολής “on delete set null” στο ξένο κλειδί ADMIN_username, θέτουμε αυτό το πεδίο null σε όσους ιατρούς έχουν δημιουργηθεί από αυτόν τον admin.

Ακόμη, πρέπει να διασφαλίσουμε ότι ένας ιατρός δεν έχει ίδιο AMKA με έναν ασθενή. Αυτόν τον έλεγχο τον έχουμε υλοποιήσει σε επίπεδο server αυτή τη φορά. Τη στιγμή που ένας χρήστης κάνει register, του εμφανίζεται σχετικό μήνυμα αν το AMKA που χρησιμοποιεί υπάρχει σε άλλο (οποιοδήποτε) χρήστη.

Τέλος, ξανά σε επίπεδο server, έχουμε προσθέσει τον περιορισμό που αποκλείει ένα ραντεβού να συμπέφτει με ένα άλλο(δηλαδή να τελειώνει ή να ξεκινάει μέσα στο διάστημα ενός άλλου) κατά την προσθήκη διαθεσιμότητας ενός ιατρού.

5 Βιβλιογραφικές πηγές

[1] “servlets - How do I call a specific Java method on a click/submit event of a specific button in JSP? - Stack Overflow.” https://stackoverflow.com/questions/14723812/how-do-i-call-a-specific-java-method-on-a-click-submit-event-of-a-specific-butto (accessed Jul. 13, 2021).

[2] “How to change date format in JavaScript - WebArchers.” https://webarchers.com/how-to-change-date-format-in-javascript/ (accessed Jul. 13, 2021).

[3] “javascript - How to change max or min value of input type date from JS - Stack Overflow.” https://stackoverflow.com/questions/16611774/how-to-change-max-or-min-value-of-input-type-date-from-js (accessed Jul. 13, 2021).

[4] “Java add days to Date and LocalDateTime - HowToDoInJava.” https://howtodoinjava.com/java/date-time/add-days-to-date-localdatetime/ (accessed Jul. 13, 2021).

[5] “java - How to get all Session values and names? - Stack Overflow.” https://stackoverflow.com/questions/33368963/how-to-get-all-session-values-and-names (accessed Jul. 13, 2021).

[6] “java - Converting byte array containing ASCII characters to a String - Stack Overflow.” https://stackoverflow.com/questions/18583279/converting-byte-array-containing-ascii-characters-to-a-string (accessed Jul. 13, 2021).

[7] “date - MySQL: How to select records for this week? - Stack Overflow.” https://stackoverflow.com/questions/20120693/mysql-how-to-select-records-for-this-week (accessed Jul. 13, 2021).