Ελληνικό Μεσογειακό Πανεπιστήμιο

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη εφαρμογής συστήματος διαχείρισης ιδιωτικής κλινικής με αποθήκευση δεδομένων σε NoSQL βάση δεδομένων

Ονοματεπώνυμο φοιτητή**: Καλκάνης Κυριάκος**

Αριθμός μητρώου **: 4407**

Επιβλέπων καθηγητής : **Παπαδάκης Νικόλαος**

Ηράκλειο Κρήτης

2020

Ευχαριστίες

Αισθάνομαι την ανάγκη να ευχαριστήσω θερμά όλους όσους με βοήθησαν και με στήριξαν έτσι ώστε να ολοκληρώσω την πτυχιακή μου εργασία. Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Νίκο Παπαδάκη για την καθοδήγηση αλλά και την ενθάρρυνση κατά την υλοποίηση της πτυχιακής μου εργασίας.

Επίσης , θα ήθελα να ευχαριστήσω την οικογένεια μου για την στήριξη, την υπομονή και την εμπιστοσύνη που μου έδειξαν κατά τη διάρκεια των σπουδών μου βοηθώντας με αυτόν τον τρόπο όχι μόνο στην υλοποίηση της πτυχιακής μου εργασίας αλλά και στο να εκπληρώσω τους στόχους μου.

Θα ήθελα να ευχαριστήσω την κοπέλα μου Θεοδώρα Δημογέροντα για την αμέριστη, συμπαράσταση και κατανόηση που μου έδειξε καθ'όλη τη διάρκεια των σπουδών μου αλλά πάνω απ'όλα για την υποστήριξη και βοήθεια που μου προσέφερε βοηθώντας με να πετύχω τους στόχους μου.

Τέλος θα ήθελα να ευχαριστήσω τους φίλους και συναδέλφους μου για την υποστήριξη και συμπαράσταση που μου έδιναν καθ' όλη την διάρκεια των σπουδών μου.

Abstract

Private clinic management system is an desktop application implemented in JAVA and MongoDB (NoSQL) that is able to organize, manage and simplify all functions of a private clinic. In more detail, this application will enable users such as administrator, doctor , receptionist and patients to manage their functions. When users access their account they will be able to do specific functions depending on the rights given to them. In short, some of the functions are to add, delete, update(any user),signup, login , make an appointment (add, delete and update) it, calculate the bill for each examination and generally to organize all the functions of the system . We will see all the functions of each user separately below.

Σύνοψη

Το σύστημα διαχείρισης ιδιωτικής κλινικής είναι μια εφαρμογή desktop υλοποιημένη σε JAVA και MongoDB (NoSQL) η οποία μπορεί να οργανώσει, διαχειριστεί και να απλοποιεί όλες τις λειτουργίες μιας ιδιωτικής κλινικής.

Πιο αναλυτικά, η παρούσα εφαρμογή θα επιτρέπει στους χρήστες οι οποίοι είναι γιατροί , ασθενείς , διαχειριστής και reception να διαχειρίζονται τις λειτουργίες τους.

Όταν οι χρήστες αποκτούν πρόσβαση στο λογαριασμό τους, θα μπορούν να κάνουν συγκεκριμένες λειτουργίες ανάλογα με τα δικαιώματα που τους έχουν δοθεί.

Συνοπτικά, κάποιες από τις λειτουργίες είναι η προσθήκη, διαγραφή και ενημέρωση οποιουδήποτε χρήστη, η εγγραφή και σύνδεση στην πλατφόρμα , η δημιουργία ραντεβού ανάμεσα σε γιατρό και ασθενείς καθώς και η διαγραφή και ενημέρωση του, ο υπολογισμός του λογαριασμού από τη ρεσεψιόν ανάλογα τις εξετάσεις που υποβλήθηκε ο κάθε ασθενής. Θα δούμε αναλυτικότερα όλες τις λειτουργίες του κάθε χρήστη παρακάτω.

Πίνακας Περιεχομένων

Πίνακας εικόνων :

Πίνακες

1. Εισαγωγή

Η εκπόνηση της πτυχιακής εργασίας είναι μεγάλης σημασίας για έναν φοιτητή αφού του επιτρέπει να εξασκηθεί πάνω στον αντικείμενο που έχει επιλέξει να εργασθεί και με αυτόν τον τρόπο γίνεται μια προσομοίωση στο πραγματικό εργασιακό περιβάλλον στο οποίο θα δραστηριοποιηθεί. Επίσης, δίνεται η ευκαιρία στον σπουδαστή να πειραματιστεί με τις πρακτικές και τις νέες τεχνολογίες του κλάδου του ούτως ώστε να αποκτήσει γνώσεις και ταυτόχρονα εμπειρία για να είναι έτοιμος να εργασθεί πάνω σε αυτόν τον τομέα.

1.1 Περίληψη

Το σύστημα διαχείρισης ιδιωτικής κλινικής είναι μια εφαρμογή desktop που έχει σαν σκοπό την απλοποίηση, την διαχείριση και την οργάνωση των λειτουργιών μιας ιδιωτικής κλινικής. Πιο συγκεκριμένα έχουμε τέσσερις οντότητες που είναι ο administrator , γιατροί, receptionist και ασθενείς.

Οι ασθενείς μπορούν να συνδεθούν στην πλατφόρμα εάν έχουν λογαριασμό, εάν δεν έχουν μπορούν να δημιουργήσουν έναν πολύ γρήγορα με τα προσωπικά τους στοιχεία. Όταν αποκτήσουν πρόσβαση στο προφίλ τους μπορούν να κλείσουν ραντεβού με έναν συγκεκριμένο γιατρό μια συγκεκριμένη ώρα και ημερομηνία ανάλογα με την εξέταση που θέλουν να κάνουν. Το κόστος της εξέτασης υπολογίζεται ανάλογα την εξέταση από την reception η οποία στέλνει τον λογαριασμό στον ασθενή όπου μπορεί να τον δει από το προφίλ του.

Οι γιατροί μπορούν να συνδεθούν στο σύστημα (μόνο ο admin μπορεί να τους δημιουργήσει λογαριασμό για προφανής λόγους). Όταν συνδεθούν στο προφίλ τους μπορούν να προσθέσουν ,τροποποιήσουν και να δουν τα ραντεβού τους.

Η reception όταν συνδεθεί στο σύστημα μπορεί να προσθέσει, να διαγράψει και να τροποποιήσει τα προσωπικά στοιχεία των ασθενών για τυχών αλλαγές καθώς και τα ραντεβού που αφορούν τους ασθενείς. Επίσης, μπορεί να δώσει κρεβάτι σε ασθενή αν είναι απαραίτητο καθώς και να δει τους λογαριασμούς όλων των εξετάσεων των ασθενών.

Ο administrator μπορεί να συνδεθεί και να προσθέσει, να διαγράψει , να τροποποιήσει όλους τους ασθενείς , γιατρούς , receptionists και να δει όλα τα προσωπικά τους στοιχεία .

Τέλος, η εφαρμογή δημιουργήθηκε με τη βοήθεια της **JAVA** για τη δημιουργία των διεπαφών (**JAVA Swing**) καθώς και τον προγραμματισμό τους , της βάσης δεδομένων **MongoDB Atlas** η οποία είναι μια **NoSQL** online βάση δεδομένων για την αποθήκευση των αρχείων κάθε χρήστη όπως επίσης και το **Jongo** που είναι μια βιβλιοθήκη η οποία επιτρέπει στον χρήστη να γράφει κώδικα σε mongoDB στην JAVA όπως θα έγραφε στο shell.

1.2 Κίνητρο για την Διεξαγωγή της Εργασίας – Στόχοι

Δεν υπάρχει αμφιβολία πως η οργάνωση μιας ιδιωτικής κλινικής είναι δύσκολη αφού πρέπει να συντονιστεί όλο το προσωπικό της για να υπάρξει το μέγιστο δυνατό αποτέλεσμα. Επίσης ένα μεγάλο πρόβλημα είναι ο μεγάλος φόρτος κλήσεων στις κλινικές μεταξύ πελατών και γραμματείας για να κλείσουν ραντεβού εξετάσεων καθώς ίσως και οι ουρές που δημιουργούνται για το λόγο αυτό.

Ένα άλλο σημαντικό πρόβλημα είναι η έλλειψη ενός πληροφοριακού συστήματος το οποίο είναι ικανό να διατηρεί αρχεία – πληροφορίες σχετικά με τους ασθενείς καθώς και το προσωπικό της κλινικής είτε αυτό είναι για προσωπικά στοιχεία είτε είναι για ημερομηνίες-ώρες ραντεβού είτε είναι το συνολικό ποσό πληρωμής των ασθενών για τις εξετάσεις που υποβλήθηκαν.

Τα παραπάνω προβλήματα μπορούν να αντιμετωπιστούν με τη βοήθεια της πτυχιακής εργασίας που μου έχει ανατεθεί η οποία έχει ως σκοπό την οργάνωση καθώς και την απλούστευση των ενεργειών μιας ιδιωτικής κλινικής ούτως ώστε να λειτουργεί άρτια και ως αποτέλεσμα οι παραπάνω λειτουργίες να μην είναι χρονοβόρες.

1.3 Δομή Εργασίας

Η παρούσα πτυχιακή εργασία είναι δομημένη σε πέντε κεφάλαια στα οποία γίνεται αναφορά στα προβλήματα που έρχεται να λύσει η παρούσα εφαρμογή και στις τεχνολογίες που χρησιμοποιήθηκαν για την σχεδίαση και ανάπτυξη της εφαρμογής. Για την καλύτερη εμπειρία του αναγνώστη υα υπάρχουν παραπομπές σε έρευνες και κείμενα άλλων τα οποία θα αναφέρονται μια φορά στο εσωτερικό του κειμένου και μια φορά στο τέλος της εργασίας στην Βιβλιογραφία με αρίθμηση και link με περαιτέρω λεπτομέρειες επί του θέματος.

Όσον αφορά τα κεφάλαια , στο **Κεφάλαιο 1** θα γίνει μια εισαγωγή στην εφαρμογή , θα αναφερθούν συνοπτικά τα χαρακτηριστικά της και θα αναφερθούν επίσης τα προβλήματα που καλείται να επιλύσει.

Στο επόμενο κεφάλαιο , δηλαδή στο **Κεφάλαιο 2** θα γίνει αναφορά στην μεθοδολογία υλοποίησης και ανάλυσης της εργασίας καθώς και σε τεχνολογίες , αλγορίθμους, μοντέλα που ακολουθήθηκαν για την επίλυση των προβλημάτων που προέκυψαν.

Στο **Κεφάλαιο 3,** γίνεται μια περαιτέρω ανάλυση στις τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής καθώς και μια μικρή εισαγωγή στην εφαρμογή.

Στο **Κεφάλαιο 4** , γίνεται μια λεπτομερής ανάλυση της εφαρμογής. Εξετάζεται η λειτουργικότητα της και η ευχρηστία της. Στα παραπάνω συμβάλουν screenshots από τα κομμάτια της εφαρμογής και ακολουθεί πάντα κείμενο ανάλογα με τα σενάρια χρήσης που θα εξεταστούν.

Τέλος, στο **Κεφάλαιο 5**, αναφέρονται τα οφέλη της εφαρμογής στους χρήστες που τη χρησιμοποιούν καθώς και επιπρόσθετες λειτουργίες στην εφαρμογή για μελλοντική επέκταση.

2. Μεθοδολογία Υλοποίησης

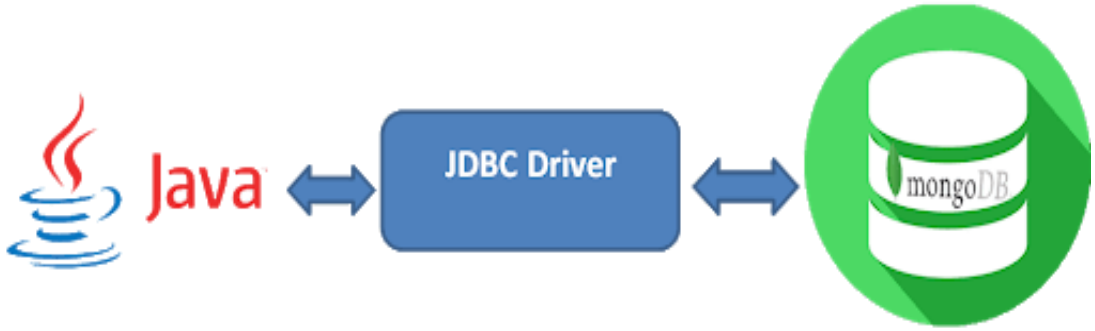
Σε αυτή την ενότητα θα γίνει μια προσπάθεια περιγραφής της μεθοδολογίας που ακολουθήθηκε για την εκπόνηση της παρούσας πτυχιακής εργασίας καθώς και αναφορά αλλά και λύση στα προβλήματα που προέκυψαν κατά τη διάρκεια υλοποίησης και σχεδιασμού της πλατφόρμας.

Επιπλέον, θα αναφερθούν και επεξηγηθούν οι τεχνολογίες που χρησιμοποιήθηκαν εις βάθος για την εκπόνηση της πτυχιακής εργασίας . Τέλος , θα δούμε το πώς θα γίνει η οργάνωση του συστήματος διαχείρισης ιδιωτικής κλινικής και έπειτα θα γίνει περαιτέρω ανάλυση στα ζητήματα που προκύπτουν για να γίνει η οργάνωση αυτή.

2.1 Μέθοδος Ανάλυσης

Ο σκελετός της εφαρμογής αποτελείται από τρία(3)βασικά χαρακτηριστικά τα οποία θα αναφερθούν επιγραμματικά παρακάτω και θα αναλυθούν εις βάθος στη συνέχεια της ενότητας σαν επιμέρους υποενότητες αυτής .Πιο συγκεκριμένα θα γίνει αναφορά στον τρόπο με τον οποίο λειτουργούν , τα χαρακτηριστικά τους (θετικά και αρνητικά) τους , τις ανάγκες που συντέλεσαν στη δημιουργία τους κ.α . Παρακάτω αναφέρονται τα κύρια χαρακτηριστικά της εργασίας.

- Η Desktop Εφαρμογή
- Ο JDBC Driver
- Η βάση δεδομένων



Εικόνα 1 : Κύρια χαρακτηριστικά εργασίας

2.1.1 Η Desktop Εφαρμογή

Με τον όρο desktop εφαρμογή εννοούμε τα προγράμματα τα οποία είναι εγκατεστημένα στον προσωπικό μας υπολογιστή και είναι φτιαγμένα για κάποια συγκεκριμένη εργασία. Είναι πιο γρήγορες σε σχέση με τις web εφαρμογές καθώς και τις mobile εφαρμογές . Αλληλεπιδρούν και διασυνδέονται με το λειτουργικό σύστημα , αυτό είναι θετικό διότι εκμεταλλεύονται τις δυνατότητες του και επίσης αξιοποιούν στο έπακρο τους πόρους του hardware. Οι Desktop εφαρμογές μπορούν να λειτουργούν είτε διαδικτυακά μέσω ενός web browser είτε εκτός δικτύου . Επίσης μπορούν να επικοινωνούν με άλλες εφαρμογές ,είτε με εφαρμογές εκτός δικτύου είτε με εφαρμογές στο διαδίκτυο. Τέλος μπορούν να επικοινωνήσουν με τοπικές βάσεις δεδομένων είτε με απομακρυσμένες στο διαδίκτυο.

Μία από τις κύριες γλώσσες προγραμματισμού για την ανάπτυξη εφαρμογών στις μέρες μας είναι η Java.

2.1.1.1 Γλώσσα προγραμματισμού JAVA

Η γλώσσα προγραμματισμού JAVA[1] κατασκευάστηκε από την εταιρία Sun Microsystems , εταιρία πληροφορικής και είναι μια αντικειμενοστραφείς γλώσσα προγραμματισμού . Το βασικό χαρακτηριστικό της JAVA είναι η ανεξαρτησίας της πλατφόρμας και λειτουργικού συστήματος το οποίο είναι ένα βασικό πλεονέκτημα έναντι άλλως γλωσσών. Τα προγράμματα τα οποία έχουν γραφτεί με τη βοήθεια της γλώσσας προγραμματισμού JAVA μπορούν να τρέξουν σε Unix,Windows,Linux και Mac χωρίς να χρειαστεί την οποιαδήποτε αλλαγή ο κώδικας των εφαρμογών ή να χρειαστεί να ξαναγίνει μεταγλώττιση, με τον ίδιο τρόπο ακριβώς για όλα τα λειτουργικά συστήματα . Για να επιτευχθεί όμως το παραπάνω δημιουργήθηκε η **εικονική μηχανή της JAVA**[2].

2.1.1.2 Εικονική μηχανή της JAVA

Όταν γίνει η εγγραφή ενός προγράμματος σε γλώσσα JAVA , γίνεται η μεταγλώττιση του στη συνέχεια από τον μεταγλωττιστή javac και έπειτα ο μεταγλωττιστής δημιουργεί μια σειρά από αρχεία με την κατάληξη .class(bytecode) . Η bytecode μορφή είναι η μορφή που έχει ο source code όταν γίνει η μεταγλώττιση του. Το JVM (Java Virtual Machine) πρέπει να είναι εγκατεστημένο σε κάθε μηχάνημα που θέλει να τρέξει ένα πρόγραμμα γραμμένο σε Java αφού αυτό είναι που διαβάζει τα αρχεία που δημιουργήθηκαν από τον μεταγλωττιστή javac προηγουμένως. Έπειτα γίνεται η μετάφραση των αρχείων σε γλώσσα μηχανής έτσι ώστε να μπορεί να τα διαβάζει τόσο ο επεξεργαστής όσο και το λειτουργικό σύστημα για να γίνει η εκτέλεση τους.

Επίσης για τη βελτίωση ταχύτητας , υπάρχουν πιο σύγχρονες εφαρμογές που μεταγλωττίζουν από πριν τα bytecode απευθείας σε γλώσσα μηχανής. Χωρίς τα

προαναφερθέντα δεν θα ήταν εφικτή η εκτέλεση προγραμμάτων JAVA. Εδώ είναι σημαντικό να τονίσουμε πως η Java Virtual Machine είναι άμεσα εξαρτώμενη από την πλατφόρμα , δηλαδή για κάθε λειτουργικό σύστημα υπάρχει διαφορετική έκδοση JVM αφού υπάρχουν διαφορές ανάμεσα στις αρχιτεκτονικές των επεξεργαστών και των λειτουργικών συστημάτων.

Τέλος, σε όποια ενέργεια και αν θέλει ο χρήστης να προβεί αυτή γίνεται μέσω της JVM η οποία είναι υπεύθυνη για την αλληλεπίδραση χρήστη-υπολογιστή και σαν αποτέλεσμα σε περίπτωση που το σύστημα θα κατάρρεε από κακόβουλο κώδικα ή από κώδικα του προγραμματιστή η JVM δεν τον αφήνει να εκτελεστεί.

2.1.1.3 Garbage Collector της JAVA

Ο garbage collector[3] της JAVA χρησιμοποιείται για να απελευθερώσει χώρο στη μνήμη από δεδομένα που η χρήση τους δεν απαιτείται. Η JVM είναι αυτή που ενημερώνει τον garbage collector να απελευθερώσει τη μνήμη μόλις δει ότι ο σωρός (heap) τείνει να γεμίσει. Με αυτόν τον τρόπο ο προγραμματιστής δεν χρειάζεται να προβεί σε κάποια περαιτέρω ενέργεια και έτσι αποφεύγονται σφάλματα δεικτών αφού η μνήμη έχει ποια απελευθερωθεί και είναι σίγουρο πως κανένας άλλος δείκτης δεν θα πάει να γράψει σε χώρο άλλου δείκτη που δεν του ανήκει.

2.1.1.4 Η Java σε επίπεδο κώδικα

Η Java είναι μία γλώσσα ειδικά σχεδιασμένη για την ανάπτυξη εφαρμογών η οποία χρησιμοποιεί αντικείμενα. Τα αντικείμενα αυτά είναι δομές πληροφορίας και ανήκουν σε κλάσεις. Οι κλάσεις αυτές έχουν δικά τους χαρακτηριστικά , setters και getters για να δώσουν και να πάρουν την τιμή των χαρακτηριστικών αντίστοιχα, απλές συναρτήσεις και μεθόδους. Επιπλέον , με τη βοήθεια της ενθυλάκωσης τα χαρακτηριστικά και οι κλάσεις μπορούν να είναι ορατά από άλλα σημεία του κώδικα η όχι με τα words public και private αντίστοιχα καθώς και να μην μπορούν να αναφερθούν σε αυτά από άλλα πακέτα άλλων κλάσεων με το word protected. Επίσης η Java είναι μια γλώσσα που επιτρέπει την κληρονομικότητα με το word extend όπου η οντότητα child μπορεί να κληρονομήσει χαρακτηριστικά και μεθόδους από την κλάση parent.

2.1.2 Βάση Δεδομένων

Η έννοια των βάσεων δεδομένων[4] αναφέρεται σε ένα εργαλείο για τη συλλογή και οργάνωση δεδομένων τα οποία είναι δυνατόν να αποθηκευτούν σε αυτή καθώς και να ανακτηθούν αλλά και να ανανεωθούν όταν αυτό είναι επιθυμητό. Η οργάνωση και η διαχείριση μεγάλου όγκου δεδομένων ήταν ένα από τα πρώτα βήματα του διαδικτύου, οι βάσεις δεδομένων και τα συστήματα διαχείρισης βάσεων δεδομένων(ΣΔΒΔ)

πρωτοεμφανίστηκαν στα τέλη του 1960.

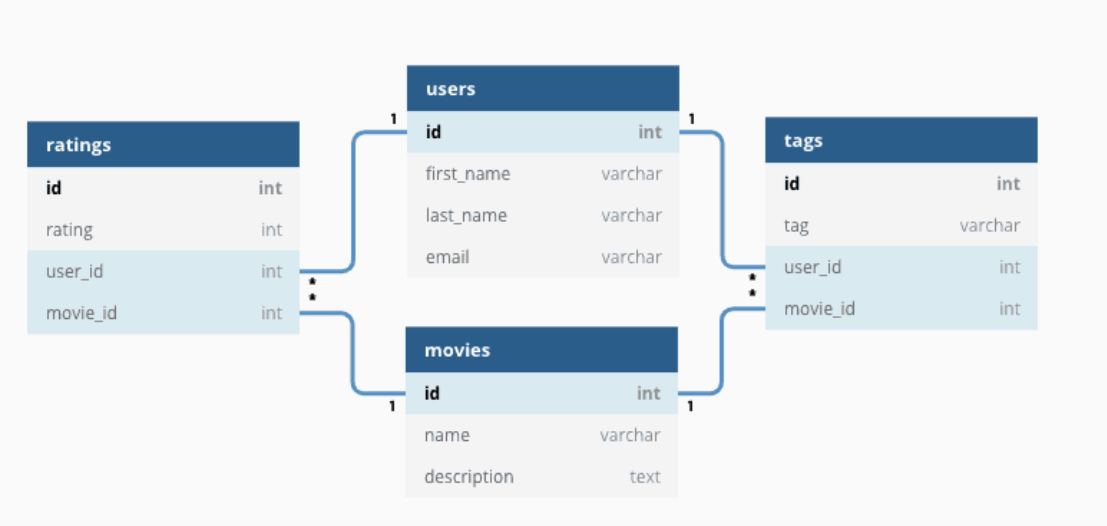
Οι βάσεις δεδομένων χωρίζονται σε δύο είδη :

- Σχεσιακές βάσεις δεδομένων (relational model of data)
- Μη σχεσιακές βάσεις δεδομένων (NoSQL- document oriented)

2.1.2.1 Σχεσιακές Βάσεις Δεδομένων

Η δημιουργία της σχεσιακής βάσης δεδομένων[5] έγινε το 1970 από τον Edgar F.Codd. Με τον όρο της σχεσιακής βάσης δεδομένων αναφερόμαστε σε μια συλλογή δεδομένων σε αυστηρώς συσχετιζόμενους πίνακες ενώ παρέχει έναν μηχανισμό για add,delete,update καθώς και άλλων πιο πολύπλοκων ενεργειών στα δεδομένα. Η σχεσιακή βάση δεδομένων αποσκοπεί στην αποθήκευση και ανάκτηση των δεδομένων. Η SQL είναι η πιο ευρέως γνωστή γλώσσα που χρησιμοποιείται στις βάσεις δεδομένων η οποία είναι υπεύθυνη για τα ερωτήματα που υποβάλλει ο χρήστης ή κάποιο λογισμικό στη βάση δεδομένων ούτως ώστε να ανακτήσει την πληροφορία που ζητά.

Κάθε χρήστης έχει διαφορετικά δικαιώματα που του έχουν δοθεί από τον administrator της βάσης δεδομένων, έτσι μόνο οι έχοντες τα δικαιώματα μπορούν να τροποποιήσουν τη βάση δεδομένων(add, delete, update) καθώς και να ζητήσουν ερωτήματα προς τη βάση δεδομένων. Κάποια από τα πιο γνωστά συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων είναι οι Oracle, MySQL και PostgreSQL.

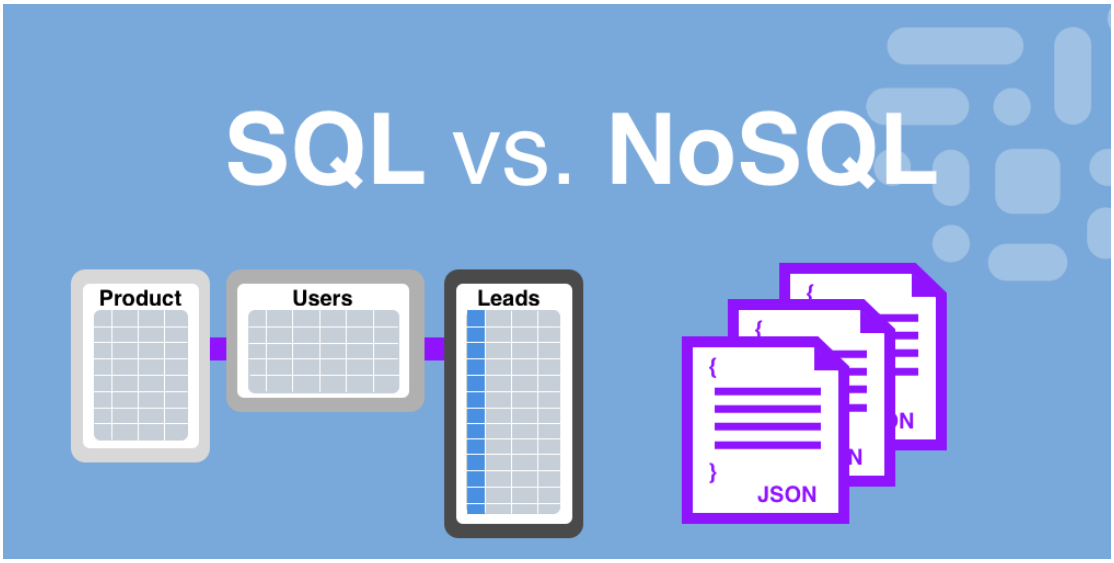


Εικόνα 2: Σχεσιακή Βάση Δεδομένων

Όπως παρατηρούμε στο παραπάνω σχήμα έχουμε τέσσερις(4) οντότητες οι οποίες είναι οι ratings, users, movies και tags οι οποίες αποτελούνται από ένα πίνακα. Η κάθε μια έχει το δικό της μοναδικό πρωτεύον κλειδί(primary key) και τα δικά της πεδία. Οι πίνακες συσχετίζονται μεταξύ τους με τη βοήθεια των ξένων κλειδιών για να αναφερθούν στον πίνακα που θέλουν και να πάρουν πληροφορία

2.1.2.2 NoSQL Βάσεις Δεδομένων (Μη σχεσιακές)

Μια NoSQL [6] βάση δεδομένων παρέχει τον μηχανισμό για την αποθήκευση και ανάκτηση της πληροφορίας αλλά δεν έχει τις εξαρτήσεις που έχει μια σχεσιακή βάση δεδομένων. Οι NoSQL βάσεις δεδομένων υπάρχουν από τα τέλη του 1960 αλλά το όνομα τους διατυπώθηκε μόνο στις αρχές του 21ου αιώνα για τις ανάγκες των εφαρμογών Web 2.0 [7] . Χρησιμοποιούνται όταν έχουμε εφαρμογές μεγάλου όγκου δεδομένων πραγματικού χρόνου. Κάποιες από τις πιο γνωστές NoSQL βάσεις δεδομένων είναι οι MongoDB[8], Redis, Couchbase και AmazonDynamoDB. Τα δεδομένα στις NoSQL βάσεις δεδομένων φυλάσσονται σε ξεχωριστά αρχεία μορφής JSON χωρίς εξαρτήσεις το ένα με το άλλο κάτι που καθιστά τις NoSQL βάσεις δεδομένων πάρα πολύ γρήγορες στην ανάκτηση πληροφορίας. Επίσης, το κάθε ένα αρχείο μπορεί να έχει ανεξάρτητο αριθμό πεδίων –τιμών κάτι που καθιστά το σχήμα των NoSQL βάσεων δεδομένων δυναμικό.



Εικόνα 3 : SQL vs NoSQL

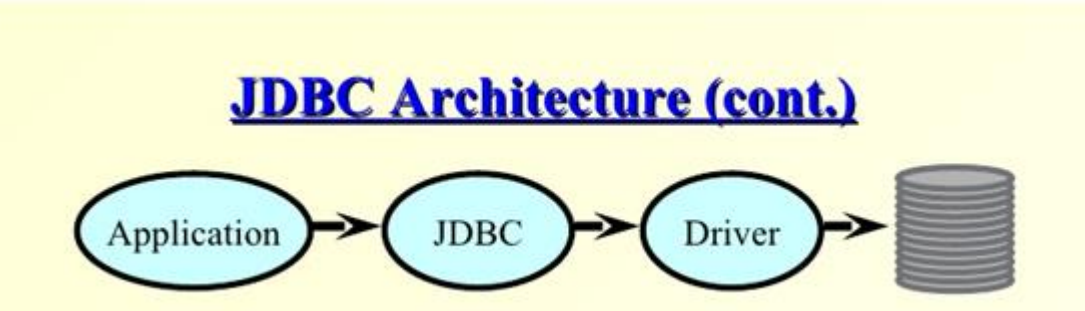
Πολλοί είναι αυτοί που επιμένουν ότι οι σχεσιακές βάσεις δεδομένων είναι ένας καλύτερος τρόπος οργάνωσης δεδομένων από τις NoSQL βάσεις δεδομένων και επίσης πολλοί επιμένουν για το αντίστροφο. Η αλήθεια είναι πως δεν υπάρχει νικητής και είναι στο χέρι του προγραμματιστή να αποφασίσει ποια βάση δεδομένων θα χρησιμοποιήσει ανάλογα με τις εκάστοτε ανάγκες τις εφαρμογής για την οποία προορίζεται η βάση δεδομένων.

2.1.3 Ο JDBC driver

Ένα πρόγραμμα οδήγησης JDBC driver[9] είναι ένα API[10] το οποίο επιτρέπει την σύνδεση και την αλληλεπίδραση ενός προγράμματος-εφαρμογής Java με μια βάση δεδομένων. Επίσης, παρέχει μεθόδους για query σε μια βάση δεδομένων. Η Sun Microsystems κυκλοφόρησε το JDBC το 1997. Η γέφυρα JDBC – to- ODBC[11] επιτρέπει συνδέσεις από οποιαδήποτε JDBC σε ODBC πηγή δεδομένων στο περιβάλλον της JVM. Οι κλάσεις του JDBC παρέχονται από τα packages της Java: java.sql και javax.sql.

2.1.3.1 Λειτουργία JDBC driver

Το JDBC υποστηρίζει την ύπαρξη πολλαπλών εφαρμογών οι οποίες μπορούν να χρησιμοποιούνται από την ίδια την εφαρμογή. Επίσης, είναι εγκατεστημένο μόνο στην πλευρά του πελάτη και η δουλειά του είναι να μετατρέπει αιτήματα από εφαρμογές της Java σε πρωτόκολλα όπου το DBMS μπορεί να κατανοήσει. Παρέχει ένα μηχανισμό για να φορτώνει δυναμικά τα σωστά πακέτα της Java και να τα καταχωρεί στο JDBC Driver Manager. Το Driver Manager χρησιμοποιείται για την ίδρυση των συνδέσεων JDBC. Παρέχει μεθόδους με τις οποίες τρέχει ερωτήματα σε SQL[12] αλλά και για insert, update και delete.



Εικόνα 4: JDBC Architecture

Αρχιτεκτονική JDBC
Ο κώδικας Java καλεί την JDBC βιβλιοθήκη
Η JDBC βιβλιοθήκη φορτώνει τον JDBC Driver
Ο Driver επικοινωνεί με μια συγκεκριμένη βάση δεδομένων
Μια εφαρμογή μπορεί να λειτουργήσει με διάφορες βάσεις δεδομένων χρησιμοποιώντας όλα τα αντίστοιχα προγράμματα οδήγησης

Πίνακας 1: JDBC Architecture

2.2 Μέθοδος Ανάπτυξης

Σύμφωνα με τα όσα αναλύθηκαν στις παραπάνω ενότητες δηλαδή τις ενότητες 2.1.1 , 2.1.2 και 2.1.3 καθώς και με τις απαιτήσεις τις εφαρμογής, έχουν επιλεχθεί οι τεχνολογίες οι οποίες θεωρώ πως είναι ιδανικές για την εν λόγω πτυχιακή εργασία .

Η MongoDB Atlas[13] βάση δεδομένων είναι ιδανική αφού είναι δυναμική και τα δεδομένα οργανώνονται σε ξεχωριστά αρχεία και η ανάκτηση τους για το λόγω αυτό είναι πολύ γρήγορη.

Ο mongo-java-driver-3.12.0 είναι μια αρκετά απλή αλλά και καλή λύση για τη σύνδεση της εν λόγω εφαρμογής με τη βάση δεδομένων καθώς και για να τρέχει να ερωτήματα τα οποία υποβάλλει ο χρήστης – εφαρμογή.

Τέλος, η εφαρμογή θα υλοποιηθεί με τη βοήθεια της Java για τον προγραμματισμό τον components που παρέχει το Java Swing[14] καθώς και για τη σύνδεση με τη βάση δεδομένων της MongoDB.

3. Σχέδιο Δράσης

Σύμφωνα με τα όσα αναλύθηκαν παραπάνω έχουν αποφασιστεί οι τεχνολογίες που θα χρησιμοποιηθούν για την εν λόγω εργασία . Οι επιλογές αυτές δεν είναι μοναδικές και κάλλιστα θα μπορούσε για παράδειγμα για την υλοποίηση της εφαρμογής να χρησιμοποιηθεί μια άλλη γλώσσα προγραμματισμού. Με γνώμονα τις τάσεις των νέων τεχνολογιών που υπάρχουν οι επιλογές για την υλοποίηση της εργασίας μου φάνηκαν οι ιδανικές αφού εκτός του ότι ταιριάζουν απόλυτα στις ανάγκες της εργασίας θα με βοηθήσουν να εμπλουτίσω τις γνώσεις μου με τις εν λόγω νέες τεχνολογίες της εποχής μας.

3.1 Σχεδιασμός και υλοποίηση συστήματος

Η επιλογή της γλώσσας προγραμματισμού για την κατασκευή του project είναι πολύ σημαντική διότι θα είναι ο σκελετός της εφαρμογής και από άποψη προγραμματισμού και από άποψη δημιουργίας των διεπαφών και προγραμματισμού των επιμέρους components.

Επίσης , η επιλογή της NoSQL βάσης δεδομένων που θα χρησιμοποιηθεί στην εν λόγω εργασία είναι πολύ σημαντική αφού όλα θα γίνονται δυναμικά γλιτώνοντας έτσι την εφαρμογή από την αναμονή που θα είχε διαφορετικά αφού όλα θα γίνονται πιο γρήγορα αφού ότι ζητείται θα υπάρχει σε ξεχωριστό αρχείο χωρίς εξαρτήσεις κι έτσι δεν θα πρέπει να γίνεται αναζήτηση σε άλλο collection[15] της βάσης δεδομένων.

Τέλος , η επιλογή του JDBC driver δεν είναι κάτι εύκολο αφού θα πρέπει να βρεθεί ο κατάλληλος driver για τη σύνδεση της εφαρμογής με τη βάση δεδομένων καθώς και τη διαρκή επικοινωνία τους.

3.1.1 Σχεδιασμός και υλοποίηση Βάσης δεδομένων ,GUI και JDBC Driver

Αρχικά , η βάση δεδομένων που αποφάσισα να χρησιμοποιήσω είναι η MongoDB Atlas. Η επιλογή αυτή βασίστηκε στο ότι η MongoDB Atlas είναι μια NoSQL βάση δεδομένων η οποία αποθηκεύει δεδομένα online σε ξεχωριστά αρχεία μορφής JSON κάτι που την κάνει πολύ γρήγορη στην ανάκτηση πληροφορίας.

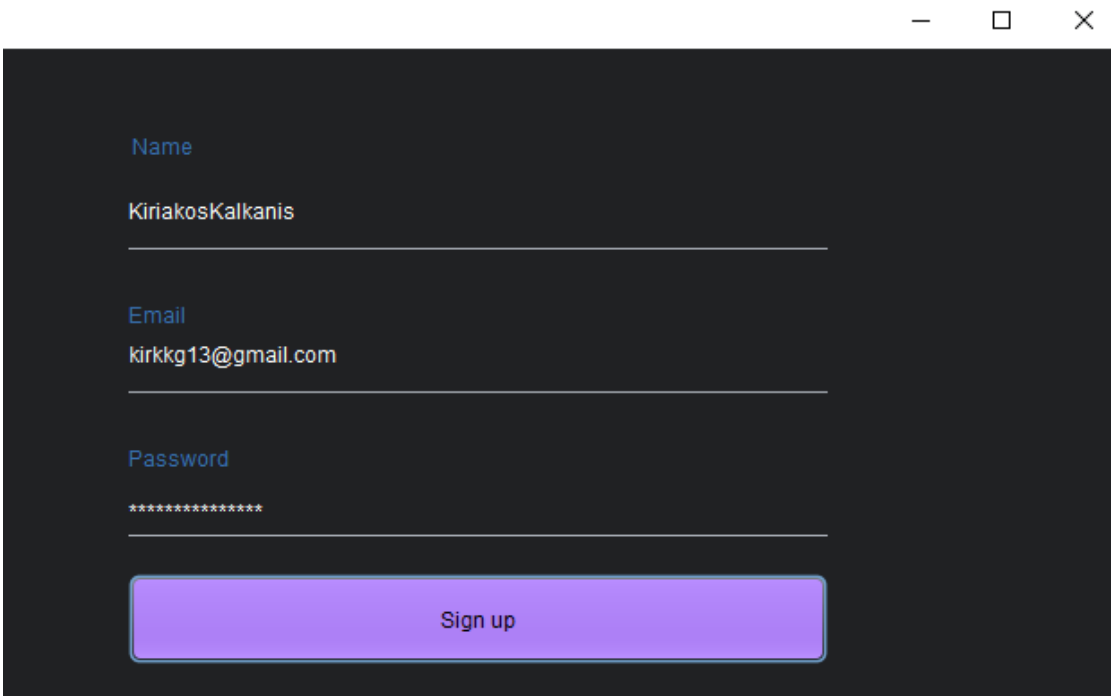
Έπειτα, η γλώσσα προγραμματισμού για την υλοποίηση του project αυτού αποφάσισα μέσα από πάρα πολλές γλώσσες ότι θα είναι η Java. Αυτό οφείλεται στο ότι η Java είναι από τις πιο διαδεδομένες αν όχι η πιο διαδεδομένη γλώσσα προγραμματισμού για τη δημιουργία εφαρμογών καθώς και περιέχει το Swing για τη δημιουργία GUI[16] .

Ο JDBC driver για τη σύνδεση και την αλληλεπίδραση της εφαρμογής με τη βάση δεδομένων που θα χρησιμοποιήσω είναι ο mongo-java-driver-3.12.0 γιατί είναι από τους πιο πρόσφατους.

3.1.1.1 JAVA Swing

Το Swing της Java είναι μια εργαλειοθήκη GUI που επιτρέπει την δημιουργία διεπαφών για τα προγράμματα της Java. Παρέχει components όπως : jbuttons, jtextfields, jlabels,radio buttons, check boxes , panels ,frames και πολλά άλλα τα οποία είναι απαραίτητα για την εικονική αναπαράσταση ενός προγράμματος . Τα components του Swing είναι άμεσα εξαρτώμενα από την πλατφόρμα αφού έχουν γραφτεί σε Java. Επίσης, ο χρήστης επιλέγει τα layout που θέλει να δώσει στις διεπαφές του αφού του δίνονται πολλά layouts από τη Java για να επιλέξει όποια επιθυμεί. Η Sun Microsystems το 2008 κυκλοφόρησε τη JavaFx css/fxml[17] η οποία συνιστάτε για τη δημιουργία πιο μοντέρνων εφαρμογών.

Στην εν λόγω, πτυχιακή εργασία έχει χρησιμοποιηθεί το Swing της Java και σαν εργαλείο ανάπτυξης της εφαρμογής χρησιμοποιήθηκε το netbeans το οποίο είναι ένα IDE[18]. Ακολουθεί εικόνα με παράδειγμα φόρμας sign up που έχει δημιουργηθεί με τη βοήθεια του Swing:



Εικόνα 5 : Παράδειγμα Swing

3.1.1.2 Ο JDBC Driver της MongoDB

Ο JDBC driver της MongoDB συνδέει την εφαρμογή Java με τη βάση δεδομένων MongoDB Atlas και τρέχει ερωτήματα προς τη βάση δεδομένων για την ανάκτηση και αποθήκευση των αρχείων. Για τις ανάγκες τις εν λόγω εργασίας για τη σύνδεση της εφαρμογής με τη βάση δεδομένων έχει χρησιμοποιηθεί ο mongo-java-driver-3.12.0.jar .

Επειδή η MongoDB είναι μια NoSQL βάση δεδομένων ο χρήστης πρέπει να δηλώσει τα πεδία στον σωστό τύπο δεδομένων, ώστε ο JDBC connector να μπορέσει να χειριστεί τα

πεδία με τα καθορισμένα μεταδεδομένα. Για να μην χρειαστεί να κάνει τα παραπάνω ο χρήστης ο driver έχει δημιουργήσει το schematool το οποίο είναι ένα εργαλείο που παίρνει δείγματα από τα collections της βάσης δεδομένων και αντιστοιχεί τα πεδία στους τύπους δεδομένων που βγάζουν νόημα.

Το MongoDB Atlas επιτρέπει στον χρήστη να συνδεθεί μέσω του driver με τις εξής εφαρμογές : Python, Java, Node.JS, Php, C, C++, C#, Ruby, Perl, Scala, Go. Έστω ότι ο χρήστης επιλέξει πως θέλει να συνδέσει τη Java εφαρμογή του με τη MongoDB Atlas βάση δεδομένων :

Εικόνα 6: URI Java MongoDB Atlas

Αυτό που πρέπει να κάνει ο χρήστης είναι να αντιγράψει το παραπάνω URI στην εφαρμογή του στη Java και στη συνέχεια να αλλάξει το key και το password με τα πραγματικά του στοιχεία.

3.1.1.3 Η βάση δεδομένων MongoDB Atlas

Η βάση δεδομένων MongoDB είναι μια NoSQL βάση δεδομένων η οποία αποθηκεύει τις εγγραφές που γίνονται σε μορφή JSON σαν ξεχωριστά αρχεία. Επίσης , πλέον δεν υπάρχει η έννοια του πίνακα όπως στις σχεσιακές βάσεις δεδομένων αλλά η έννοια των collections. Η MongoDB και γενικά οι NoSQL βάσεις δεδομένων είναι δυναμικές βάσεις κάτι το οποίο σημαίνει που δεν απαιτείται κάθε αρχείο που ανήκει στο ίδιο collection(πίνακα στις σχεσιακές βάσεις δεδομένων) να έχει τον ίδιο αριθμό πεδίων αλλά κάθε εγγραφή λειτουργεί αυτόνομα σαν ξεχωριστό αρχείο. Επιπλέον, οι ανάκτηση των αρχείων από τη βάση δεδομένων γίνονται δυναμικά. Στη MongoDB δεν έχουμε joins με άλλα collections και δεν χρειάζεται να δηλώσουμε primary key αφού η MongoDB φτιάχνει ένα μοναδικό id σε κάθε collection για εμάς.

Η MongoDB Atlas είναι μια επέκταση της MongoDB που επιτρέπει στον χρήστη είτε να έχει τη βάση δεδομένων του στο cloud είτε σε server. Για τις ανάγκες της πτυχιακής εργασίας επιλέχθηκε η λύση που δίνει το Atlas με το Cloud. Σε αυτήν την περίπτωση ο

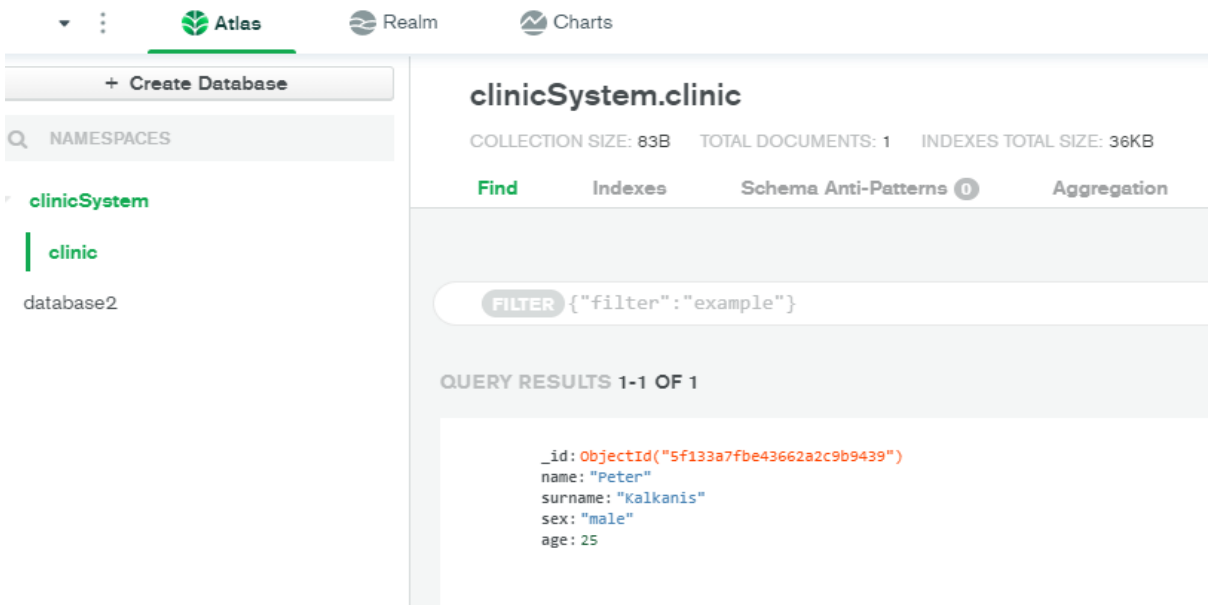
χρήστης που θέλει να το δημιουργήσει πρέπει να δημιουργήσει ένα νέο cluster το οποίο επιτρέπει σε πολλούς διακομιστές βάσεων δεδομένων να μοιράζονται τα ίδια δεδομένα από οπουδήποτε . Έπειτα, γίνεται δημιουργία χρήστη για τη βάση δεδομένων και ο χρήστης δίνει όνομα χρήστη που θα είναι ο admin στη βάση δεδομένων και τα δικαιώματα που θα έχει πχ read and write . Στη συνέχεια ο χρήστης δημιουργεί μια νέα βάση δεδομένων και τέλος είναι έτοιμος να δημιουργήσει collections και documents.

Στη συνέχεια βλέπουμε πως γίνεται η σύνδεση της Java με την mongoDB Atlas και η δημιουργία ενός νέου document στο collection clinic , στη βάση δεδομένων clinicSystem με τη βοήθεια της Java:

```
//Connect with database
String uri = "mongodb+srv://Kiriakos_Kalkanis:1234123412341234@cluster0-xbyif.mongodb.net/test?retryWrites=true&w=majority";
MongoClientURI clientURI = new MongoClientURI(uri);
MongoClient mongoClient = new MongoClient(clientURI);
MongoDatabase mongoDatabase = mongoClient.getDatabase("clinicSystem");

//access to collection
MongoCollection collection = mongoDatabase.getCollection("clinic");

System.out.println("Database connected");
//create new document
Document document = new Document("name", "Peter");
document.append("surname", "Kalkanis");
document.append("sex", "male");
document.append("age", 25);

// insert the document
 collection.insertOne(document);
```

Εικόνα 7: Connect & Insert Document with Java

Αν στη συνέχεια ανοίξουμε το Atlas θα παρατηρήσουμε το νέο insert που μόλις κάναμε :



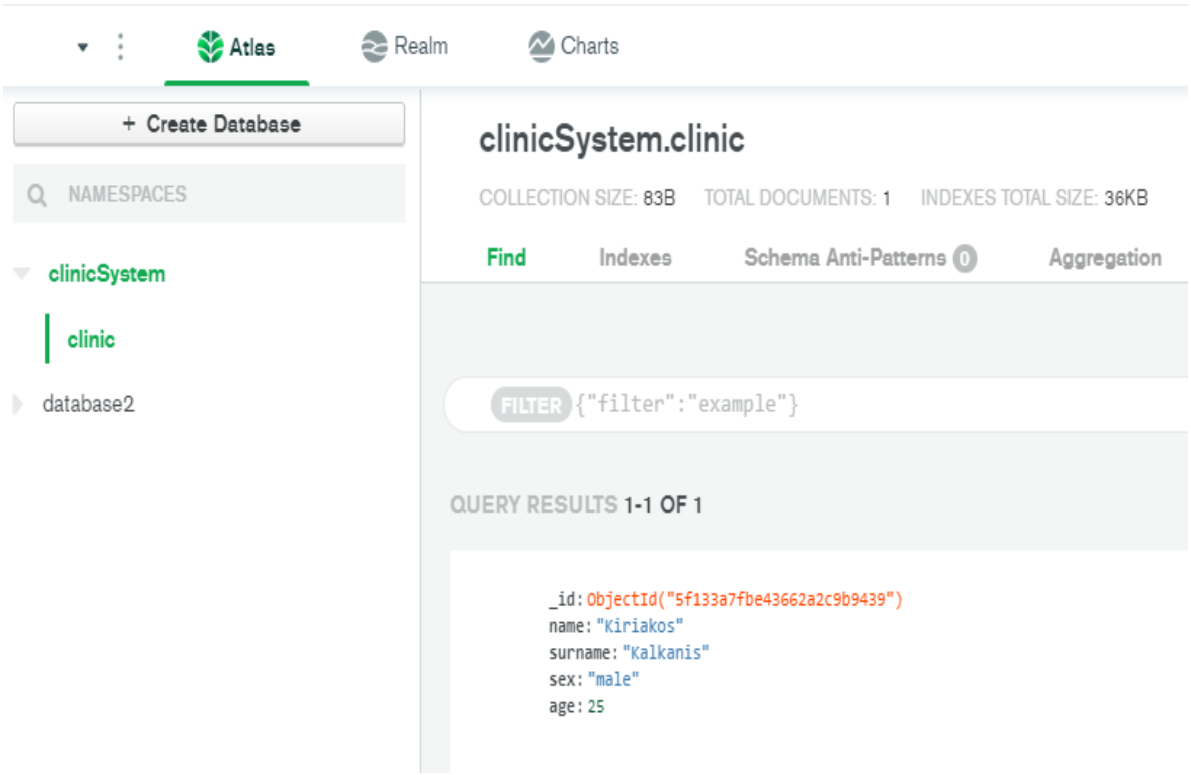
Εικόνα 8: Successful Insertion in mongoDB Atlas

Αντίστοιχα μπορούμε να κάνουμε update το καινούριο document που μόλις δημιουργήσαμε :

```
// Update an existing value
Document found = (Document) collection.find(new Document("name", "Peter")).first();
if (found != null) {
    System.out.println("user found");
    Bson updateValue = new Document("name", "Kiriakos");
    Bson updateOperation = new Document("$set", updateValue);
    collection.updateOne(found, updateOperation);
    System.out.println("User Updated");
```

Εικόνα 9: Update document in Atlas with Java

Αν στη συνέχεια ανοίξουμε το Atlas θα δούμε πως στο προηγούμενο document το name αντί για Peter είναι Kiriakos:



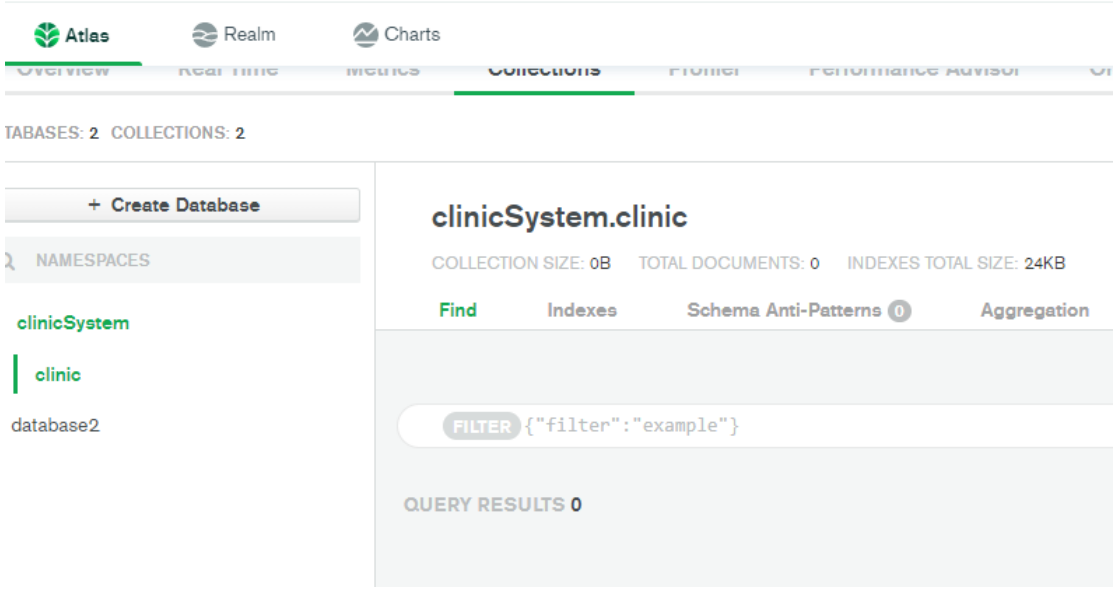
Εικόνα 10: Successful update in Atlas

Τέλος, με τη βοήθεια της Java θα διαγράψουμε το document που κάναμε insert και update :

```
//delete the document
collection.deleteOne(document);
System.out.println("Delete of document is done !");
```

Εικόνα 11: Delete a document in Atlas with Java

Αν ανοίξουμε το Atlas θα παρατηρήσουμε πως δεν υπάρχουν documents αφού ήταν και το μοναδικό :



Εικόνα 12: Successful Delete of document

4. Κύριο Μέρος- Σενάρια Χρήσης

Σε αυτό το κεφάλαιο θα εξετάσουμε τη λειτουργικότητα της εφαρμογής της εν λόγω πτυχιακής εργασίας . Για να επιτευχθεί αυτό , θα πρέπει να εξετάσουμε όλα τα δυνατά σενάρια χρήσης τα οποία παρέχει η εφαρμογή στους χρήστες της. Για την καλύτερη επεξήγηση της εφαρμογής θα χωρίσουμε τα τέσσερα(4) σενάρια σε μια ξεχωριστή υποενότητα το κάθε ένα και θα ακολουθεί πάντα screenshot από την εφαρμογή για την καλύτερη εμπειρία του αναγνώστη.

4.1 Σενάριο χρήσης εφαρμογής από admin

Σε αυτήν την υποενότητα θα εξετάσουμε το σενάριο χρήσης της εφαρμογής από τον administrator του συστήματος. Αρχικά, ο admin θα επιλέξει να συνδεθεί σαν administrator και θα συνδεθεί με τα στοιχεία του τα οποία είναι username και password. Αν είναι σωστά τότε θα έχει πρόσβαση στην καρτέλα του για να εκτελέσει τις όποιες ενέργειες, διαφορετικά θα πρέπει να προσπαθήσει ξανά να συνδεθεί.

Εικόνα 13: Αρχική διεπαφή εφαρμογής

Στη συνέχεια ο χρήστης θα συνδεθεί σαν admin :

Εικόνα 14: admin login form

Αν τα στοιχεία του είναι λάθος θα πρέπει να ξαναπροσπαθήσει :

Εικόνα 15:admin login not successful

Σε διαφορετική περίπτωση θα έχει πρόσβαση στην καρτέλα του:

Εικόνα 16: successful admin's login

Εικόνα 17: Admin's home card

4.1.1 Patient Details

Όταν ο admin συνδεθεί στην καρτέλα του θα έχει πρόσβαση στις εξής καρτέλες :

- Patient Details
- Indoor Patient
- Outdoor Patient
- Doctor Details
- Reception Details

Έστω ότι ο admin επιλέξει την καρτέλα Patient Details όπως βλέπουμε παρακάτω έχει τη δυνατότητα να κάνει add,delete,update και view όλους τους ασθενείς. Σε περίπτωση που ο admin πληκτρολογήσει τα παρακάτω στο tab add Patient και τελικά πατήσει add , στο Atlas της MongoDB θα δημιουργηθεί ένα νέο document της μορφής που είδαμε προηγουμένως στην υποενότητα 3.1.1.3 το οποίο θα περιέχει τα παρακάτω στοιχεία για έναν νέο ασθενή.

Εικόνα 18:Patient Details-add patient

Ακολούθως θα πάρει μήνυμα ότι το insert έγινε με επιτυχία :

Εικόνα 19:Patient Details-add patient success

Στη συνέχεια υποθέτουμε ότι ο admin επιλέγει το tab Update Patient και στο πεδίο Search SSRN συμπληρώσει το SSRN του χρήστη που δημιούργησε προηγουμένως δηλαδή το 1609961122 και πατήσει Search , θα δει τα στοιχεία του ασθενή και θα αλλάξει την τιμή σε όποιο πεδίο θέλει . Έστω ότι αυτό το πεδίο είναι το SSRN και από 1609961122 θέλει να το αλλάξει σε 1609962233 και το age από 24 σε 23 :

Εικόνα 20:Patient Details-update patient

Στη συνέχεια αλλάζει το πεδίο SSRN και age με τις νέες τιμές, πληκτρολογεί update και κάνει search με το νέο SSRN:

Εικόνα 21:Patient Details-update patient successful

Στη συνέχεια ο admin θα πάει στο tab view Patients για να δει τους ασθενείς που έχει δημιουργήσει καθώς και τα στοιχεία τους :

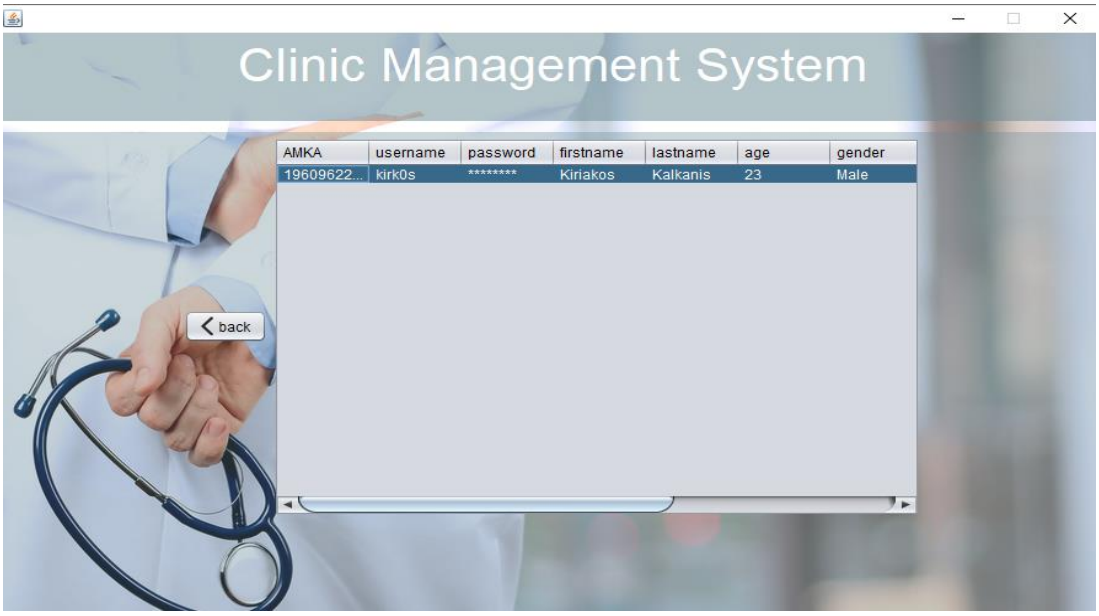
Εικόνα 22:Patient Details-View patients tab

Έστω ότι ο χρήστης πατήσει στο tab delete Patient θέλοντας να διαγράψει τον ασθενή που δημιούργησε. Αρχικά θα κάνει τον ασθενή που επιθυμεί να διαγράψει Search by SSRN και στη συνέχεια θα τον διαγράψει από τη MongoDB:

Εικόνα 23:Patient Details-delete patient successful

4.1.2 Indoor Patient

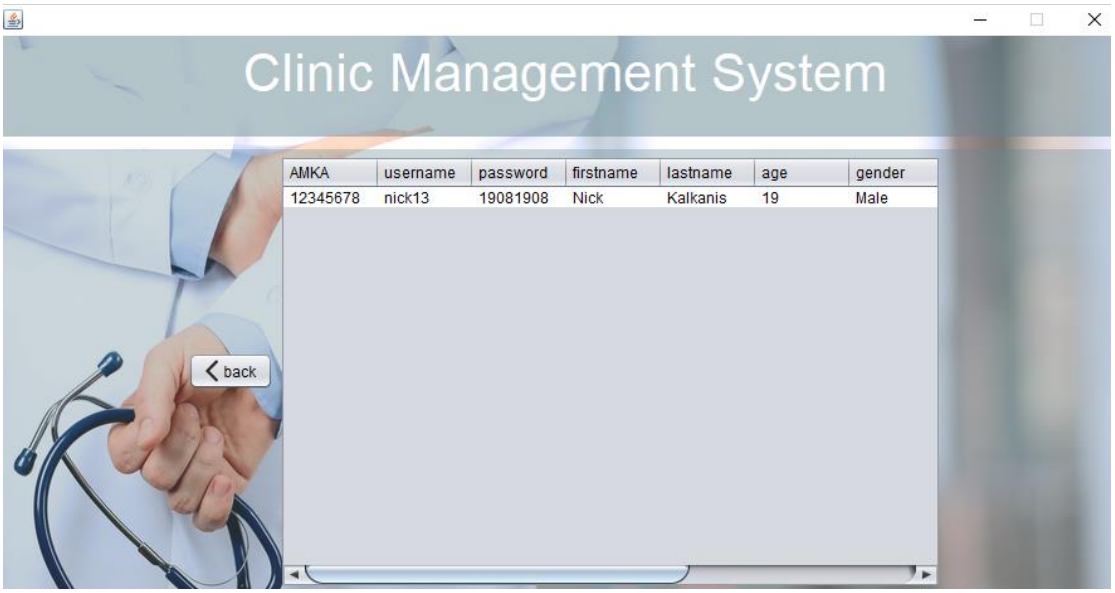
Σε αυτή την υποενότητα ο admin μπορεί να δει όλους τους Indoor Patients . Ένας indoor patient μπορεί να δηλωθεί από όλο το προσωπικό της ιδιωτικής κλινικής . Έστω ότι ο ασθενής στο insert που έκανε ο admin προηγουμένως δηλώθηκε σαν indoor , ο admin θα δει στο indoor patients στο jtable τους indoor patients που έχουν δηλωθεί :



Εικόνα 24:Indoor Patient button

4.1.3 Outdoor Patient

Αντίστοιχα, ο admin μπορεί να μπει στο tab outdoor patient και να δει στο jtable όλους τους outdoor patients που έχουν δηλωθεί από το προσωπικό :

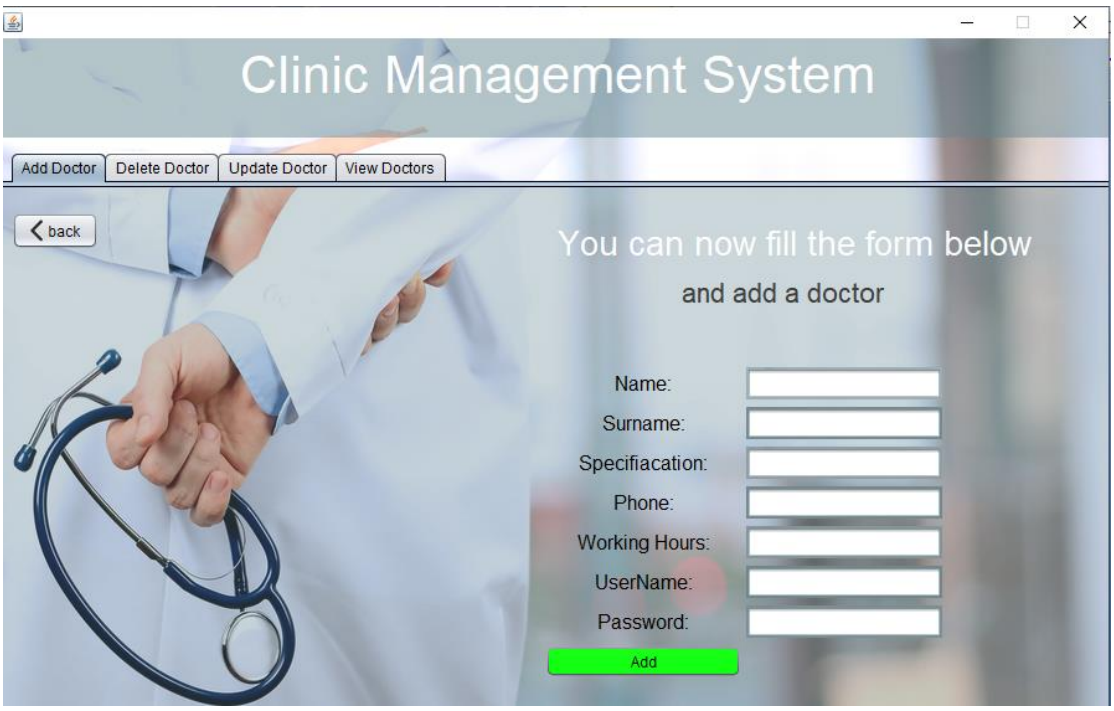


Εικόνα 25:Outdoor patient button

4.1.4 Doctor Details

Αν ο admin επιλέξει το button Doctor Details έχει τις εξής επιλογές :

- Add Doctor
- Delete Doctor
- Update Doctor
- View Doctors

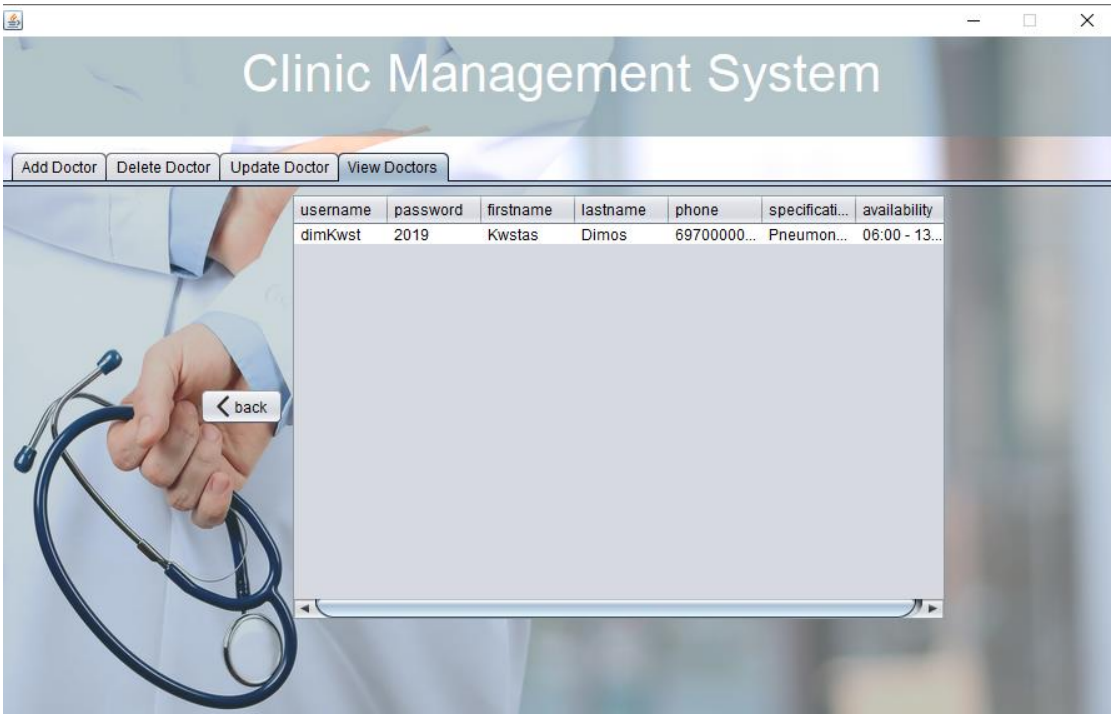


Εικόνα 26:Doctor Details -add doctor

Έστω ότι ο admin συμπληρώσει τη φόρμα και πατήσει add τότε θα δημιουργηθεί ένα νέο document στη MongoDB με τα στοιχεία που συμπλήρωσε ο admin:

Εικόνα 27:Doctor Details -add doctor successful

Έπειτα ,ο χρήστης μπορεί να πάει στο tab View Doctors και να δει όλους τους γιατρούς που υπάρχουν :



Εικόνα 28:Doctor Details -View doctors

Έπειτα ο admin μπορεί να κάνει update τα στοιχεία του γιατρού στο tab Update Doctor αναζητώντας τον με το username του .Έστω ότι το πεδίο Availability από 06:00 – 13:00 θέλουμε να αλλάξει σε 09:00 – 14:00:

Εικόνα 29:Doctor Details -update doctor

Τέλος, ο admin μπορεί να διαγράψει γιατρούς . Για παράδειγμα θα γίνει η διαγραφή του νέου γιατρού που φτιάξαμε σε αυτή την ενότητα :

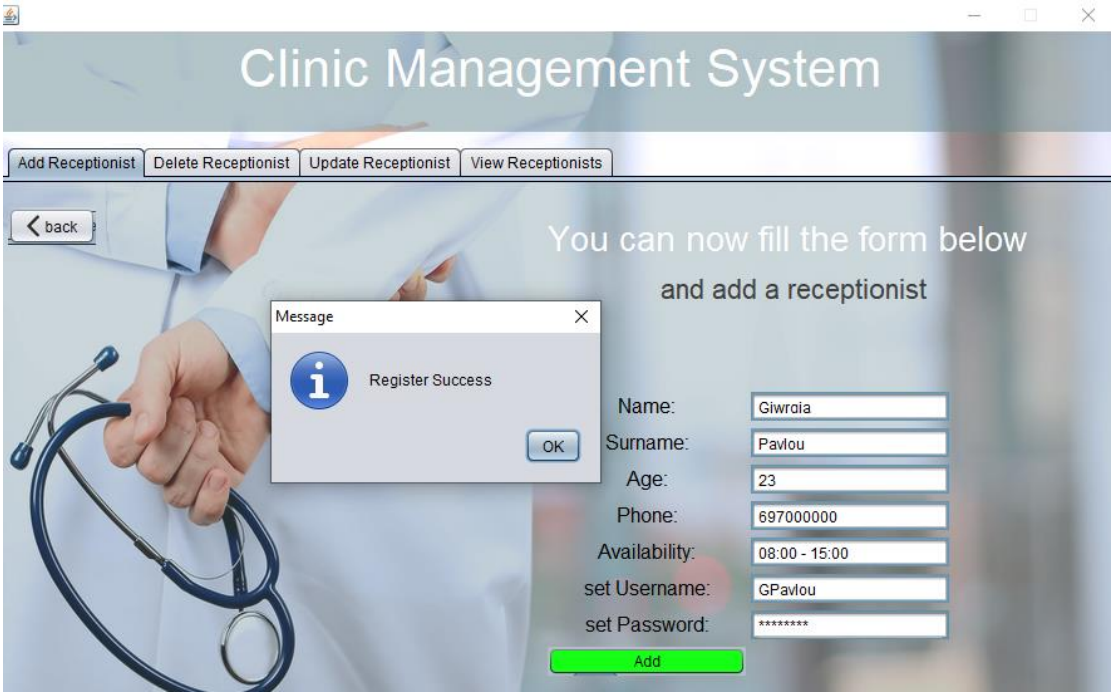
Εικόνα 30:Doctor Details -update doctor successful

4.1.5 Reception Details

Σε αυτήν την υποενότητα ο admin μπορεί να κάνει τις εξής ενέργειες :

- Add Receptionist
- Delete Receptionist
- Update Receptionist
- View Receptionists

Έστω ότι αρχικά ο admin δημιουργεί λογαριασμό για έναν receptionist , θα πρέπει να συμπληρώσει τη φόρμα πχ με τα παρακάτω στοιχεία και στην συνέχεια να πατήσει το add button :



Εικόνα 31:Receptionist Details -add receptionist

Έπειτα μπορεί στο tab View Receptionists να δει τους όλους τους receptionists που υπάρχουν αποθηκευμένοι :

Εικόνα 32:Receptionist Details -view receptionists

Στη συνέχεια θα γίνει update του receptionist που δημιουργήσαμε στο πεδίο availability :

Εικόνα 33:Receptionist Details -update receptionist

Τέλος , θα γίνει η διαγραφή του receptionist :

Εικόνα 34:Receptionist Details - delete receptionist successful

Αν ο admin δεν επιθυμεί άλλη ενέργεια μπορεί σε οποιαδήποτε στιγμή να κάνει logout από την αρχική του καρτέλα.

4.2 Σενάριο χρήσης εφαρμογής από Doctor

Σε αυτήν την υποενότητα θα εξετάσουμε το σενάριο χρήσης της εφαρμογής από τους γιατρούς του συστήματος. Αρχικά, ο γιατρός θα επιλέξει να συνδεθεί σαν doctor και θα συνδεθεί με τα στοιχεία του τα οποία είναι username και password. Αν είναι σωστά τότε θα έχει πρόσβαση στην καρτέλα του για να εκτελέσει τις όποιες ενέργειες, διαφορετικά θα πρέπει να προσπαθήσει ξανά να συνδεθεί:

Εικόνα 35:Αρχική διεαφή -επιλογή doctor button

Εικόνα 36: Doctor's login successful

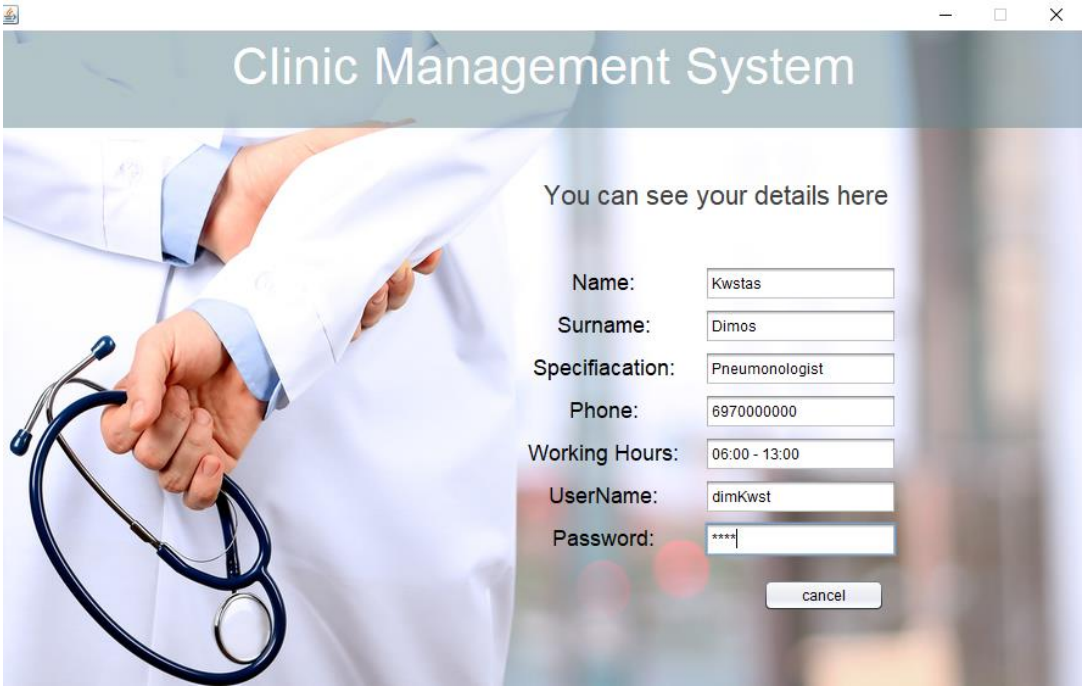
Στη συνέχεια ο χρήστης μπορεί να εκτελέσει τις παρακάτω ενέργειες :

- My Profile
- Appointments

Εικόνα 37: Doctor's home Card

4.2.1 Doctor's Profile

Αν ο γιατρός πατήσει στο my profile button , θα τον κατευθύνει σε μια καρτέλα με τα στοιχεία του :



Εικόνα 38: Doctor's Profile

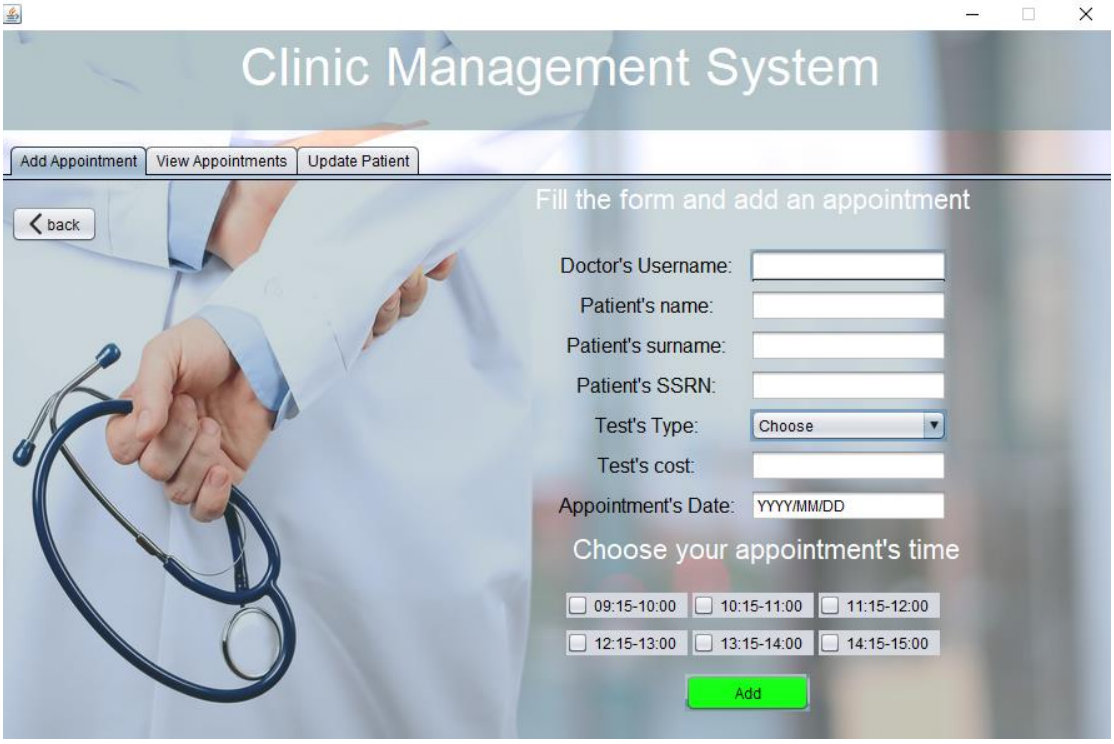
Έπειτα μπορεί να πατήσει cancel για να πάει ένα βήμα πίσω.

4.2.2 Appointments

Σε αυτήν την ενότητα ο συνδεδεμένος γιατρός μπορεί να εκτελέσει μια από τις παρακάτω ενέργειες :

- Add Appointment
- View Appointment
- Update Appointment

Έστω ότι αρχικά θα συμπληρώσει την παρακάτω φόρμα για να προσθέσει ένα ραντεβού για μια συγκεκριμένη εξέταση. Εδώ αξίζει να σημειωθεί πως το πεδίο Test's Type είναι ένα combo Box και ανάλογα με το είδος του test που θα επιλέξει ο γιατρός το πεδίο test's cost θα συμπληρωθεί αυτόματα με το κόστος της εξέτασης το οποίο και λαμβάνεται από τη βάση δεδομένων mongoDB Atlas :

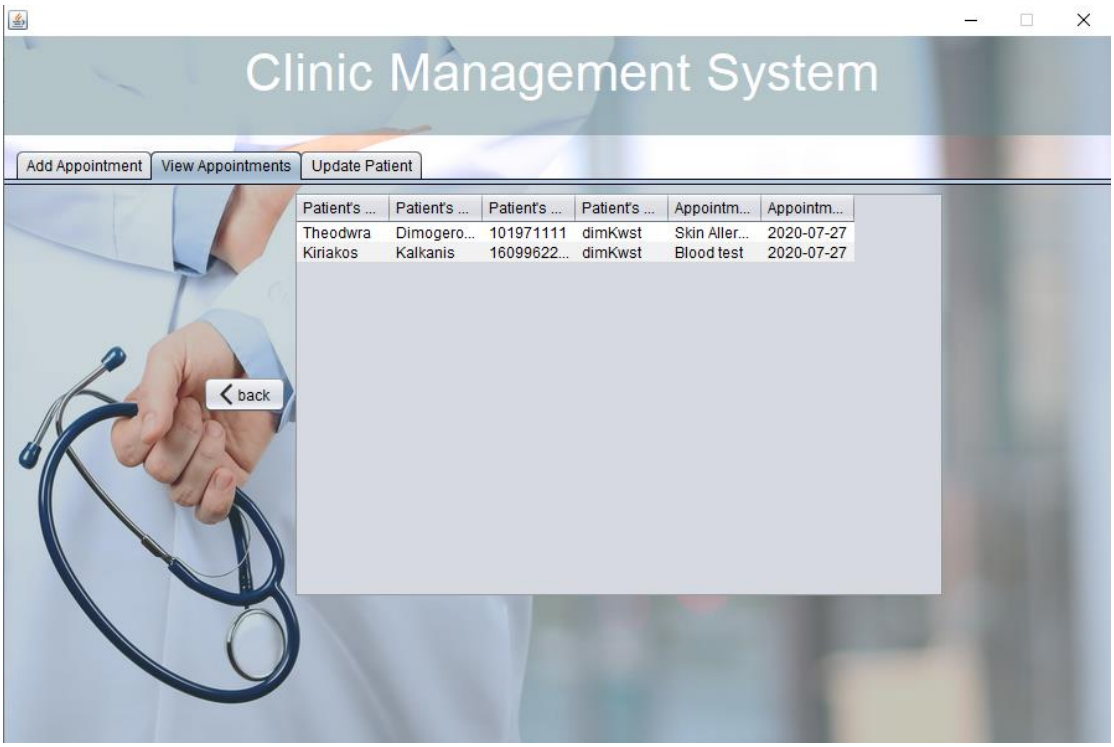


Εικόνα 39:Appointments button

Έστω ότι ο γιατρός συμπληρώνει τη φόρμα με τα ακόλουθα στοιχεία και πατήσει το add button :

Εικόνα 40: doctor adds appointment successfully

Έπειτα ο γιατρός μπορεί να πάει στο View Appointments tab για να δει τα ραντεβού του :



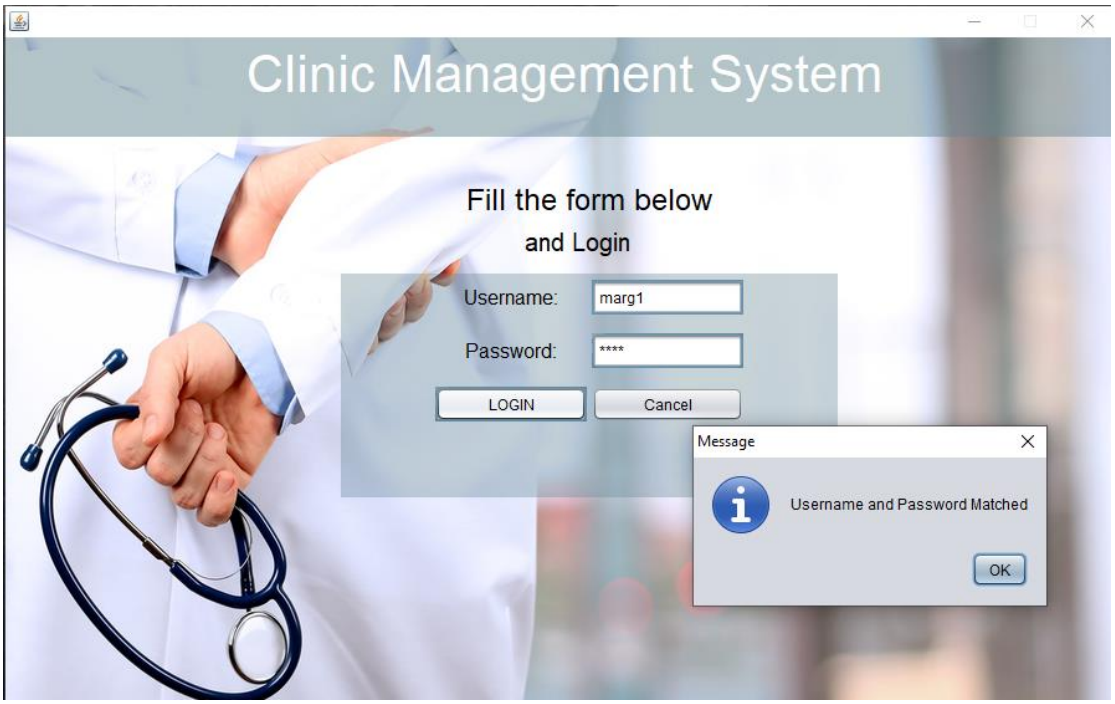
Εικόνα 41:Appointments -View

4.3 Σενάριο χρήσης εφαρμογής από Receptionist

Σε αυτή την ενότητα θα δούμε όλες τις λειτουργίες που μπορούν να εκτελέσουν οι receptionists του συστήματος. Αρχικά , ο χρήστης πρέπει να επιλέξει το Receptionist button από την παρακάτω αρχική διεπαφή :

Εικόνα 42:Αρχική διεπαφή- Recepitonist button

Έπειτα θα πρέπει να συμπληρώσει τα στοιχεία του σωστά για να κάνει login και να έχει πρόσβαση στην καρτέλα του . Έστω ότι έκανε επιτυχές login :



Εικόνα 43: Reeptionist login successful

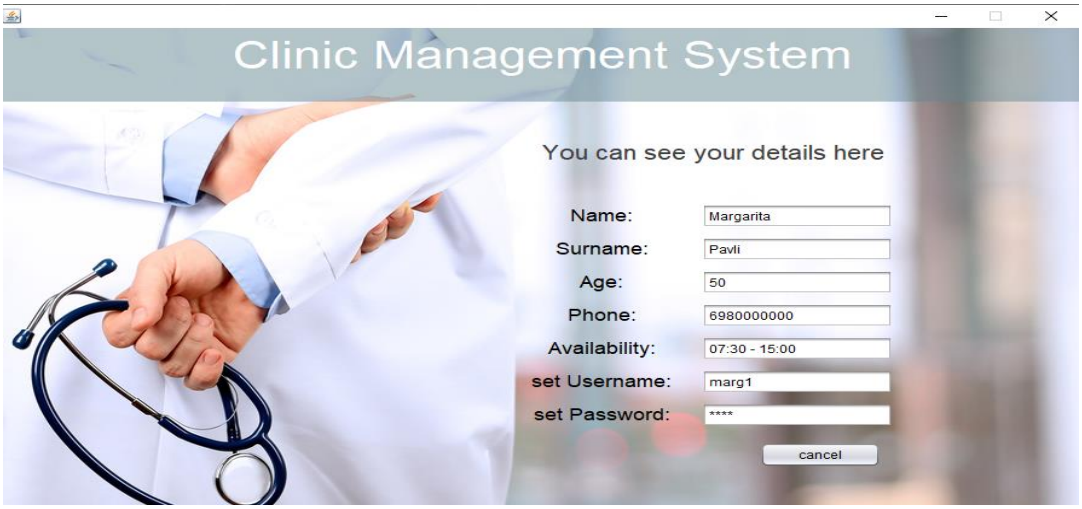
Αφού έκανε επιτυχές login μπορεί να έχει πρόσβαση στην καρτέλα του με τα εξής :

- My Profile
- Appointments
- Patient Details
- Bed Details

Εικόνα 44:Receptionist-Home card

4.3.1 Receptionist's Profile

Σε αυτήν την ενότητα αν ο Receptionist πατήσει στο My Profile button το σύστημα θα τον παραπέμψει σε μια φόρμα με τα στοιχεία του :

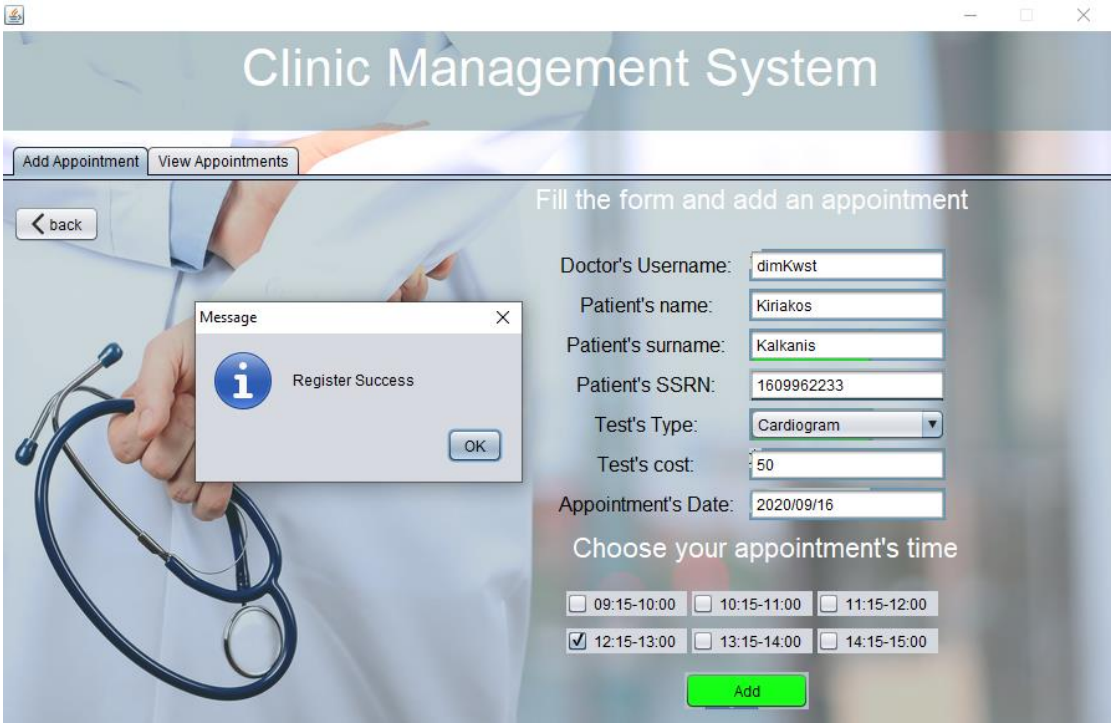


Εικόνα 45:Receptionist's Profile button

4.3.2 Appointments

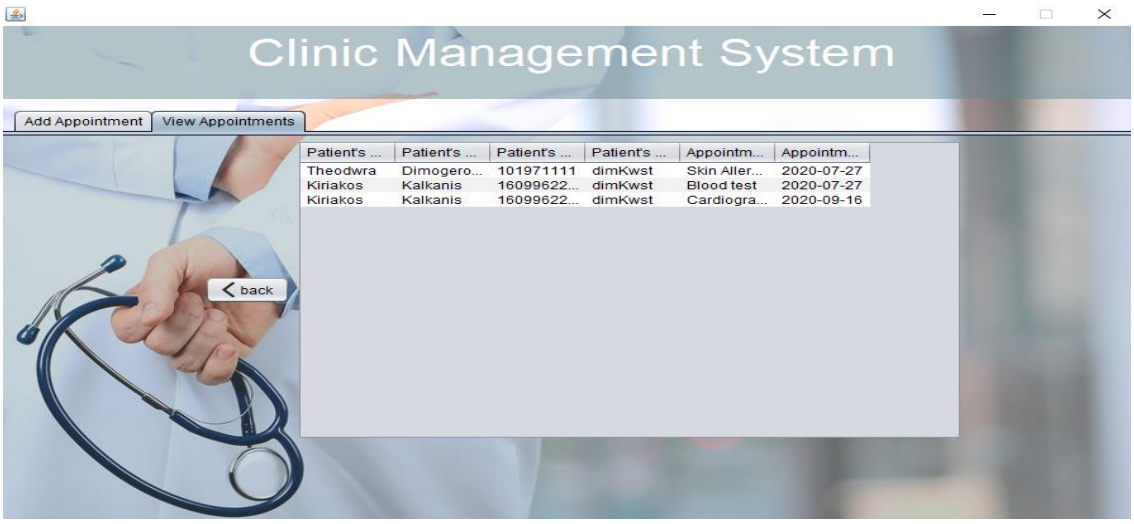
Σε αυτήν την ενότητα ο receptionist μπορεί να κλείσει ραντεβού ανάμεσα σε ασθενή και γιατρό για ιατρική εξέταση καθώς και να δει όλα τα ραντεβού που είναι αποθηκευμένα στην mongoDB βάση δεδομένων ανάμεσα σε γιατρούς-ασθενείς.

Έστω ότι ο /η receptionist θέλει να κλείσει ένα ραντεβού ανάμεσα σε γιατρό και ασθενή για κάποια εξέταση μια συγκεκριμένη ημερομηνία και ώρα συμπληρώνοντας τη φόρμα όπως ακολούθως :



Εικόνα 46:receptionist adds appointments successfully

Στη συνέχεια ο/η receptionist μπορεί να πάει στο tab view Appointments και να δει όλα τα ραντεβού ανάμεσα σε γιατρούς ασθενείς :



Εικόνα 47:Receptionist View appointments

4.3.3 Patient Details

Σε αυτήν την ενότητα ο /η receptionist μπορεί να κάνει τα εξής :

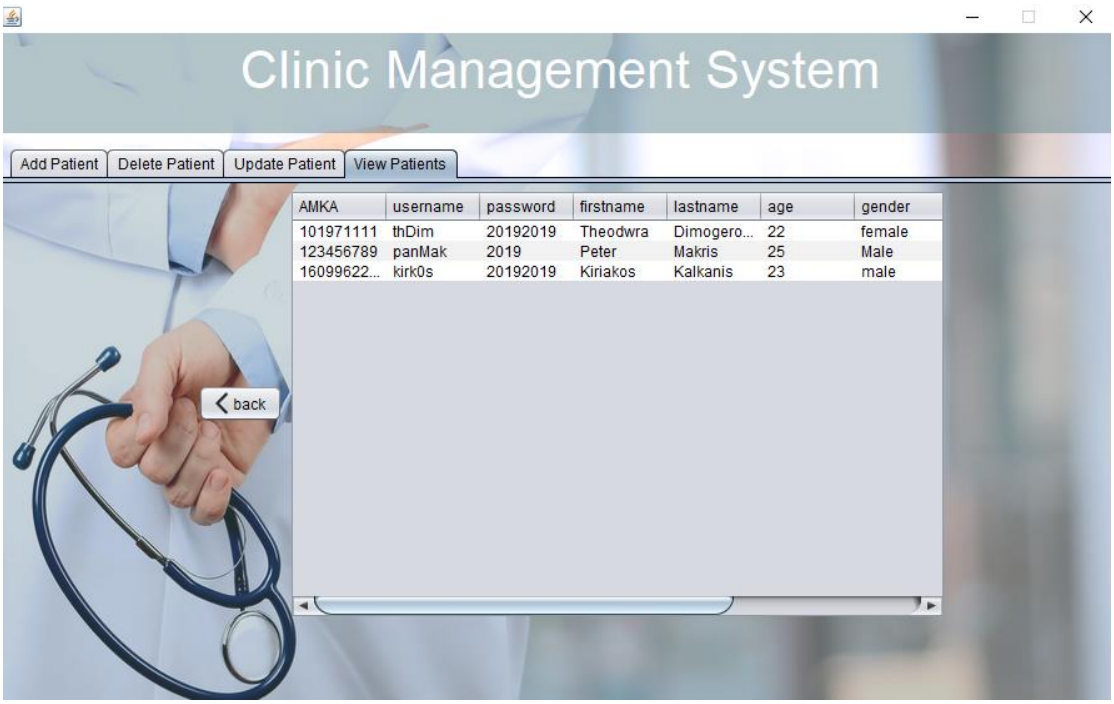
- Add Patient
- Delete Patient
- Update Patient
- View Patient

Εικόνα 48:Receptionist- Patient Details

Ας υποθέσουμε αρχικά ότι ο/η receptionist επιλέγει να κάνει add patient συμπληρώνοντας την φόρμα όπως ακολούθως :

Εικόνα 49:Receptionist- adds new patient successfully

Έστω ότι στη συνέχεια πηγαίνει στο View Tab :



Εικόνα 50:Receptionist- View Patients

Έπειτα , ο/η receptionist θα κάνει Delete του χρήστη που μόλις δημιούργησε και ύστερα θα πατήσει back για να επιστρέψει στην αρχική του καρτέλα :

Εικόνα 51:Receptionist- Delete patient successfully

4.3.4 Bed Details

Σε αυτήν την ενότητα ο/η receptionist μπορεί να εκτελέσει τις ακόλουθες ενέργειες:

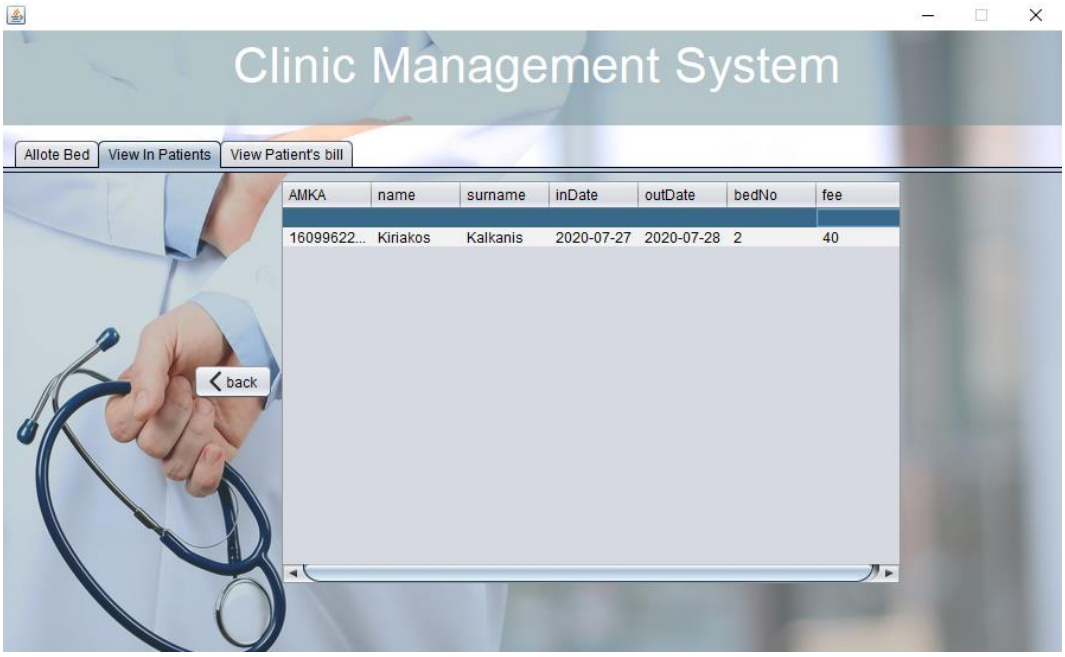
- Allote Bed
- View inPatients
- View Patient's Bill

Εικόνα 52:Receptionist Bed Details button

Αρχικά η receptionist θα δώσει κρεβάτι σε ασθενή αφού ο γιατρός την έχει ενημερώσει και το έχει κρίνει σκόπιμο . Να προσθέσουμε εδώ ότι η receptionist μπορεί ταυτόχρονα είτε να δημιουργήσει κατευθείαν έναν χρήστη και να του δώσει αριθμό κρεβατιού είτε να αναζητήσει έναν υπάρχον ασθενή με το SSRN του και να του δώσει αριθμό κρεβατιού καταχωρώντας τα στοιχεία του σαν document στη MongoDB :

Εικόνα 53:Receptionist gives a new bed

Στη συνέχεια η receptionist μπορεί να πάει στο View in Patients tab, το κόστος της διανυκτέρευσης είναι 40 ευρώ :



Εικόνα 54:Receptionist View inpatients

Έπειτα μπορεί να πάει στο View Patient's bill και να δει τα συνολικά ποσά των ασθενών , το παρακάτω κόστος είναι 40 από τη μία(1) διανυκτέρευση + 40 από το κόστος του blood test:

Εικόνα 55: Receptionist view patient's bill

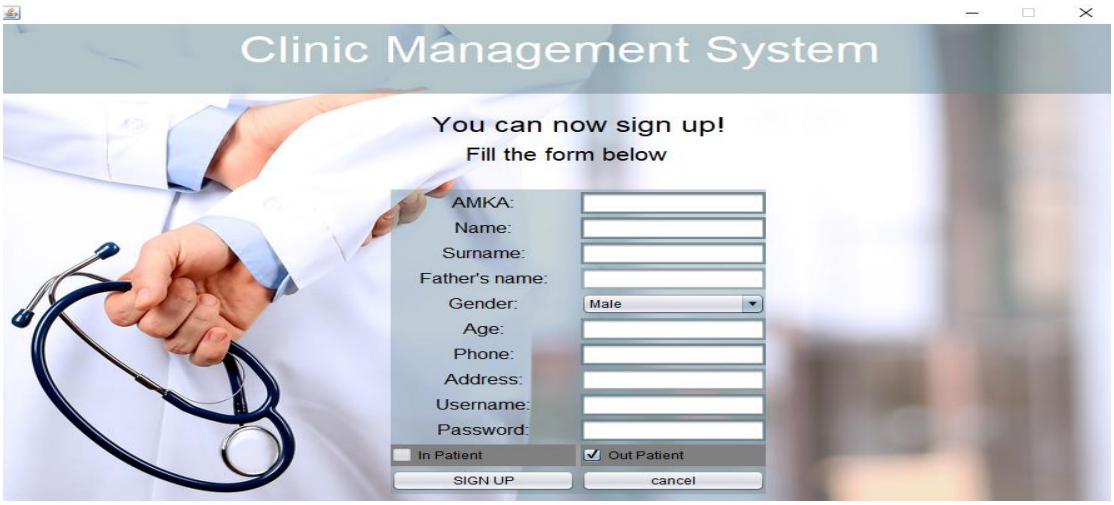
4.4 Σενάριο χρήσης εφαρμογής από Patient

Σε αυτή την ενότητα θα δούμε τις ενέργειες που μπορούν να κάνουν οι ασθενείς . Αρχικά θα εκτελέσουμε το σενάριο πως ο χρήστης είναι νέος και δεν έχει λογαριασμό . Υπενθυμίζεται , πως μόνο οι ασθενείς μπορούν να δημιουργήσουν λογαριασμό , το λοιπό προσωπικό μπορεί να έχει λογαριασμό μόνο δημιουργημένο από τον administrator του συστήματος.

Εικόνα 56:Patient's login form

4.4.1 Sign Up Patient

Έστω πως ο χρήστης επιλέγει το button ,Click here to Sign Up :

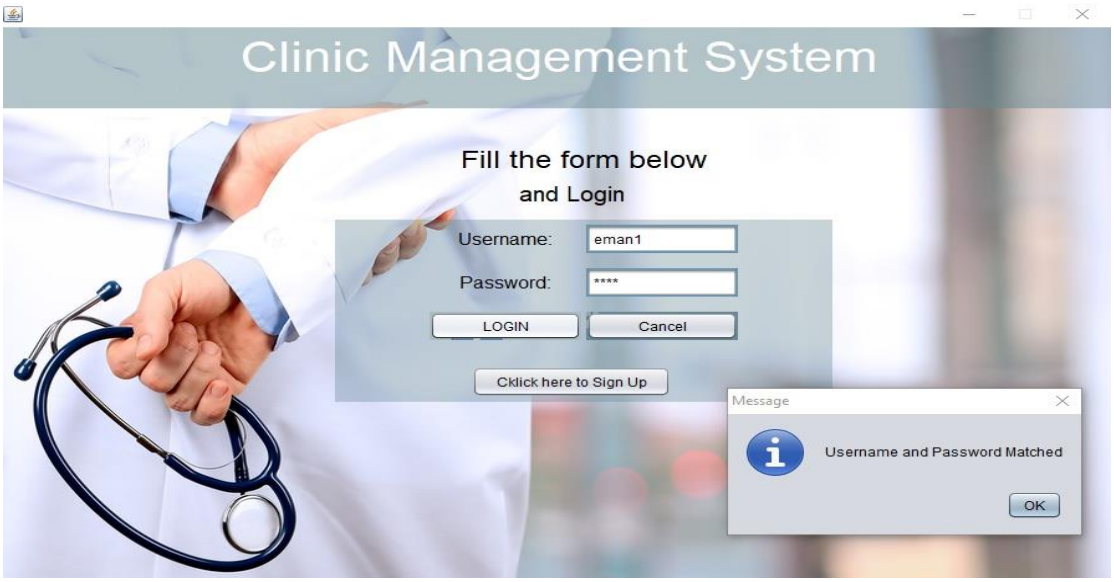


Εικόνα 57:Patient's signup form

Στη συνέχεια συμπληρώνει την παραπάνω φόρμα ως ακολούθως και επιλέγει το button Sign Up :

Εικόνα 58: Successful Sign Up

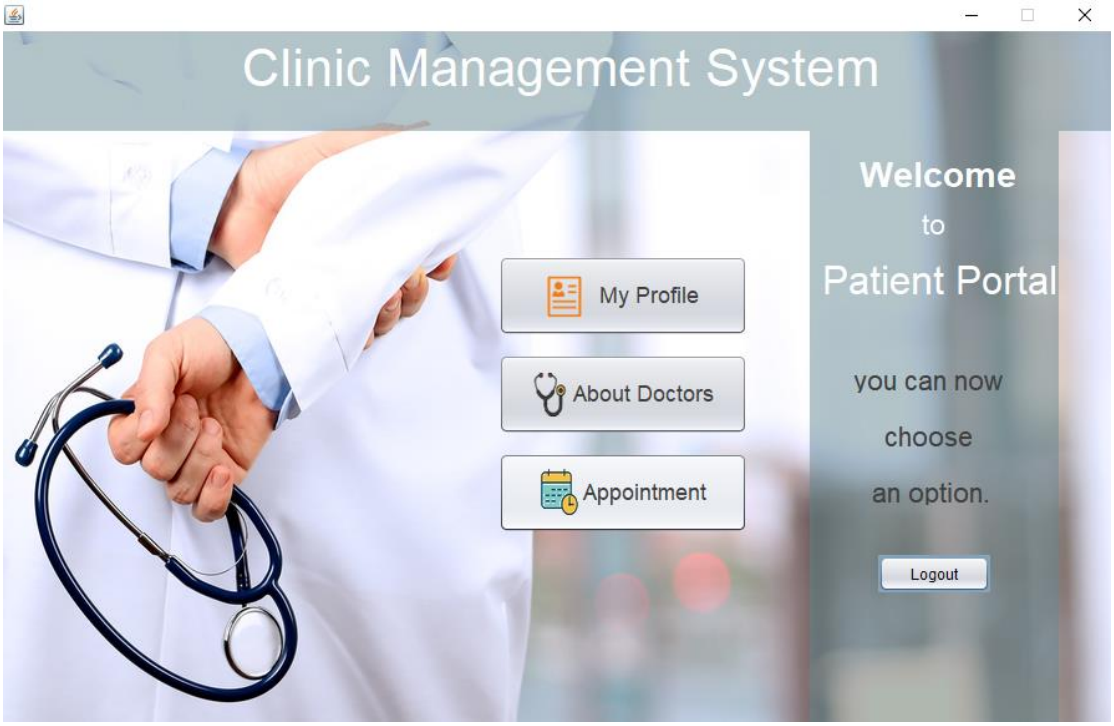
Έπειτα μπορεί να κάνει login με τα στοιχεία του:



Εικόνα 59::Patient's login successful

Αφού ο ασθενής εισέλθει επιτυχώς στον λογαριασμό του μπορεί να εκτελέσει τις εξής λειτουργίες :

- My Profile
- About Doctors
- Appointment



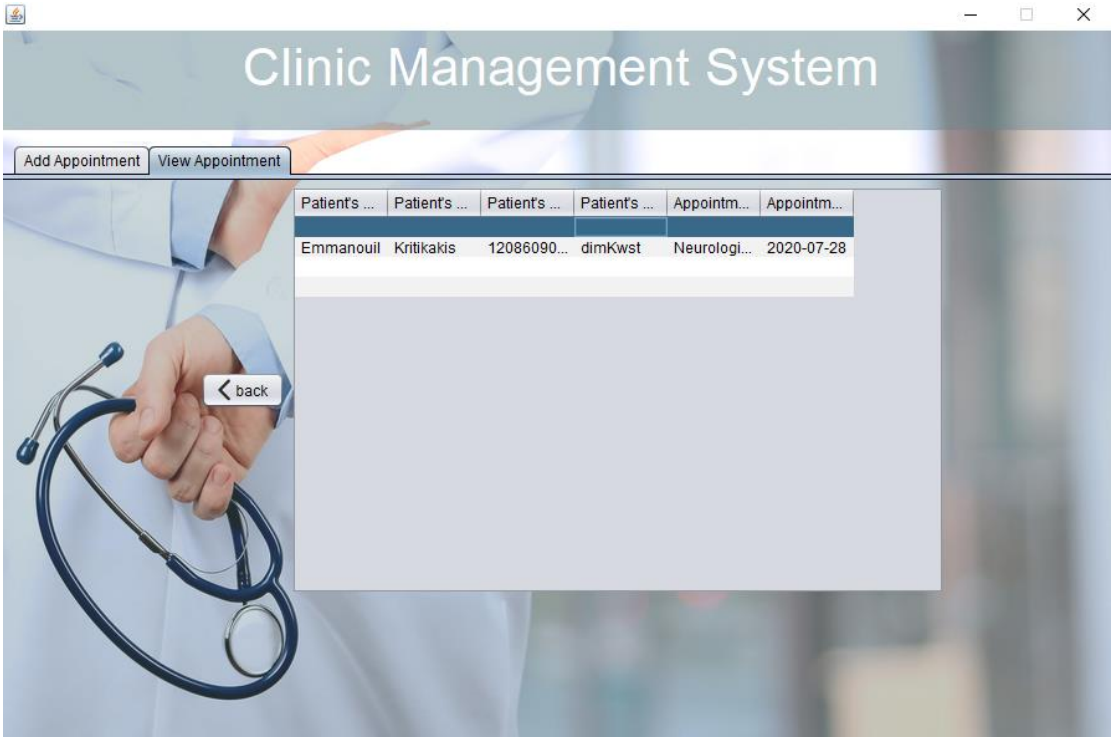
Εικόνα 60: Patient's home Card

4.4.2 Appointments

Έστω πως ο χρήστης αρχικά πηγαίνει στο Appointment , θα μπορεί να κλείσει ραντεβού για εξέταση και να δει τα ραντεβού του :

Εικόνα 61:Patient adds Appointment successfully

Έπειτα μπορεί να δει όλα τα ραντεβού τα οποία αναφέρονται σε αυτόν :



Εικόνα 62:Patient view his appointments

4.4.3 About doctors

Σε αυτήν την ενότητα ο ασθενής μπορεί να δει τα στοιχεία των γιατρών που υπάρχουν στο σύστημα :

Εικόνα 63:About doctors button

4.4.4 My Profile

Σε αυτήν την ενότητα ο χρήστης μπορεί να δει τα στοιχεία του , συμπεριλαμβανομένου των λογαριασμό που οφείλει να εξοφλήσει το οποίο είναι 60 ευρώ αφού αυτό είναι το κόστος της εξέτασης που έκλεισε προηγουμένως .

Εικόνα 64:Patients profile button

Αν ο χρήστης δεν επιθυμεί άλλες ενέργειες μπορεί να πατήσει cancel και στη συνέχεια logout.

5. Αποτελέσματα

Η εν λόγω εφαρμογή, θα μπορούσε να βοηθήσει τόσο το προσωπικό μια ιδιωτικής κλινικής στην οργάνωση των καθηκόντων και ενεργειών τους όσο και τους πελάτες της κλινικής έτσι ώστε να αποφύγουν την αναμονή λόγω υπερβολικού φόρτου κλήσεων στην τηλεφωνική γραμμή κλείνοντας απλά και εύκολα ραντεβού για κάποια εξέταση μέσω της εφαρμογής. Το παραπάνω μπορεί να γίνει ανεξαρτήτου ωραρίου λειτουργίας της κλινικής.

5.1 Συμπεράσματα

Ολοκληρώνοντας την εν λόγω εφαρμογή ,η οποία αποτελεί το αντικείμενο της πτυχιακής μου μελέτης, προσθέτω στο προσωπικό μου portfolio μια σωστά λειτουργικά και άρτια δομημένη desktop εφαρμογή την οποία υλοποίησα από την αρχή ως το τέλος. Η υλοποίηση του συγκεκριμένου project με βοήθησε να αναπτύξω τις γνώσεις μου και να τις εφαρμόσω στην πράξη . Η έρευνα μου στηρίχτηκε στην αναζήτηση διαφόρων νέων τεχνολογιών που χρησιμοποιούνται στην εποχή μας αλλά και στην εφαρμογή των γνώσεων μου.

5.2 Μελλοντική Επέκταση

Θα μπορούσαν να προστεθούν μερικές επιμέρους λειτουργίες όπως η κράτηση του ιστορικού των ασθενών , η προσθήκη ενός φαρμακοποιού στην εφαρμογή ή επικοινωνία της παρούσας εφαρμογής με μία άλλη εξωτερική εφαρμογή η οποία θα διαχειρίζεται τα φάρμακα των ασθενών. Τέλος, θα αποτελούσε κίνητρο για εμένα αν έκανα την εν λόγω εφαρμογή να δουλεύει και σε κινητές συσκευές.

Βιβλιογραφία

[1] Γλώσσα προγραμματισμού Java , στο σύνδεσμο : https://en.wikipedia.org/wiki/Java_(programming_language)

[2] JVM of Java , στο σύνδεσμο : https://en.wikipedia.org/wiki/Java_virtual_machine

[3] Garbage collector of Java, στο σύνδεσμο: https://www.geeksforgeeks.org/garbage-collection-java/

[4] Βάσεις Δεδομένων στο σύνδεσμο : https://en.wikipedia.org/wiki/Database

[5] Σχεσιακές βάσεις δεδομένων, στο σύνδεσμο : https://en.wikipedia.org/wiki/Relational_database

[6] NoSQL βάσεις δεδομένων στο σύνδεσμο: https://en.wikipedia.org/wiki/NoSQL

[7] Web 2.0 ,στο σύνδεσμο : https://en.wikipedia.org/wiki/Web_2.0

[8]MongoDB , στο σύνδεσμο : https://www.mongodb.com/what-is-mongodb

[9] JDBC driver , στο σύνδεσμο: https://en.wikipedia.org/wiki/JDBC_driver

[10] API, στο σύνδεσμο : https://en.wikipedia.org/wiki/Application_programming_interface

[11] : JDBC to ODBC bridge, στο σύνδεσμο : https://www.progress.com/tutorials/jdbc/odbc-jdbc-bridge-quick-and-easy-odbc-wrapper-for-a-jdbc-data-source

[12]: SQL, στο σύνδεσμο : https://en.wikipedia.org/wiki/SQL

[13]MongoDB Atlas, στο σύνδεσμο : https://www.mongodb.com/cloud/atlas

[14] Java Swing, στο σύνδεσμο : https://en.wikipedia.org/wiki/Swing_(Java)

[15] Collections in NoSQL, στο σύνδεσμο : https://docs.mongodb.com/manual/core/databases-and-collections/

[16] : GUI, στο σύνδεσμο : https://en.wikipedia.org/wiki/Graphical_user_interface

[17] : Java FX, στο σύνδεσμο : https://openjfx.io/

[18] IDE , στο σύνδεσμο : https://hackr.io/blog/best-java-ides