ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών

«ΠΜΣ ΠΛΗΡΟΦΟΡΙΚΗ»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ανάπτυξη ιατρικού ηλεκτρονικού φακέλου με τη χρήση spring boot** **Development of electronic medical record using spring boot**
Ονοματεπώνυμο Φοιτητή	**Στάμου Μαρία Νίκη**
Πατρώνυμο	**Σπυριδων**
Αριθμός Μητρώου	**ΜΠΠΛ20078**
Επιβλέπων	**Αλέπης Ευθύμιος, Αναπληρωτής Καθηγητής**

Φεβρουάριος 2023

Τριμελής Εξεταστική Επιτροπή

Αλέπης Ευθύμιος, Αναπληρωτής Καθηγητής	Βίρβου Μαρία Καθηγήτρια	Σακκόπουλος Ευάγγελος Αναπληρωτής Καθηγητής

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά, και από τη θέση αυτή, τον επιβλέποντα καθηγητή μου κ. Ευθύμιο Αλέπη που μου εμπιστεύτηκε αυτό το τόσο ενδιαφέρον και επίκαιρο θέμα για την υλοποίηση της διπλωματικής μου εργασίας. Χωρίς την πολύτιμη καθοδήγηση του και τις εύστοχες παρατηρήσεις του δε θα μπορούσα να φέρω εις πέρας το έργο αυτό. Επίσης, θα ήθελα να εκφράσω τις ευχαριστίες μου για την οικογένεια μου και ιδιαίτερα τον σύζυγο μου οποίος με στήριξε όσο κανείς σε αυτή την προσπάθεια.

Πίνακας περιεχομένων

ΠΕΡΙΛΗΨΗ

Είναι γνωστό πως ακόμα και στην σύγχρονη εποχή που χαρακτηρίζεται από υψηλό βαθμό τεχνολογικής ανάπτυξης, η γραφειοκρατία και η κακή οργάνωση παραμένουν αναπόσπαστο χαρακτηριστικό του ελληνικού συστήματος υγείας. Από την αρχαιότητα, την εποχή του Ιπποκράτη, μέχρι και σήμερα έχουν γίνει προσπάθειες οργάνωσης για αποτελεσματικότερη παρακολούθηση και θεραπεία μέσω τον ιατρικών φακέλων ασθενών οι οποίοι μέχρι λίγα χρόνια πριν ήταν αποκλειστικά σε χειρόγραφη μορφή, γεγονός που τα τελευταία χρόνια τείνει να μεταβληθεί, καθώς η είσοδος του ηλεκτρονικού υπολογιστή στα νοσοκομεία και τα ιδιωτικά ιατρεία έχει συμβάλλει στην ψηφιοποίησή τους. Με αυτό τον τρόπο τα στοιχεία των ασθενών, το ιατρικό ιστορικό τους και όλες οι θεραπευτικές μέθοδοι που ακολουθήθηκαν είναι αποθηκευμένα στην ηλεκτρονική «καρτέλα» του καθενός, κάνοντας έτσι πιο εύκολη την παρακολούθηση της εξέλιξή τους από τον θεράποντα ιατρό.

Η παρούσα εργασία έχει ως σκοπό να αποδείξει την αναγκαιότητα της χρήσης των ηλεκτρονικών ιατρικών φακέλων τόσο στα νοσοκομεία όσο και στις ιδιωτικές ιατρικές μονάδες, καθώς βασικά βήματα για την υλοποίηση ενός ηλεκτρονικού ιατρικού φακέλου. Αρχικά, παρατίθενται πληροφορίες που αφορούν την κατάσταση του συστήματος υγείας της χώρας μας, παρουσιάζεται μία σύντομη ιστορική αναδρομή του ιατρικού φακέλου και τους σκοπούς που εξυπηρετεί, τα μειονεκτήματά του κλασικού ιατρικού φακέλου και τα πλεονεκτήματα του ηλεκτρονικού ιατρικού φακέλου. Στη συνέχεια ορίζεται η έννοια του ιατρικού φακέλου καθώς και πραγματοποιείται απαρίθμηση και η ανάλυση τεχνολογιών τεχνικής λύσης. Επιπροσθέτως, πραγματοποιείται ο σχεδιασμός τεχνικής λύσης, η παρουσίαση υψηλού επιπέδου διαγράμματος και η παρουσίαση διαγραμμάτων οντοτήτων και συσχετίσεων. Τέλος, υπάρχει η παρουσίαση των βασικών κομματιών του κώδικα της τεχνικής υλοποίησης καθώς και της δομής αυτής και τονίζεται η λειτουργικότητα της εφαρμογής παραθέτοντας στιγμιότυπα.

ABSTRACT

It is known that even in the modern era characterized by a high degree of technological development, bureaucracy and poor organization remain an integral feature of the Greek health system. From ancient times, at the time of Hippocrates, until today, efforts have been made to organize more effective patient monitoring and treatment through the medical records of patients, which until a few years ago were exclusively in handwritten form, a fact that in recent years tend to change, as the entry of the computer in hospitals and private medical institutes has contributed to their digitization. In this way, the patients' personal data, their medical history and all the treatment methods followed are stored in each person's electronic "record", thus making it easier for the attending doctor to monitor their progress.

This thesis aims to demonstrate the necessity of using electronic medical records both in hospitals and in private medical units, as well as basic steps for the implementation of an electronic medical record. First, we provide information regarding the state of our country's health system, a brief historical overview of the medical file and the purposes it serves, the disadvantages of the classic medical file and the advantages of the electronic medical file. The concept of the medical file is then defined and technologies used in technical solution are enumerated and analyzed. In addition, technical solution design, high-level diagram presentation, and entity and relationship diagram presentation are performed. Finally, the basic parts of code of the technical implementation as well as its structure are presented and the functionality of the application is emphasized by provided snapshots.

ΚΕΦΑΛΑΙΟ 1 ΕΙΣΑΓΩΓΗ

Τα Δημόσια Νοσοκομεία είναι γεγονός ότι αντιμετωπίζουν σοβαρά προβλήματα διαχείρισης και εκσυγχρονισμού των παρεχόμενων υπηρεσιών υγείας με κύριο αποτέλεσμα τη μη εύρυθμη λειτουργίας τους και δυστυχώς το χαμηλό βαθμό ικανοποίησης των πολιτών. Ο πολίτης που καταφεύγει σε ένα Δημόσιο Νοσοκομείο ζητώντας την απαραίτητη υγειονομική φροντίδα, δηλώνει σιωπηρά την εμπιστοσύνη του στις υπηρεσίες παροχής υγείας, καθώς και την αποδοχή του στο σύστημα της Δημόσιας Διοίκησης. Επίσης, έχει την «απαίτηση» η νοσοκομειακή περίθαλψη που ο ίδιος πληρώνει μέσω των κρατήσεων από την προσωπική του εργασία να μπορεί να καλύψει τις ιδιαίτερες ανάγκες του, όποτε το χρειαστεί. Το σύστημα, όμως, δεν αποδεικνύεται αντάξιο των προσδοκιών του, καθώς δημιουργεί παραλείψεις, καθυστερήσεις, χαοτική γραφειοκρατεία, υπέρογκες χρηματικές και μη- επιβαρύνσεις που έχουν ως αποτέλεσμα τη χαμηλή απόδοση των υπηρεσιών υγείας. Ταυτόχρονα, έρχεται αντιμέτωπος με κακοσυντηρημένες εγκαταστάσεις, υποδομές και τρομερές ελλείψεις – μικρός αριθμός κλινών, «σκοτεινά» και ασφυκτικά γεμάτα δωμάτια, απουσία ιατρικών μηχανημάτων απαραίτητων για τη διεξαγωγή σοβαρών εξετάσεων, ελλείψεις σε φάρμακα, ελλείψεις ακόμα και σε υγειονομικό προσωπικό- οι οποίες επιβαρύνουν την ήδη επιβαρυμένη ψυχολογία του, ενώ είναι γνωστό, και μάλιστα οι ίδιοι οι ιατροί το επιβεβαιώνουν, πως η καλή ψυχολογία, σε συνδυασμό βέβαια με την κατάλληλη και αποτελεσματική θεραπευτική αγωγή, αποτελεί τον ακρογωνιαίο λίθο της ίασης ακόμα και σοβαρών ασθενειών. Στη διπλωματική αυτή θα παρουσιάσουμε μια μορφή ηλεκτρονικού ιατρικού φακέλου ασθενούς η οποία πραγματικά αποτελεί μια νέα πρόταση και ουσιαστικά προσφέρει ως «από μηχανής» θεός την λύση στο πρόβλημα της χαοτικής και ξεπερασμένης συλλογής χαρτιών, δικαιολογητικών και παραπεμπτικών, που μόνο σκοπέλους δημιουργούν κατά τη διαδικασία της νοσηλείας και κατ' επέκτασιν της θεαραπείας. Είναι μια πρόταση για ένα νέο πρωτοπόρο εργαλείο στην παρουσίαση και οργάνωση της πληροφορίας, το οποίο κυριολεκτικά θα φανεί σωτήριο, τόσο για τους ιατρούς και το προσωπικό του εκάστοτε νοσοκομείου, όσο και για τους ασθενείς οι οποίοι θα αποφύγουν μια κουραστική και χρονοβόρο διαδικασία. Η ανάπτυξη της τεχνολογίας, παράλληλα με άλλα οφέλη που μας έχει προσφέρει σε όλες της πτυχές του βίου μας, έχει εισάγει την ψηφιοποίηση των δεδομένων και με αυτό τον τρόπο έχει κάνει επιτακτική την ανάγκη της νέας οργάνωσης της πληροφορίας, καθώς και της πρότασης για μελέτη των δεδομένων και στατιστικών για περιθώρια βελτίωσης και εξαγωγής συμπερασμάτων.

Ο ηλεκτρονικός ιατρικός φάκελος, που αποτελεί και την πρόταση μας, είναι η "αποθήκη" όλων των πληροφοριών που αφορούν στο ιατρικό ιστορικό του ασθενούς. Αποτελεί επομένως την βάση της διάγνωσης και της θεραπευτικής αντιμετώπισης του ασθενούς και επιπλέον, παρέχει πληροφορίες διοικητικής, οικονομικής, στατιστικής φύσεως, καθώς και ποιοτικού ελέγχου. Είναι ένα σύγχρονο εργαλείο, το οποίο κάθε ιατρός και νοσοκομείο που σέβεται το λειτούργημα και τους ασθενείς του, οφείλει να διαθέτει.

Η ΙΣΤΟΡΙΑ ΤΟΥ ΙΑΤΡΙΚΟΥ ΦΑΚΕΛΟΥ

Η ιστορία του ιατρικού φακέλου ξεκινά από αρκετά παλιά. Τον 5ο αιώνα π.Χ οι ιατρικές εκθέσεις επηρεάστηκαν πολύ από τον Ιπποκράτη. Ο Ιπποκράτης πρώτος συνηγόρησε το ιατρικό ιστορικό ώστε να εξυπηρετεί τους δύο βασικούς στόχους:

i. Να αντικατοπτρίζει με ακρίβεια την πορεία της ασθένειας του ασθενή.
ii. Να υποδεικνύει τις πιθανές αιτίες της.

Μέχρι το τέλος του 18ου αιώνα, οι ιατροί χρησιμοποιούσαν ότι άκουγαν, έπιαναν, έβλεπαν για να στηρίζουν τις παρατηρήσεις τους. Τότε ήταν που άρχισαν να εφευρίσκονται τα διάφορα διαγνωστικά όργανα και έτσι άρχισε να αναπτύσσεται μια καινούρια ορολογία προκειμένου να εκφραστούν τα καινούρια ευρήματα των οργάνων αυτών. Όπως είναι φυσικό , η πρόοδος της τεχνολογίας έφερε και την περαιτέρω επέκταση του ιστορικού του ασθενούς, πέρα από την ιστορία που διηγούνταν μέχρι τότε οι ίδιοι οι ασθενείς και οι συγγενείς τους.

Συνεχίζοντας την ιστορική αναδρομή , μετά την προτυποποίηση του ιστορικού των ασθενών, τα γραπτά ήταν ένα μείγμα από αποτελέσματα εξετάσεων, σκέψεων, θεραπευτικών πλάνων και ευρημάτων και άλλες άναρχες σημειώσεις. Έτσι σε περίπτωση που οι ασθενείς παρακολουθούνταν για παραπάνω από μια ασθένεια, η παραπάνω μέθοδος δε βοηθούσε ιδιαίτερα.

Πηγαίνοντας πιο σύγχρονα, στο 1960, ο Weed βελτίωσε την οργάνωση του ιστορικού εισάγοντας το πρόβλημα προσανατολισμένο στο ιατρικό ιστορικό σύμφωνα με το οποίο σε κάθε ασθενή αποδίδεται ένα ή περισσότερα προβλήματα. Οι σημειώσεις καταγράφονται για κάθε πρόβλημα χωριστά σύμφωνα με τη δομή SOAP

i. Subjective (Υποκείμενο)
ii. Objective (Αντικείμενο)
iii. Assessment (Αξιολόγηση)
iv. Plan (Θεραπεία)

Πέρα από την περαιτέρω βελτίωση στην προτυποποίηση και διάταξη του ιστορικού του ασθενή, κύριος στόχος του SOAP είναι να αναπαραστήσει καλύτερα τη γραμμή κρίσης και λήψης αποφάσεων του θεράποντα. Αν και το πρόβλημα προσανατολισμένο στο ιατρικό ιστορικό έγινε αποδεκτό, στην πράξη παρατηρήθηκε ότι τα δεδομένα που σχετίζονται με περισσότερα από ένα πρόβλημα πρέπει να καταγράφονται αρκετές φορές.

Στην εποχή μας, ο σύγχρονος φάκελος ασθενούς διαφέρει αρκετά από τα παραπάνω, καθώς χωρίζεται σε μέρη βάσει της πηγής δεδομένων. Αρχικά, τα δεδομένα που προκύπτουν από τις ιατρικές συνεδρίες αποτελούν το πρώτο μέρος του φακέλου, ενώ τα δεδομένα που προέρχονται από εργαστηριακές εξετάσεις συνθέτουν το δεύτερο μέρος του. Υπάρχει άλλο ένα μέρος, το τρίτο, το οποίο περιλαμβάνει γνωματεύσεις από ακτινογραφίες, αξονικές τομογραφίες κι υπερήχους. Όλα τα δεδομένα που συλλέγονται κατατάσσονται χρονολογικά σε κάθε μέρος. Από τα παραπάνω, γίνεται σαφές ότι η δομή του φακέλου βάσει του χρόνου και του προβλήματος του ασθενούς εξακολουθεί να πραγματοποιείται. Δυστυχώς, όμως, δεν λείπουν οι περιπτώσεις που πολλοί χειρόγραφοι φάκελοι βρίσκονται εγκαταλελειμένοι σε ράφια αποθηκών και αγνοούνται ή ενώ το προσωπικό γνωρίζει ακριβώς που βρίσκονται, η κατάστασή τους είναι τόσο κακή, που είναι δύσκολη ακόμα και η ανάγνωσή τους. Κι ακόμη κι αν η ανάγνωσή τους είναι δυνατή, το περιεχόμενό τους είναι τόσο περιττό και ανακριβές, οι πληροφορίες τόσο ανοργάνωτες και λάθος ταξινομημένες, με αποτέλεσμα να μην είναι καθόλου χρηστικός και να δυσχεραίνει το έργο του ιατρού.

ΤΙ ΣΚΟΠΟΥΣ ΕΞΥΠΗΡΕΤΕΙ Ο ΙΑΤΡΙΚΟΣ ΦΑΚΕΛΟΣ

1. Είναι ένα μέσο επικοινωνίας ανάμεσα στο γιατρό και τον ασθενή. Οδηγίες, θεραπείες, διαγνώσεις , παραπεμπτικά, καταγραφή πορείας νόσου δρομολογούνται και επικοινωνούνται στους διαφόρους εμπλεκομένους.
2. Αποτελεί το σημείο αναφοράς στο οποίο ανατρέχει κάποιος για να έχει μια εικόνα της κατάστασης του ασθενούς.
3. Ανεπίσημα, ο ιατρικός φάκελος χρησιμεύει και ως χώρος εργασίας όπου καταγράφονται ιδέες και εντυπώσεις για το πρόβλημα του ασθενούς, καθώς και την πορεία εξέλιξης του προβλήματος.
4. Αποτελεί ένα μέρος που φυλάσσονται όλα τα κλινικά δεδομένα για μελλοντική χρήση, είτε για περαιτέρω θεραπεία του ασθενούς, είτε για παράδειγμα κλινική έρευνα ή επιδημιολογική μελέτη.
5. Μπορεί να χρησιμεύσει για μετέπειτα έλεγχο των διαδικασιών που ακολουθήθηκαν κατά τη διάρκεια της θεραπείας του ασθενή, για παράδειγμα σε υποψία ιατρικού λάθους.

ΜΕΙΟΝΕΚΤΗΜΑΤΑ ΚΛΑΣΙΚΟΥ ΧΕΙΡΟΓΡΑΦΟΥ ΙΑΤΡΙΚΟΥ ΦΑΚΕΛΟΥ

Αναμφισβήτητα, η λύση του κλασσικού χειρόγραφου ιατρικού φακέλου έχει χρησιμοποιηθεί με σχετική επιτυχία. Αυτή η μέθοδος χρησιμοποιεί ως βάση το χαρτί με το οποίο είναι εξοικειωμένοι οι περισσότεροι άνθρωποι. Επιπλέον στην εξοικείωση του ανθρώπου με το χαρτί και τη γραφή, αξίζει να αναφέρουμε την ευκολία μεταφοράς του και το αυτόνομο της μεθόδου.(π. χ δε χρειαζόμαστε πρίζα, ρεύμα, ή υπολογιστή για να ανακτήσει κανείς το περιεχόμενο του φακέλου, γνώσεις χρήσης ηλεκτρονικού υπολογιστή).

Το χαρτί όμως ως υλικό έχει κάποια σημαντικά μειονεκτήματα:

- ✓ Μπορεί να καταστραφεί εύκολα, ενώ είναι αρκετά επίπονη η διαδικασία της δημιουργίας αντιγράφων ασφαλείας.
- ✓ Φθείρεται με τη χρήση ή τον χρόνο, επομένως έχει περιορισμένο χρόνο ζωής.
- ✓ Είναι διαθέσιμο μόνο σε ένα μέρος την ίδια στιγμή.

Παρόλα αυτά, τα σημαντικότερα προβλήματα που εμφανίζονται από τη χρήση ενός χάρτινου ιατρικού φακέλου δεν οφείλονται στο βασικό χρησιμοποιούμενο υλικό αλλά στο τι συνεπάγεται. Όταν για παράδειγμα ο ιατρός έχει μπροστά του μια κόλλα χαρτί, κυριολεκτικά μπορεί να γράψει το οτιδήποτε. Αυτό που θα γράψει δεν είναι βέβαιο ότι θα είναι χρησιμοποιήσιμο από αυτούς που θα το διαβάσουν. Για παράδειγμα, μπορεί ο γραφικός χαρακτήρας να είναι δυσδιάκριτος ή χειρότερα ο συγγραφέας να έχει παραλείψει σημαντικά στοιχεία που δεν τον απασχολούν αλλά είναι καίριας σημασίας για τον αναγνώστη. Δεν οφείλονται στο χαρτί και τη μορφή του έντυπου φακέλου φυσικά αυτά αλλά στην προσέγγιση που ακολουθείται για τη συμπλήρωση του ηλεκτρονικού φακέλου. Τυχαίο δεν είναι καθόλου το γεγονός ότι τέτοιας φύσεως προβλήματα αφορούν αποκλειστικά σχεδόν σε κλασσικούς χάρτινους φακέλους ασθενών. Αρκετές μελέτες έχουν δείξει πως οι ιατρικοί φάκελοι είναι απροσπέλαστοι σε ποσοστό περίπου 30% καθώς επίσης το περιεχόμενο τους είναι διασκορπισμένο σε νοσοκομεία, γραφεία, ιατρεία, διαγνωστικά κέντρα. Όπως αναφέρθηκε και παραπάνω, δεν είναι λίγες οι φορές που οι χειρόγραφοι ιατρικοί φάκελοι βρίσκονται τοποθετημένοι σε αποθήκες μαζί με άλλο αντικείμενα, με αποτέλεσμα η εύρεσή τους να είναι δύσκολη έως αδύνατη. Ακόμη κι αν κάποιος φάκελος βρεθεί, υπάρχει τεράστια πιθανότητα το περιεχόμενό τους να είναι σε κακή κατάσταση, δυσανάγνωστο, ανοργάνωτο και σε μεγάλο βαθμό ελλιπές.

ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΗΛΕΚΤΡΟΝΙΚΟΥ ΦΑΚΕΛΟΥ ΑΣΘΕΝΟΥΣ

- ✓ Το πρώτο βασικό πλεονέκτημα είναι η οικονομία χαρτιού. Είναι γνωστό πως η καθημερινότητά μας βασίζεται στη χρήση του, καθώς είναι πολλές οι δραστηριότητες που την απαιτούν, με αποτέλεσμα να καταστρέφονται πολλά δάση και να πλήττεται με αυτόν τον τρόπο το «οξυγόνο» του πλανήτη. Με την εφαρμογή, λοιπόν, του ηλεκτρονικού φακέλου ασθενούς, παύουν όλα να είναι με βάση το χαρτί και με αυτό τον τρόπο ωφελούμε εμάς και κατ΄ επέκτασιν το περιβάλλον μας. Η καλλιέργεια οικολογικής συνείδησης αναπτύσσεται και αυτό αποτελεί σανίδα σωτηρίας για το μέλλον του πλανήτη μας.
- ✓ Ένα ακόμη πολύ σημαντικό πλεονέκτημα είναι η ευκολία μεταφοράς του ηλεκτρονικού φακέλου. Ο ιατρός και γενικότερα το ιατρικό προσωπικό, δεν είναι απαραίτητο να κουβαλά μαζί του τις εξετάσεις των ασθενών του για να κάνει διάγνωση αλλά και να συγκρίνει τιμές ενδείξεων κατάστασης του ασθενούς. Πολύ απλά με την είσοδο του στο σύστημα μπορεί να έχει πρόσβαση ανά πάσα στιγμή στο ιστορικό του ασθενούς και να εξάγει τα συμπεράσματά του, κάτι που μπορεί να γίνει με τη χρήση οποιασδήποτε ηλεκτρονικής συσκευής – ενός laptop, ενός tablet, ακόμα και μέσω ενός κινητού τηλεφώνου, εφόσον η σύγχρονη τεχνολογία έχει επιτρέψει να έχουμε στα χέρια μας μια συσκευή με πάρα πολλές δυνατότητες.
- ✓ Επιπλέον, ένα πλεονέκτημα είναι το χαμηλό κόστος της δημιουργίας του ηλεκτρονικού φακέλου, καθώς και το εύκολο format του. Ο ιατρός, όποτε το επιθυμεί, μπορεί να διαγράφει δεδομένα

ασθενούς, να κάνει διορθώσεις πιθανώς σε προσωπικά στοιχεία, να μεταβάλλει τιμές και ενδείξεις. Γενικότερα, να κάνει πολύ απλά και αποτελεσματικά update.

- ✓ Επιπλέον, ο χρήστης μπορεί πολύ γρήγορα να προσθέτει στοιχεία, ακόμη και κατά τη διάρκεια του ραντεβού του με τον ασθενή, με αποτέλεσμα να εξοικονομεί χρόνο και να είναι πλήρως ενημερωμένος για το ιστορικό αλλά και τις τελευταίες εξελίξεις της υγείας του ασθενούς του μόλις μετά το πέρας της συνεδρίας τους. Μπορεί μάλιστα να δημιουργεί και να εισάγει άλλον ασθενή οποιαδήποτε στιγμή.
- ✓ Ο χρήστης έχει τη δυνατότητα να παρακολουθεί τη φαρμακευτική αγωγή των ασθενών του, να κάνει αν επιθυμεί αλλαγές, διορθώσεις και βελτιώσεις. Το σύστημα αποθηκεύει το ιστορικό των φαρμάκων του ασθενούς και μ΄ αυτό τον τρόπο μπορεί οποιαδήποτε στιγμή να ελέγξει την αγωγή και να διαπιστώσει κατά πόσο είναι αποτελεσματική.

Επιπροσθέτως ο ιατρός με την είσοδο του στο σύστημα, μπορεί να δει στατιστικά εισαγωγών, εξιτηρίων καθώς και θανάτων που αναφέρονται στο σύνολο των ημερών είτε για κάποιες x ημέρες. Με αυτό τον τρόπο μπορεί να διεξάγει τα συμπεράσματά του για κάποιους χειρισμούς του στην αντιμετώπιση των ασθενών, για διαγνώσεις του και για την παρακολούθηση εξέλιξης των ασθενών του.

ΦΑΚΕΛΟΣ ΑΣΘΕΝΟΥΣ ΚΑΙ ΗΘΙΚΗ

Είναι ξεκάθαρο ότι το δικαίωμα του ασθενούς για διασφάλιση της εμπιστευτικότητας των προσωπικών του δεδομένων . Αυτό είναι κάτι το οποίο δεν μπορεί να υποβιβασθεί εξαιτίας της χρήσης του ηλεκτρονικού φακέλου υγείας. Ο καθορισμός ηθικών, νομικών διαδικασιών, κριτηρίων και πλαισίων όσον αφορά στην ηλεκτρονική συλλογή, επεξεργασία και διακίνηση των προσωπικών ευαίσθητων δεδομένων ασθενών σε πιθανούς χρήστες δεδομένων υγείας είναι απαραίτητος, αφού τυχόν αποκάλυψή τους θέτει σε κίνδυνο την σχέση τόσο ιατρού ή νοσηλευτή - ασθενή, όσο και των μελών ολόκληρης της κοινωνίας αφού είναι πιθανό από τον φόβο αποκάλυψης τους, ο ασθενής να μην εμπιστευθεί κρίσιμες πληροφορίες που αφορούν όχι μόνο στην υγεία του αλλά και στην διατήρηση της δημόσιας υγείας.

ΚΕΦΑΛΑΙΟ 2 ΟΡΙΣΜΟΣ ΗΛΕΚΤΡΟΝΙΚΟΥ ΦΑΚΕΛΟΥ

Ο Ηλεκτρονικός Φάκελος Ασθενούς είναι, όπως δηλώνει και η ονομασία του, ένα ηλεκτρονικό ανάλογο των υπαρχόντων αρχείων ασθενών που επισκέπτονται τα νοσοκομεία. Πρόκειται για μια συστηματοποιημένη συλλογή του ιστορικού και της κατάστασης υγείας του ασθενούς, ο οποίος δημιουργείται και συντηρείται από έναν ιατρό, μια Μονάδα υγείας ή άλλον επαγγελματία φροντίδας υγείας. Σύμφωνα με την Ευρωπαϊκή Επιτροπή Προτυποποίησης, ο Ιατρικός Φάκελος είναι «η αποθήκη όλων των πληροφοριών που αφορούν στο ιατρικό ιστορικό του ασθενούς, έτσι ώστε να αποτελεί τη βάση της διάγνωσης και της θεραπευτικής αντιμετώπισης του ασθενούς αλλά και τη βάση επιδημιολογικών ερευνών. Επιπλέον, παρέχει πληροφορίες διοικητικής, οικονομικής και στατιστικής φύσεως, καθώς και ποιοτικού ελέγχου».

Πιο συγκεκριμένα, ο Ηλεκτρονικός Ιατρικός Φάκελος (ΗΙΦ) ή γενικότερα, ο Ηλεκτρονικός Φάκελος Υγείας (ΗΦΥ) [Electronic Medical Record (EMR), Electronic Health Record (HER), Computer- based Patient Record (CPR), Computer- based Health Record (CHR)] ενός ασθενούς είναι «όλες οι πληροφορίες οι σχετιζόμενες με τη φυσική/ ψυχική υγεία ή κατάσταση ενός ασθενούς στο παρελθόν, παρόν και μέλλον, οι οποίες καταγράφονται (ψηφιακά) σε ηλεκτρονικό σύστημα καταλλήλως, ώστε να επεξεργάζονται στους Ηλεκτρονικούς Υπολογιστές (και, κυρίως, με τη βοήθεια πολυμέσων) και να κυκλοφορούν στο Διαδίκτυο, με πρωταρχικό σκοπό πάντοτε την υγειονομική περίθαλψη και φροντίδα του ασθενούς». Στα συστήματα υγείας διαφόρων κρατών δεν υπάρχει ομοφωνία ως προς την έννοια του Ηλεκτρονικού Ιατρικού Φακέλου, αφού αποδίδεται με ποικίλες ερμηνείες. Για παράδειγμα, άλλοτε θεωρείται αντίγραφο του χειρόγραφου φακέλου μέσω διαδικασιών scanner (EMR), άλλοτε ως αυτοματοποιημένος εργαστηριακός (LMR) και άλλοτε ως Ηλεκτρονικός φάκελος Υγείας (ΗΦΥ ή EHR).

Τα τελευταία χρόνια στην Ευρώπη χρησιμοποιείται όλο και περισσότερο ο όρος Φάκελος Υγείας του Πολίτη (ΦΥΠ) [Citizen Health Record (CHR)]. Ο όρος αυτός είναι ο αντιπροσωπευτικότερος από τους προηγούμενους όρους και δηλώνει με περισσότερη ακρίβεια το σύγχρονο όραμα του παγκόσμιου πολίτη ως προς τις απαιτήσεις του από τις υπηρεσίες υγείας. Ο Φάκελος Υγείας του Πολίτη υπερκαλύπτει την (ψηφιακή) καταγραφή και συντήρηση του περιεχομένου του ιατρικού φακέλου και επιπλέον αντιμετωπίζει επιτυχώς όλα τα προβλήματα που προκύπτουν από την ηλεκτρονική φύση του. Στην Ελλάδα εξακολουθεί να χρησιμοποιείται ευρέως ο όρος Ηλεκτρονικός Ιατρικός Φάκελος.

ΚΕΦΑΛΑΙΟ 3 ΤΕΧΝΟΛΟΓΙΕΣ ΤΕΧΝΙΚΗΣ ΛΥΣΗΣ

Η δομή της εργασίας μπορεί να αναλυθεί στην διεπαφή με τον χρήστη (Graphic User Interface) και στις διεργασίες οι οποίες τρέχουν στο παρασκήνιο στον εξυπηρετητή (server). Η διεπαφή με τον χρήστη έχει υλοποιηθεί με την χρήση της HTML5 για την δομημένη απεικόνιση της πληροφορίας και με JavaScript για την εισαγωγή δυναμικών στοιχείων. Στο παρασκήνιο του προγράμματος χρησιμοποιήθηκαν τεχνολογίες οι οποίες εξειδικεύονται στην αποθήκευση, ανάκτηση και διαχείριση με ασφαλή τρόπο της πληροφορίες αλλά και για την διασύνδεση με τον τελικό χρήστη. Πιο αναλυτικά, έχει χρησιμοποιηθεί η H2 Database για την αποθήκευση της πληροφορίας και η Spring Boot JPA για να γίνεται η διαχείρισή τους αποδοτικά. Τέλος με την χρήση της Spring Data Rest επιτυγχάνεται η ανταλλαγή της πληροφορίας και με την Spring Boot OAuth02 εξασφαλίζεται η ασφάλεια της επικοινωνίας .

3.1 Back end

3.1.1 Spring Framework

Το Spring Framework είναι ένα πλαίσιο ανοιχτού κώδικα για τη δημιουργία εταιρικών εφαρμογών σε Java. Η Spring στοχεύει να απλοποιήσει την πολύπλοκη διαδικασία ανάπτυξης εφαρμογών java για επιχειρήσεις, προσφέροντας ένα πλαίσιο που περιλαμβάνει τεχνολογίες όπως προγραμματισμός προσανατολισμένος στις πτυχές (Aspect-Oriented Programming) και έγχυση εξάρτησης (Discrepancy Injection)

3.1.2 Spring Boot JPA

Ορισμός JPA (Java Persistence API)

JPA είναι μια συλλογή από κλάσεις και μεθόδους ώστε να αποθηκεύονται μεγάλου όγκου δεδομένα σε βάσεις με την τεχνική του persistence (εναπόθεση αντιγράφων αντικειμένων της βάσης στην προσωρινή μνήμη).

Πολλές λύσεις , χειρίζονται δεδομένα βάσεων είτε ανακτώντας τα είτε αποθηκεύοντας τα σε αυτές. Είναι κοινό μυστικό ότι οι προγραμματιστές δυσκολεύονται να πραγματοποιήσουν διεργασίες με τη βάση δεδομένων αποδοτικά μιας και αυτό προαπαιτεί αρκετή χρήση κώδικα. Με την χρήση της τεχνολογίας JPA , οι χρόνοι που χρειάζονται για την αλληλεπίδραση με τη βάση δεδομένων μειώνονται αισθητά. Δημιουργείται μια γέφυρα μεταξύ των μοντέλων αντικειμένων (object models) και τα σχεσιακά μοντέλα της βάσης δεδομένων (relational model)

Spring Boot JPA

Μέρος της οικογένειας του Spring Boot , διευκολύνει την υλοποίηση repositories βασισμένα σε JPA.Ασχολείται με τη βελτιωμένη υποστήριξη για επίπεδα πρόσβασης δεδομένων που βασίζονται σε JPA

Μερικά από τα χαρακτηριστικά του Spring Boot JPA:

- Εξελιγμένη υποστήριξη για τη δημιουργία repositories με βάση το Spring και το JPA
- Υποστήριξη για κατηγορήματα και επομένως ερωτήματα JPA ασφαλή για τον τύπο
- Υποστήριξη σελιδοποίησης, δυναμική εκτέλεση ερωτήματος, δυνατότητα ενσωμάτωσης προσαρμοσμένου κώδικα πρόσβασης δεδομένων

Πλεονεκτήματα:

- Δημιουργία Spring εφαρμογών οι οποίες είναι εύκολες στην κατανόηση και υλοποίηση
- Αυξάνει την παραγωγικότητα
- Μειώνει τον χρόνο προγραμματισμού

3.1.3 Spring Data Rest

Το Spring Data REST είναι μέρος του Spring Data και διευκολύνει τη δημιουργία υπηρεσιών web REST πάνω από τα αποθετήρια Spring Data.

Το Spring Data REST βασίζεται σε αποθετήρια Spring Data, αναλύει το μοντέλο τομέα της εφαρμογής και εκθέτει πόρους HTTP που βασίζονται σε υπερμέσα για συγκεντρωτικά στοιχεία που περιέχονται στο μοντέλο.

Χαρακτηριστικά

- Εκθέτει ένα ανιχνεύσιμο REST API για το μοντέλο του τομέα σας χρησιμοποιώντας το HAL ως τύπο μέσου.
- Εκθέτει πόρους συλλογής, αντικειμένων και συσχετίσεων που αντιπροσωπεύουν το μοντέλο σας.
- Υποστηρίζει σελιδοποίηση μέσω συνδέσμων πλοήγησης .
- Επιτρέπει το δυναμικό φιλτράρισμα πόρων συλλογής.
- Εμφανίζει αποκλειστικούς πόρους αναζήτησης για μεθόδους ερωτημάτων που ορίζονται στα αποθετήρια σας.
- Επιτρέπει τη σύνδεση στο χειρισμό των αιτημάτων REST με το χειρισμό του Spring ApplicationEvents.
- Εκθέτει μεταδεδομένα σχετικά με το μοντέλο που ανακαλύφθηκε ως ALPS και JSON Schema.
- Επιτρέπει τον ορισμό συγκεκριμένων αναπαραστάσεων πελάτη μέσω προβολών .
- Αποστέλλει μια προσαρμοσμένη παραλλαγή του HAL Explorer για να αξιοποιήσει τα εκτεθειμένα μεταδεδομένα.
- Προς το παρόν υποστηρίζει JPA, MongoDB, Neo4j, Solr, Cassandra, Gemfire.

- Επιτρέπει προηγμένες προσαρμογές των προεπιλεγμένων πόρων που εκτίθενται.

3.1.4 Spring Boot OAuth02 Resource Server

Το Spring Security παρέχει ολοκληρωμένη υποστήριξη OAuth2. Υποστηρίζει την προστασία των endpoints χρησιμοποιώντας δύο μορφές OAuth2.0 Bearer Token:

- JWT
- Opaque Tokens

Αυτό είναι βολικό σε περιπτώσεις όπου μια εφαρμογή έχει εκχωρήσει τη διαχείριση εξουσιών σε έναν authorization server (για παράδειγμα, Okta ή Ping Identity). Οι διακομιστές πόρων μπορούν να συμβουλευτούν αυτόν τον διακομιστή εξουσιοδότησης για την εξουσιοδότηση αιτημάτων.

JWT :

Το JWT Token , χρησιμοποιείται για να εξουσιοδοτήσει(authorize) τον χρήστη που έχει ήδη πιστοποιηθεί(authenticated) στο σύστημα . Ελέγχει με άλλα λόγια αν ο πιστοποιημένος χρήστης , είναι ο ίδος με τον χρήστη που προσπαθεί να εξουσιοδοτηθεί για μια κλήση .

Πως λειτουργούν τα JWT:

Κατά τον έλεγχο ταυτότητας, όταν ο χρήστης συνδεθεί με επιτυχία χρησιμοποιώντας τα διαπιστευτήριά του, θα επιστραφεί ένα JSON Web Token. Δεδομένου ότι τα διακριτικά είναι διαπιστευτήρια .

Κάθε φορά που ο χρήστης θέλει να αποκτήσει πρόσβαση σε μια προστατευμένη διαδρομή ή πόρο, ο παράγοντας χρήστη θα πρέπει να στέλνει το JWT, συνήθως στην κεφαλίδα Εξουσιοδότηση χρησιμοποιώντας το σχήμα Bearer .

3.1.5 H2 Database

Η H2 είναι μια βάση δεδομένων την Java . Είναι ενσωματωμένη στο Spring , κάτι που το κάνει εύκολο και γρήγορο στη χρήση .

Μερικά από τα χαρακτηριστικά της H2 :

- Πολύ γρήγορο, ανοιχτού κώδικα, JDBC API
- Ενσωματωμένες λειτουργίες και λειτουργίες διακομιστή. Βάσεις δεδομένων που βασίζονται σε δίσκο ή στη μνήμη
- Υποστήριξη συναλλαγών, συγχρονισμός πολλαπλών εκδόσεων
- Παρέχει κρυπτογραφημένες βάσεις δεδομένων

3.2 GUI

3.2.1 JavaScript

Η JavaScript, είναι μια γλώσσα προγραμματισμού που είναι μια από τις βασικές τεχνολογίες του Παγκόσμιου Ιστού , μαζί με την HTML και το CSS . Το μεγαλύτερο μέρος των ιστότοπων χρησιμοποιούν JavaScript στην πλευρά του πελάτη για συμπεριφορά ιστοσελίδων , συχνά ενσωματώνοντας βιβλιοθήκες τρίτων . Όλα τα μεγάλα προγράμματα περιήγησης ιστού διαθέτουν μια αποκλειστική μηχανή JavaScript για την εκτέλεση κώδικα στις συσκευές των χρηστών .

3.2.2 HTML5

HyperText Markup Language ή αλλιώς HTML είναι η τυπική γλώσσα σήμανσης για έγγραφα που έχουν σχεδιαστεί για εμφάνιση σε πρόγραμμα περιήγησης ιστού. Μπορεί να υποστηριχθεί από τεχνολογίες όπως τα Cascading Style Sheets (CSS) και γλώσσες δέσμης ενεργειών όπως η JavaScript .

Τα προγράμματα περιήγησης Ιστού λαμβάνουν έγγραφα HTML από έναν διακομιστή ιστού ή από τοπική αποθήκευση και αποδίδουν τα έγγραφα σε ιστοσελίδες πολυμέσων. Η HTML περιγράφει τη δομή μιας ιστοσελίδας σημασιολογικά και αρχικά περιλάμβανε ενδείξεις για την εμφάνιση του εγγράφου.

ΚΕΦΑΛΑΙΟ 4 Σχεδιασμός Τεχνικής Λύσης

4.1 Γενική περιγραφή τεχνικής λύσης

Η λύση, η οποία προτείνεται για τη λειτουργία του Ιατρικού Φακέλου , θα αποτελείται από τα παρακάτω μέρη:

- **Back end application:** Δημιουργία εφαρμογής με σε Java η οποία θα συλλέγει, τροποποιεί δεδομένα ασθενών
- **GUI:** Υλοποίηση του περιβάλλοντος διεπαφής του χρήστη με χρήση τεχνολογιών Javascript και HTML5

4.2 Διάγραμμα τεχνικής λύσης

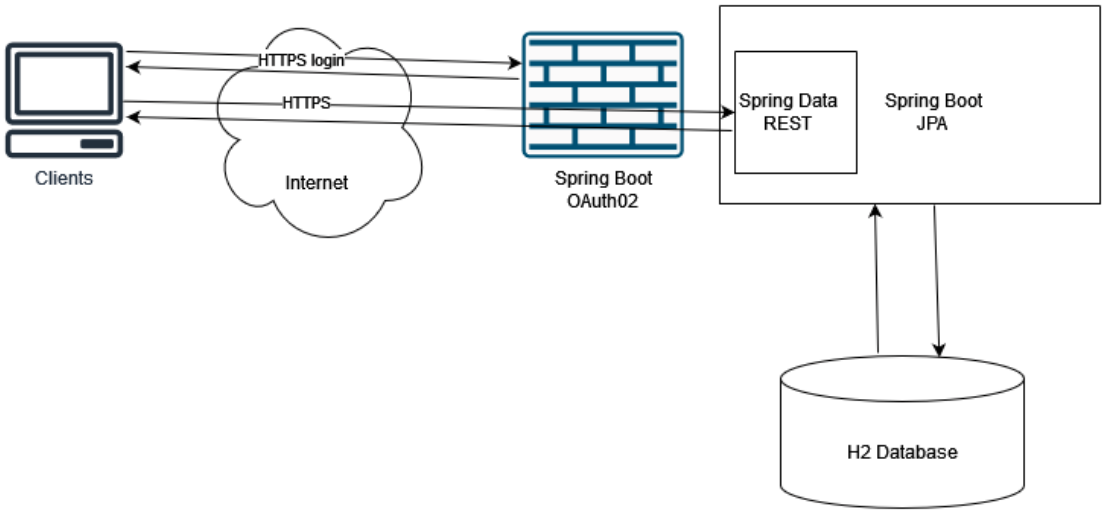

Figure 1: Διάγραμμα τεχνικής λύσης

4.3 Database Relation Diagram

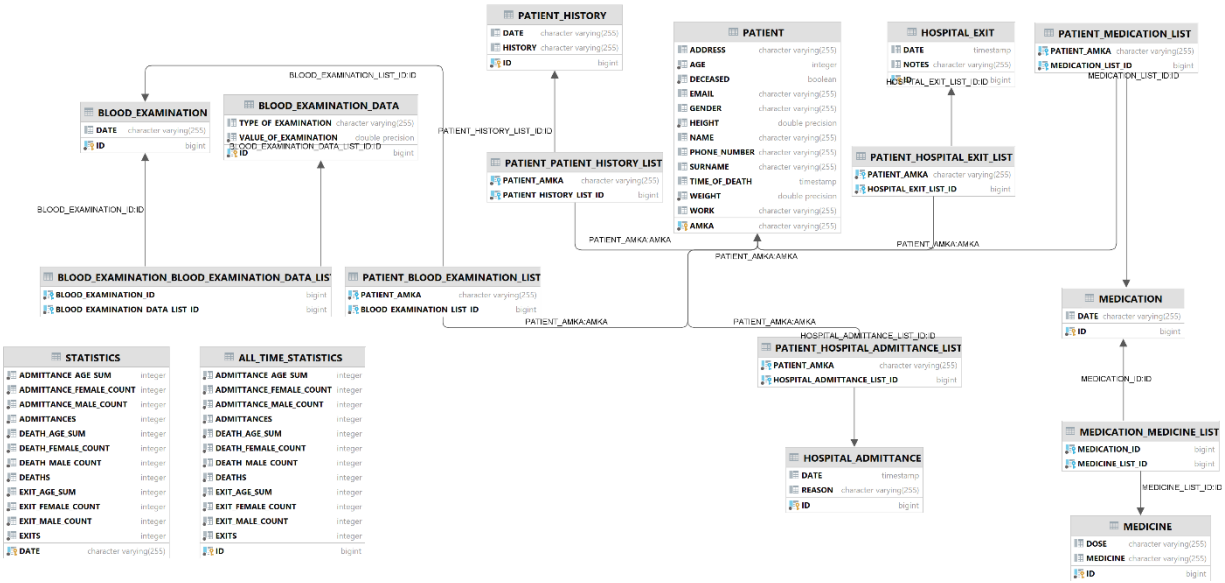

Figure 2: Σχεσιακό διάγραμμα βάσης δεδομένων

4.4 Πίνακες βάσης δεδομένων

4.4.1 Πίνακας Patient

Στον πίνακα Patient , αποθηκεύονται οι βασικές πληροφορίες για τον ασθενή.

PATIENT	
ADDRESS	character varying(255)
AGE	integer
DECEASED	boolean
EMAIL	character varying(255)
GENDER	character varying(255)
HEIGHT	double precision
NAME	character varying(255)
PHONE_NUMBER	character varying(255)
SURNAME	character varying(255)
TIME_OF_DEATH	timestamp
WEIGHT	double precision
WORK	character varying(255)
AMKA	character varying(255)

Figure 3: Πίνακας Patient

4.4.2 Πίνακας Hospital Admittance

Ο πίνακας αυτός , περιέχει τις πληροφορίες σχετικά με την ημέρα εισιτηρίου και τον λόγο αυτού.

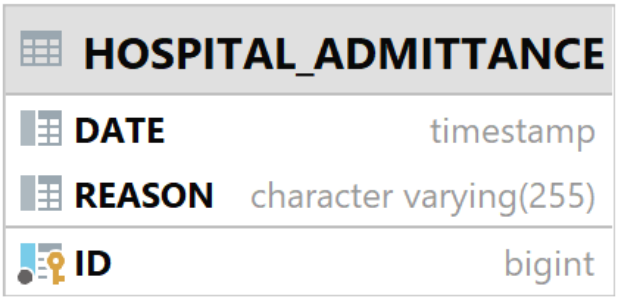

Figure 4: Πίνακας Hospital_Admittance

Η σύνδεση του εκάστοτε εισιτηρίου με κάποιον ασθενή γίνεται με τον πίνακα **Patient_Hospital_Admittance_List:**

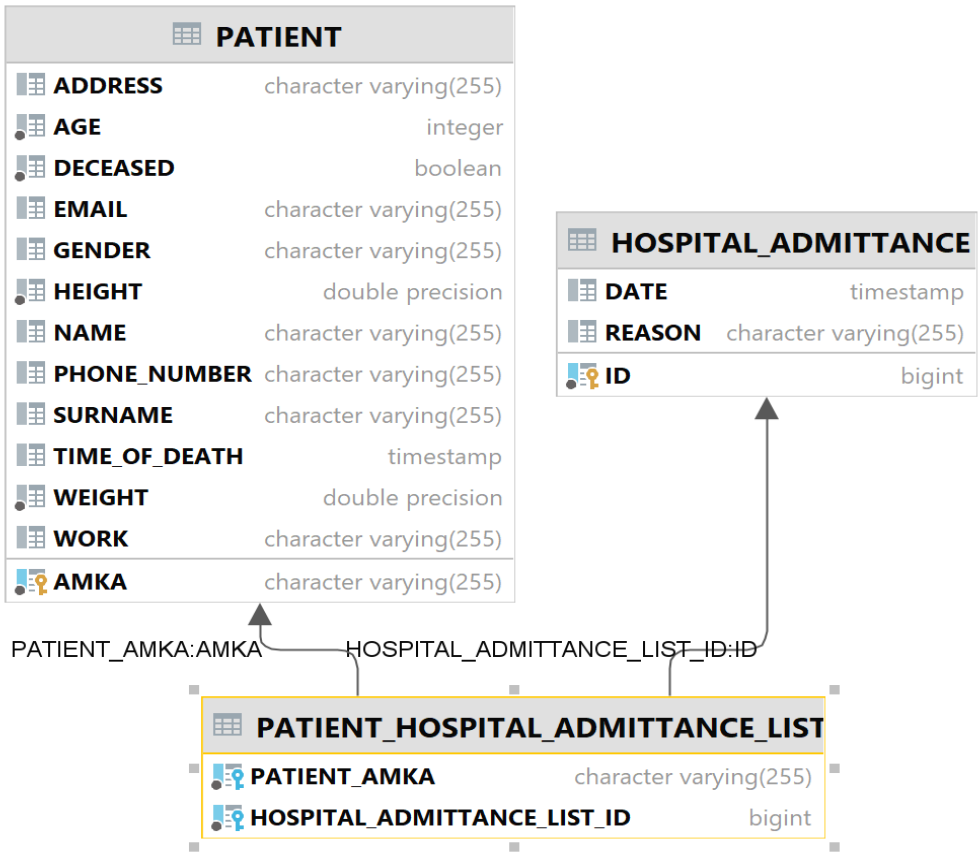

Figure 5: Σύνδεση πίνακα Patient με τον πίνακα Patient_Admittance

4.4.3 Πίνακας Hospital Exit

Τα εξιτήρια των ασθενών από την ιατρική μονάδα , εγγράφονται στον πίνακα αυτόν. Πληροφορίες που περιέχονται σε αυτόν τον πίνακα είναι η ημερομηνία του εξιτηρίου καθώς και κάποιες οδηγίες προς τον ασθενή.

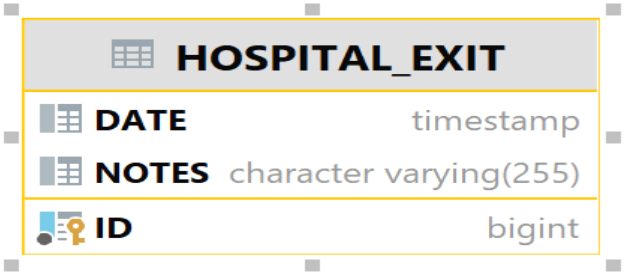

Figure 6: Πίνακας Hospital_Exit

Για να γίνει η σύνδεση μεταξύ των εξιτηρίων και των ασθενών στους οποίου ανήκουν , έχουμε τον πίνακα **Patient Hospital Exit List:**

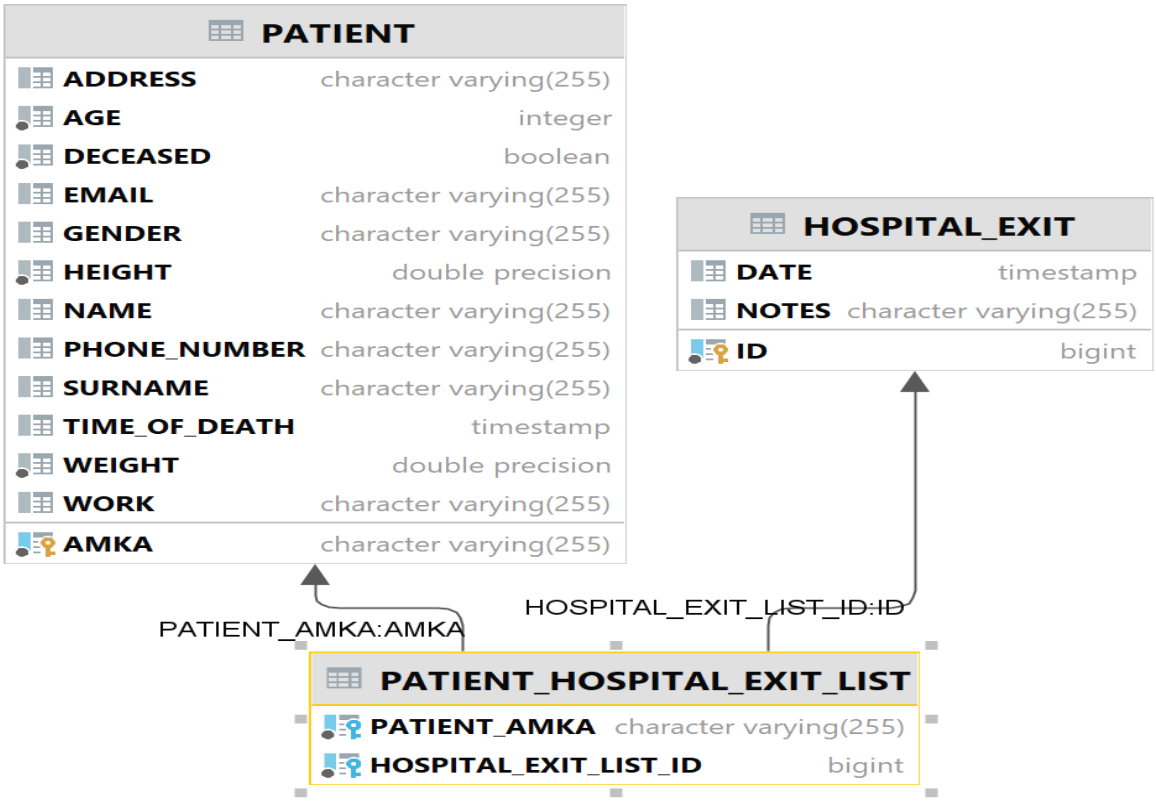

Figure 7: Σύνδεση πίνακα Patient με τον πίνακα Patient_Exit

4.4.4 Πίνακας Blood Examination

Όταν εισάγουμε στο σύστημα μια σειρά από εξετάσεις αίματος , μπαίνει μια νέα εγγραφή σε αυτόν τον πίνακα .

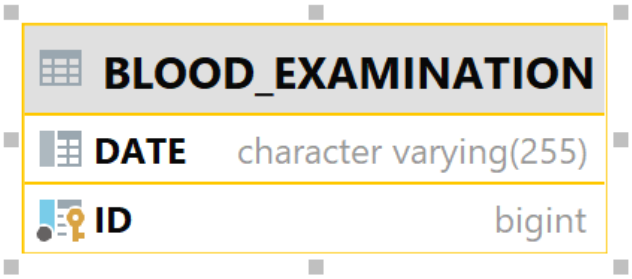

Figure 8: Πίνακας Blood_Examination

Για κάθε στοιχείο αιματολογική εξέταση ξεχωριστά , εισάγονται οι πληροφορίες στον πίνακα **Blood Examination Data:**

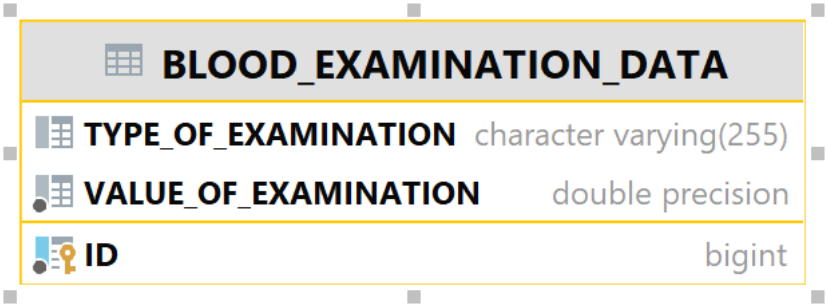

Figure 9: Πίνακας Blood_Examination_Data

Η σύνδεση μεταξύ αιματολογικής εξέτασης και των επιμένουν στοιχείων της , γίνεται με τον πίνακα **Blood Examination Blood Examination Data List:**

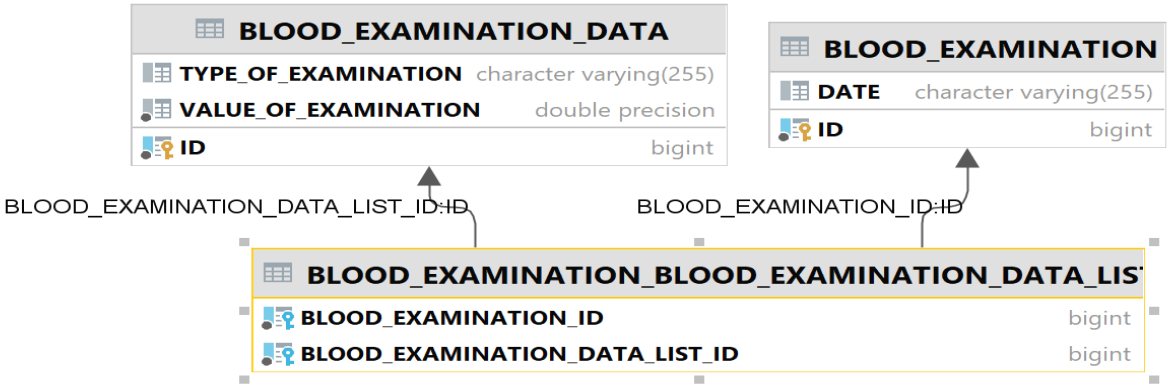

Figure 10: Σύνδεση πίνακα Blood_Examination_Data με τον πίνακα Blood_Examination

4.4.5 Πίνακας Medication

Οι ασθενείς , κυρίως κατά την είσοδο τους στην ιατρική μονάδα , καλούνται να πληροφορήσουν το προσωπικό για την φαρμακευτική αγωγή που ακολουθούν. Οι απαντήσεις τους αποθηκεύονται στον πίνακα Medication.

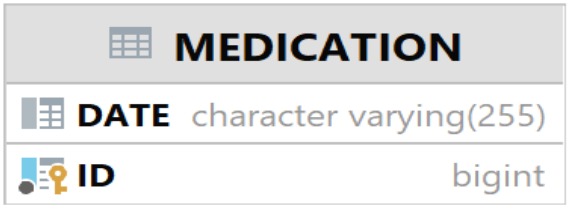

Figure 11: Πίνακας Medication

Κάθε δραστική ουσία , καθώς και η δοσολογία που χορηγείται αποθηκεύεται στον πίνακα **Medicine**:

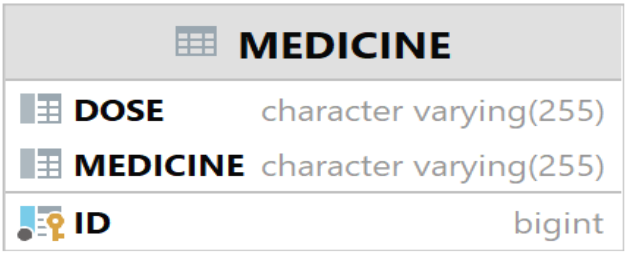

Figure 12: Πίνακας Medicine

Για την ανάκτηση των ζευγών δραστικη ουσία – δόση στη κάθε φαρμακευτική αγωγή χρησιμοποιείται ο πίνακας **Medication Medicine List:**

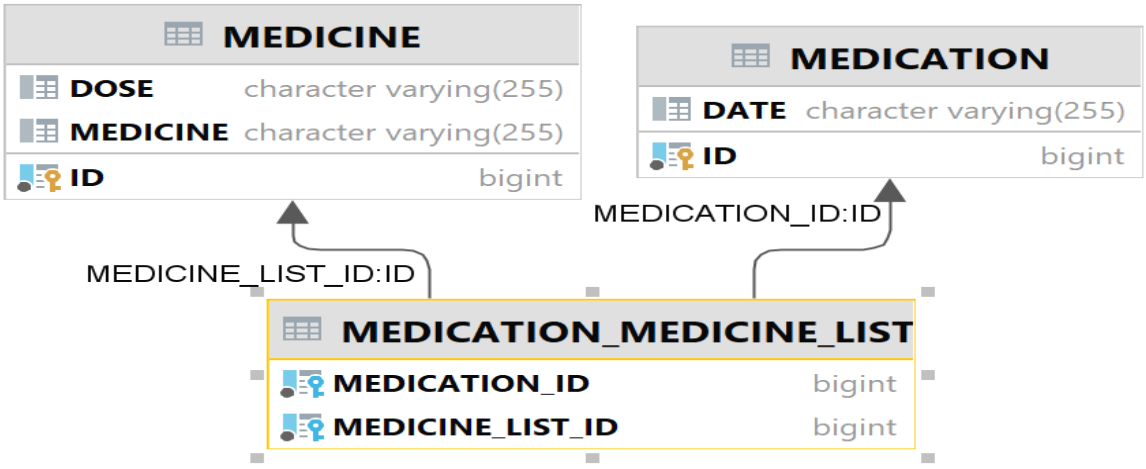

Figure 13: Σύνδεση πίνακα Medicine με τον πίνακα Medication

Η σύνδεση φαρμακευτικής αγωγής και ασθενούς επιτυγχάνεται μέσω του πίνακα **Patient Medication List:**

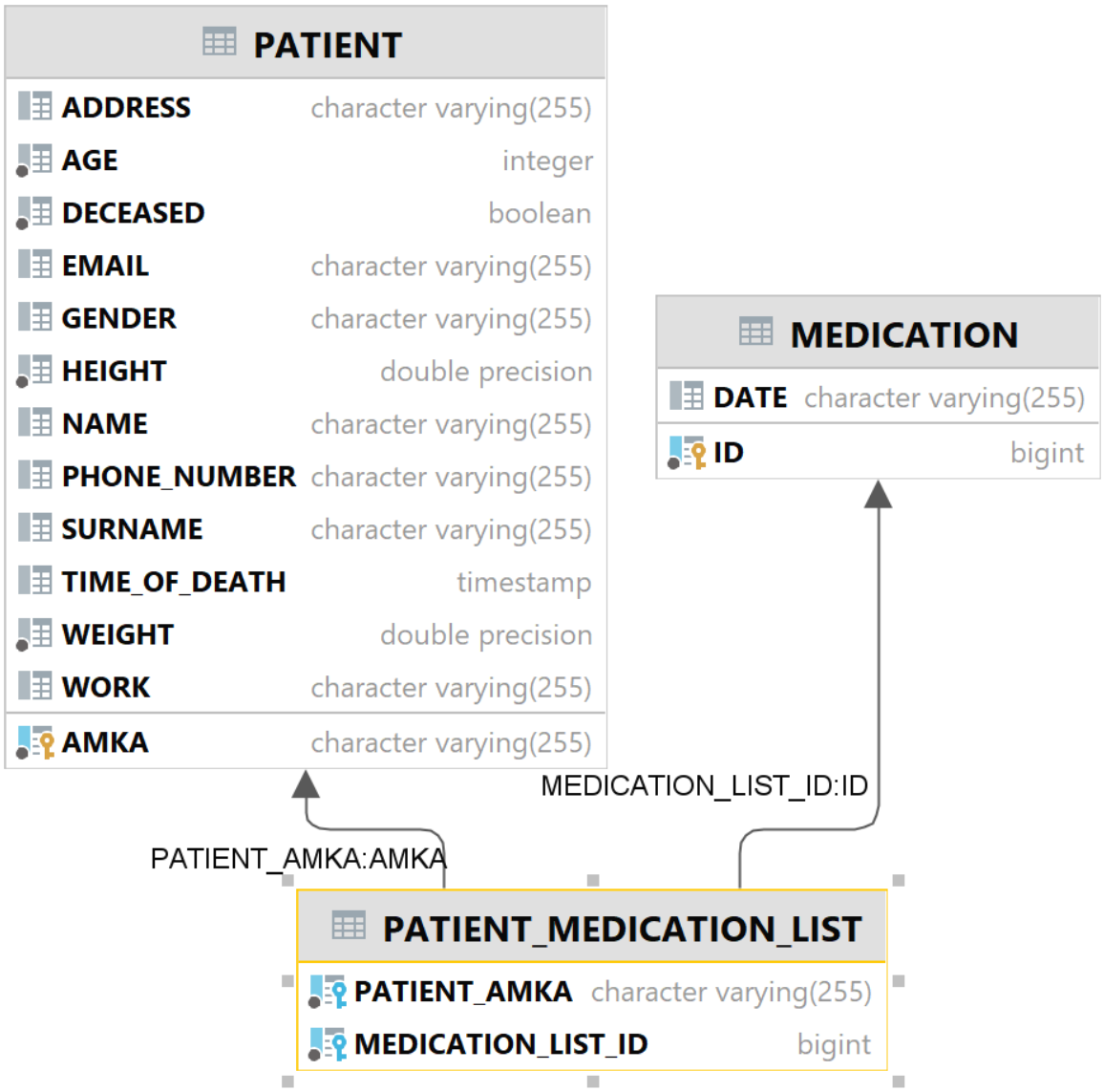

Figure 14: Σύνδεση του πίνακα Medication με τον πίνακα Patient

4.4.6 Πίνακας Statistics

Κατά την είσοδο ή την έξοδο ενός ασθενούς από την ιατρική μονάδα , καθώς και σε κάθε θάνατο , το σύστημα προσθέτει στον πίνακα Statistics με σκοπό την εξαγωγή στατιστικών συμπερασμάτων

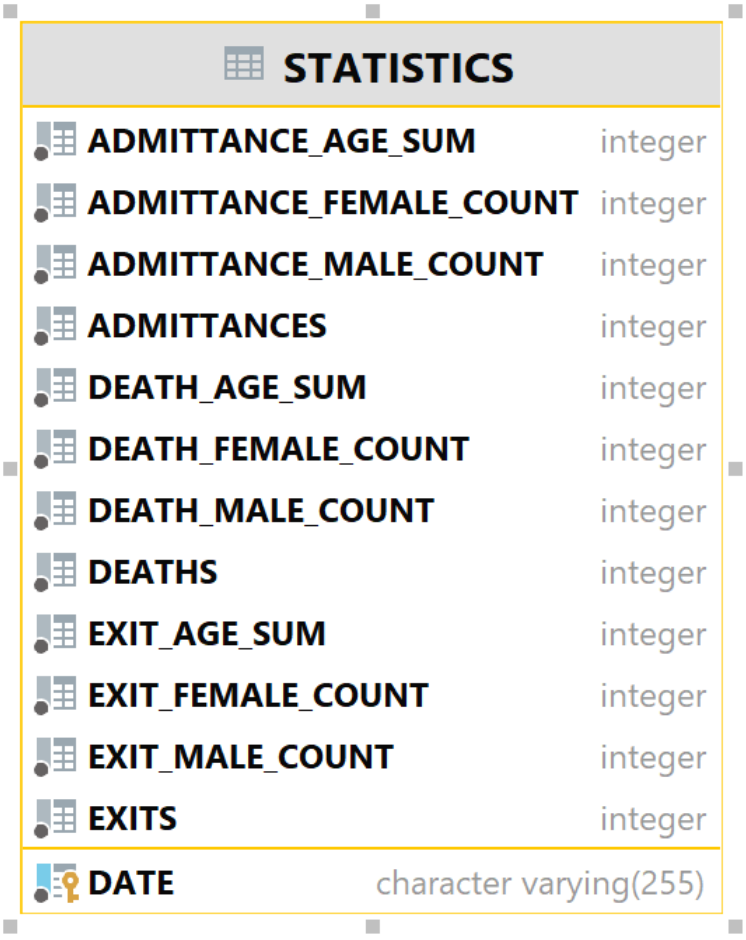

Figure 15: Πίνακας Statistics

4.4.7 Πίνακας AllTimeStatistics

Στον πίνακα AllTimeStatistics , βρίσκουμε την συνολική πληροφορία σχετικά με τα εισιτήρια και εξιτήρια ασθενών από την ιατρική μονάδα όπως και για τους θανάτους ασθενών

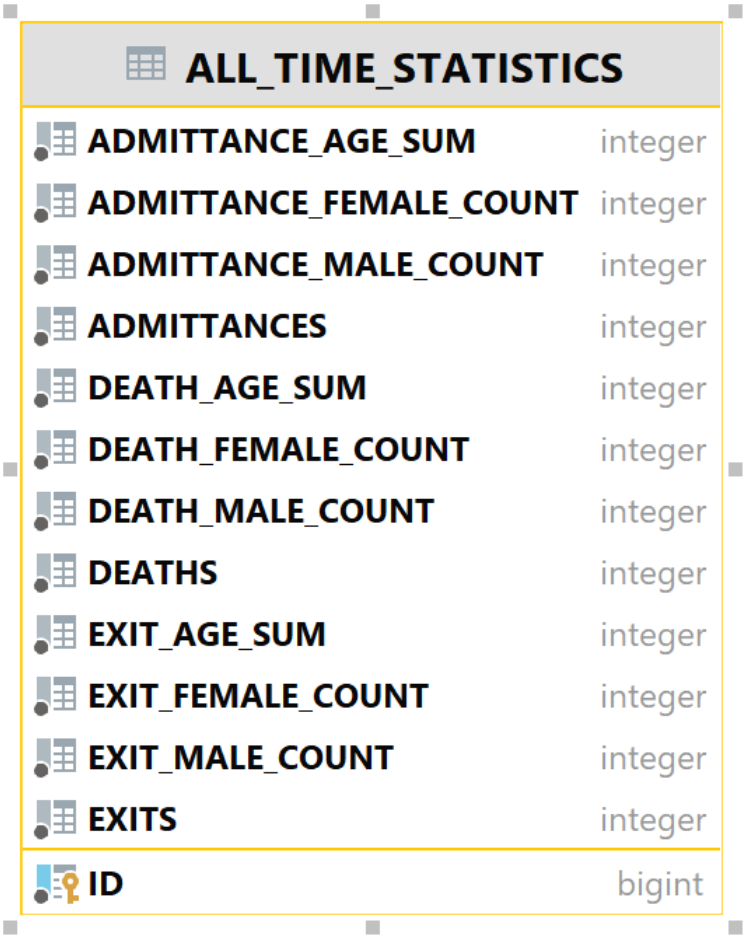

Figure 16: Πίνακας All_Time_Statistics

ΚΕΦΑΛΑΙΟ 5 Υλοποίηση τεχνικής λύσης

5.1 Δομή κώδικα

5.1.1 Δομή κώδικα Back End

Η υλοποίηση του back end έχει πραγματοποιηθεί με τη χρήση τεχνολογιών Java

Executable

- **IatrikosFakelosApplication.java:** Είναι η κλάση με την οποία τρέχει το project

  Objects

- **Patient.java:** Η κλάση για τα αντικείμενα των Patient. Το βασικό αντικείμενο της υλοποίησης μας , το οποίο έχει όλη την πληροφορία του κάθε ασθενή
- **BloodExamination.java:** Η κλάση για τα αντικείμενα των εξετάσεων αίματος. Η κάθε εξέταση αίματος περιέχει μια λίστα από ζεύγη στοιχείων-τιμών

- **BloodExaminationData.java:** Η κλάση για τα ζεύγη στοιχείων-τιμών στις εξετάσεις αίματος
- **HospitalAdmittance.java:** Η κλάση για τα αντικείμενα των εισιτηρίων στην ιατρική μονάδα
- **HospitalExit.java:** Η κλάση για τα αντικείμενα των εξιτηρίων από την ιατρική μονάδα
- **Medication.java:** Η κλάση για τα αντικείμενα της φαρμακευτικής αγωγής του ασθενούς. Η κάθε φαρμακευτική αγωγή περιέχει μια λίστα από ζεύγη φαρμακευτικής ουσίας-δόσης
- **Medicine.java:** Η κλάση για τα ζεύγη φαρμακευτικής ουσίας-δόσης
- **PatientHistory.java:** Η κλάση για τα αντικείμενα του ιστορικού του ασθενούς
- **Statistics.java:** Η κλάση που υλοποιεί τα στατιστικά ανά μέρα
- **AllTimeStatistics.java:** Η κλάση που υλοποιεί τα στατιστικά από την αρχή λειτουργίας του συστήματος

Configurations/Services/Controllers

- **CorsFilter.java:** Η κλάση που υλοποιεί ένα προσαρμοσμένο φίλτρο για το CORS(Cross-Origin Resource Sharing)
- **SecurityConfig.java:** Κλάση που ενεργοποιεί το security στα web services και περιέχει τις βασικές παραμέτρους του
- **RsaKeyProperties.java:** Ένα record στο οποίο ορίζονται οι βασικές παράμετροι για το Token
- **TokenService.java:** Η κλάση η οποία υλοποιεί το Token με το οποίο μας επιτρέπεται η επικοινωνία με τα endpoints της εφαρμογής
- **AuthController.java:** Ο controller που υλοποιεί το endpoint για το Token
- **AppController.java:** Ο Controller που υλοποιεί τα endpoints που αφορούν τον ασθενή και τα στατιστικά

Interfaces

Τα ακόλουθα Interfaces κάνουν extend το crud repository και είναι τα repositories των αντίστοιχων κλάσεων:

- PatientRepository.java
- BloodExaminationRepository.java
- BloodExaminationDataRepository.java
- HospitalAdmittanceRepository.java
- HospitalExitRepository.java
- MedicationRepository.java
- MedicineRepository.java
- PatientHistoryRepository.java
- StatisticsRepository.java
- AllTimeStatisticsRepository.java

5.1.1.1 Το αρχείο IatrikosFakelosApplication.java

Ο πηγαίος κώδικας, που παρατίθεται παρακάτω, αποτελεί το αρχείο IatrikosFakelosApplication.java με το οποίο ξεκινάει το πρόγραμμα μας

```
@SpringBootApplication
@EnableConfigurationProperties(RsaKeyProperties.class)
public class IatrikosFakelosApplication {

    public static void main(String[] args) {
        SpringApplication.run(IatrikosFakelosApplication.class, args);

    }
```

5.1.1.2 Το αρχείο Patient.java

Σε αυτό το αρχείο , ορίζονται τα βασικά χαρακτηριστικά του κύριου αντικειμένου πάνω στο οποίο βασίζεται το πρόγραμμα μας , αυτό του ασθενή

```
@Entity
public class Patient {
    @Id
    private String amka;
    private String name;
    private String surname;
    private String gender;
    private int age;
    private double weight;
    private double height;
    private String work;
    private String address;
    private String phoneNumber;
    private String email;
    private boolean deceased;
    private LocalDateTime timeOfDeath;

    @OneToMany
    private List<BloodExamination> bloodExaminationList;

    @OneToMany
    private List<HospitalAdmittance> hospitalAdmittanceList;
    @OneToMany
    private List<HospitalExit> hospitalExitList;
    @OneToMany
    private List<Medication> medicationList;
    @OneToMany
    private List<PatientHistory> patientHistoryList;
```

5.1.1.3 Το αρχείο BloodExamination.java:

Η κλάση για τα αντικείμενα των εξετάσεων αίματος. Η κάθε εξέταση αίματος περιέχει μια λίστα από ζεύγη στοιχείων-τιμών

```
@Entity
public class BloodExamination {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;
    private String date;
    @OneToMany
    private List<BloodExaminationData> bloodExaminationDataList;
```

5.1.1.4 Το αρχείο BloodExaminationData.java

Η κλάση για τα ζεύγη στοιχείων-τιμών στις εξετάσεις αίματος

```
@Entity
public class BloodExaminationData {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;
   private String typeOfExamination;
   private double valueOfExamination;
```

5.1.1.5 Το αρχείο HospitalAdmittance.java:

Η κλάση για τα αντικείμενα των εισιτηρίων στην ιατρική μονάδα

```
@Entity
public class HospitalAdmittance {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;
    private LocalDateTime date;
    private String reason;
```

5.1.1.6 Το αρχείο HospitalExit.java:

Η κλάση για τα αντικείμενα των εξιτηρίων από την ιατρική μονάδα

```
@Entity
public class HospitalExit {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;
    private LocalDateTime date;
    private String notes;
```

5.1.1.7 Το αρχείο Medication.java:

Η κλάση για τα αντικείμενα της φαρμακευτικής αγωγής του ασθενούς. Η κάθε φαρμακευτική αγωγή περιέχει μια λίστα από ζεύγη φαρμακευτικής ουσίας-δόσης.

```
@Entity
public class Medication {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;
    private String date;
    @OneToMany
    private List<Medicine> medicineList;
```

5.1.1.8 Το αρχείο Medicine.java:

Η κλάση για τα ζεύγη φαρμακευτικής ουσίας-δόσης

```
@Entity
public class Medicine {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;
    private String medicine;
    private String dose;
```

5.1.1.9 Το αρχείο PatientHistory.java:

Η κλάση για τα αντικείμενα του ιστορικού του ασθενούς

```
@Entity
public class PatientHistory {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;
    private String date;
    private String history;
```

5.1.1.10 Το αρχείο Statistics.java:

Η κλάση που υλοποιεί τα στατιστικά ανά μέρα

```
@Entity
public class Statistics {
    @Id
    private String date;

    private int admittances;

    private int exits;

    private int deaths;

    private int admittanceFemaleCount;

    private int admittanceMaleCount;

    private int admittanceAgeSum;

    private int exitFemaleCount;

    private int exitMaleCount;

    private int exitAgeSum;
```

```
    private int deathFemaleCount;

    private int deathMaleCount;

    private int deathAgeSum;

    public Statistics(String date, int admittances, int exits, int deaths, int admittanceFemaleCount, int
admittanceMaleCount, int admittanceAgeSum, int exitFemaleCount, int exitMaleCount, int exitAgeSum,
int deathFemaleCount, int deathMaleCount, int deathAgeSum) {
        this.date = date;
        this.admittances = admittances;
        this.exits = exits;
        this.deaths = deaths;
        this.admittanceFemaleCount = admittanceFemaleCount;
        this.admittanceMaleCount = admittanceMaleCount;
        this.admittanceAgeSum = admittanceAgeSum;
        this.exitFemaleCount = exitFemaleCount;
        this.exitMaleCount = exitMaleCount;
        this.exitAgeSum = exitAgeSum;
        this.deathFemaleCount = deathFemaleCount;
        this.deathMaleCount = deathMaleCount;
        this.deathAgeSum = deathAgeSum;
    }


            public Statistics() {
    }
```

5.1.1.11 Το αρχείο AllTimeStatistics.java:

Η κλάση που υλοποιεί τα στατιστικά από την αρχή λειτουργίας του συστήματος

```
@Entity
public class AllTimeStatistics {
    @Id
    private int id;
    private int admittances;

    private int exits;

    private int deaths;

    private int admittanceFemaleCount;

    private int admittanceMaleCount;

    private int admittanceAgeSum;

    private int exitFemaleCount;

    private int exitMaleCount;
```

```
    private int exitAgeSum;

    private int deathFemaleCount;

    private int deathMaleCount;

    private int deathAgeSum;

    public AllTimeStatistics(int id, int admittances, int exits, int deaths, int admittanceFemaleCount, int
admittanceMaleCount, int admittanceAgeSum, int exitFemaleCount, int exitMaleCount, int exitAgeSum,
int deathFemaleCount, int deathMaleCount, int deathAgeSum) {

        this.id=id;
        this.admittances = admittances;
        this.exits = exits;
        this.deaths = deaths;
        this.admittanceFemaleCount = admittanceFemaleCount;
        this.admittanceMaleCount = admittanceMaleCount;
        this.admittanceAgeSum = admittanceAgeSum;
        this.exitFemaleCount = exitFemaleCount;
        this.exitMaleCount = exitMaleCount;
        this.exitAgeSum = exitAgeSum;
        this.deathFemaleCount = deathFemaleCount;
        this.deathMaleCount = deathMaleCount;
        this.deathAgeSum = deathAgeSum;
    }


            public AllTimeStatistics() {
    }
```

5.1.1.12 Το αρχείο CorsFilter.java:

Η κλάση που υλοποιεί ένα προσαρμοσμένο φίλτρο για το CORS(Cross-Origin Resource Sharing)

```
public class CorsFilter implements Filter {

    @Override
    public void init(FilterConfig filterConfig) throws ServletException {

    }

    @Override
    public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse, FilterChain
filterChain) throws IOException, ServletException {
        HttpServletResponse response = (HttpServletResponse) servletResponse;
        HttpServletRequest request= (HttpServletRequest) servletRequest;

        response.setHeader("Access-Control-Allow-Origin", "*");
        response.setHeader("Access-Control-Allow-Methods", "GET,POST,DELETE,PUT,OPTIONS");
```

```
        response.setHeader("Access-Control-Allow-Headers", "*");
        response.setHeader("Access-Control-Allow-Credentials", "true");
        response.setHeader("Access-Control-Max-Age", "180");
        filterChain.doFilter(servletRequest, servletResponse);
    }
```

Με το από πάνω φίλτρο , επιτρέπουμε την επικοινωνία μεταξύ της java και του browser. Για λόγους ασφαλείας οι browsers περιορίζουν τα HTTP requests που έρχονται από προελεύσεις διαφορετικές από τη δική τους . Σε παραγωγικό περιβάλλον οι περιορισμοί είναι πιο αυστηροί και επιτρέπουν την επικοινωνία μόνο από συγκεκριμένες προελεύσεις.

5.1.1.13 Το αρχείο SecurityConfig.java:

Κλάση που ενεργοποιεί το security στα web services και περιέχει τις βασικές παραμέτρους του

```
@Configuration
@EnableWebSecurity
public class SecurityConfig {

    private final RsaKeyProperties rsaKeys;

    public SecurityConfig (RsaKeyProperties rsaKeys){
        this.rsaKeys=rsaKeys;
    }

    @Bean
    CorsFilter corsFilter() {
        CorsFilter filter = new CorsFilter();
        return filter;
    }
    @Bean
    public InMemoryUserDetailsManager user(){
        return new InMemoryUserDetailsManager(
                User.withUsername("Niki")
                        .password("{noop}password")
                        .authorities("read")
                        .build()
        );
    }

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        return http
                .addFilterBefore(corsFilter(), SessionManagementFilter.class)
                .csrf().disable()
                .authorizeHttpRequests(auth->auth
                        .antMatchers(HttpMethod.OPTIONS,"/**").permitAll()
                        .antMatchers("/token").permitAll()
                        .anyRequest().authenticated()
                )
```

```
            .oauth2ResourceServer(OAuth2ResourceServerConfigurer::jwt)
            .sessionManagement(session -
>session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
            .httpBasic(withDefaults())
            .build();
    }

    @Bean
    JwtDecoder jwtDecoder() {
        return NimbusJwtDecoder.withPublicKey(rsaKeys.publicKey()).build();
    }

    @Bean
    JwtEncoder jwtEncoder(){
        JWK jwk = new RSAKey.Builder(rsaKeys.publicKey()).privateKey(rsaKeys.privateKey()).build();
        JWKSource<SecurityContext> jwks = new ImmutableJWKSet<>(new JWKSet(jwk));
        return new NimbusJwtEncoder(jwks);
    }
```

Στο αρχείο του Security Configuration , ορίζουμε τους χρήστες οι οποίοι έχουν δικαίωμα εισόδου στο σύστημα , ποιες μέθοδοι χρειάζονται authenticated χρήστη καθώς και τα jwtEncoder kai jwtDecoder ώστε να γίνεται ο έλεγχος μεταξύ του jwt token που ήρθε στην κλήση με τα ανοιχτά session που έχουμε.

5.1.1.14 RsaKeyProperties.java:

Το αρχείο αυτό είναι ένα record στο οποίο ορίζονται οι βασικές παράμετροι για το Token

```
@ConfigurationProperties(prefix = "rsa")
public record RsaKeyProperties(RSAPublicKey publicKey, RSAPrivateKey privateKey) {

}
```

5.1.1.15 TokenService.java:

Η κλάση η οποία υλοποιεί το Token με το οποίο μας επιτρέπεται η επικοινωνία με τα endpoints της εφαρμογής. Δέχεται ένα Authentication και χτίζει ένα encoded JWT token το οποίο είναι αναγκαίο για να επιτύχουν οι υπόλοιπες HTTP κλήσεις.

```
@Service
public class TokenService {

    private final JwtEncoder encoder;

    public TokenService(JwtEncoder encoder) {
        this.encoder = encoder;
    }

    public String generateToken(Authentication authentication) {
        Instant now = Instant.now();
        String scope = authentication.getAuthorities().stream()
```

```
            .map(GrantedAuthority::getAuthority)
            .collect(Collectors.joining(""));
        JwtClaimsSet claims = JwtClaimsSet.builder()
            .issuer("self")
            .issuedAt(now)
            .expiresAt(now.plus(1, ChronoUnit.HOURS))
            .subject(authentication.getName())
            .claim("scope", scope)
            .build();
        return this.encoder.encode(from(claims)).getTokenValue();
    }

}
```

5.1.1.16 AuthController.java:

Ο controller που υλοποιεί το endpoint για το Token

```
@RestController
public class AuthController {
    private final TokenService tokenService;

    public AuthController(TokenService tokenService) {
        this.tokenService = tokenService;
    }

    @PostMapping("/token")
    public String token(Authentication authentication) {
        String token = tokenService.generateToken(authentication);
        return token;
    }
}
```

Όταν κληθεί η /token με Authorization = Basic Authentication , το πρόγραμμα ελέγχει αν υπάρχει αυτός ο συνδυασμός user και password και επιστρέφει ένα encoded jwt Token.

5.1.1.17 AppController.java:

Ο Controller που υλοποιεί όλα τα endpoints που αφορούν τον ασθενή και τα στατιστικά.

Έχοντας χρησιμοποιήσει το @Autowired annotation στις δηλώσεις όλων των repositories που χρειαζόμαστε , η σωστή εξάρτηση εκχωρείται από το Spring Container σε κάθε περίπτωση ξεχωριστά.

```
@RestController
public class AppController {


    @Autowired
    private PatientRepository patientRepository;
```

```
    @Autowired
    private BloodExaminationRepository bloodExaminationRepository;
    @Autowired
    private HospitalAdmittanceRepository hospitalAdmittanceRepository;

    @Autowired
    private HospitalExitRepository hospitalExitRepository;

    @Autowired
    private MedicationRepository medicationRepository;

    @Autowired
    private MedicineRepository medicineRepository;

    @Autowired
    private PatientHistoryRepository patientHistoryRepository;

    @Autowired
    private BloodExaminationDataRepository bloodExaminationDataRepository;

    @Autowired
    private StatisticsRepository statisticsRepository;

    @Autowired
    private AllTimeStatisticsRepository allTimeStatisticsRepository;


}
```

Παρακάτω παραθέτονται οι μέθοδοι που υλοποιούνται μέσα στην κλάση AppController :

5.1.1.17.1 addPatient:

```
@PostMapping("/patient/addPatient")
public String addPatient(@RequestBody Patient body){
    if(patientRepository.findPatientByAmka(body.getAmka())==null){
        patientRepository.save(body);
        return "Patient added";}
    else
        return "Patient already exists";
}
```

Όταν καλείται το endpoint /patient/addPatient τότε η μέθοδος δέχεται το RequestBody , ελέγχει αν υπάρχει ήδη εγγεγραμμένος ασθενής με αυτό το amka. Αν υπάρχει επιστρέφει το αντίστοιχο μήνυμα , αν όχι , τότε με τη χρήση του PatientRepository το αποθηκεύει στην βάση δεδομένων.

5.1.1.17.2 getPatientList:

```
@RequestMapping("/patients")
public Iterable<Patient> getPatientList(){
   return patientRepository.findAll();
}
```

Όταν καλείται το endpoint /patients τότε η μέθοδος με τη χρήση του PatientRepository βρίσκει και επιστρέφει όλους τους εγγεγραμμένους ασθενείς της βάσης.

5.1.1.17.3 getPatientByAmka:

```
@RequestMapping("/patient/{amka}")
public Patient getPatientByAmka(@PathVariable String amka){
   return patientRepository.findPatientByAmka(amka);
}
```

Όταν καλείται το endpoint /patient/{amka} , τότε με το PathVariable amka , γίνεται μια αναζήτηση στη βάση με τη χρήση του PatientRepository για να βρεθεί αν υπάρχει ασθενής με αυτό το amka . Αν ναι , επιστρέφεται αλλιώς το response έχει κενό body.

5.1.1.17.4 addHospitalAdmittance:

```
@PostMapping("/patient/{amka}/addHospitalAdmittance")
public Patient addHospitalAdmittance(@PathVariable String amka, @RequestBody HospitalAdmittance
body){
   Patient patient = patientRepository.findPatientByAmka(amka);
   hospitalAdmittanceRepository.save(body);
   patient.getHospitalAdmittanceList().add(body);
   patientRepository.save(patient);
   if (statisticsRepository.findStatisticsByDate(body.getDate().toLocalDate().toString())==null){
      Statistics statistics = new
Statistics(body.getDate().toLocalDate().toString(),1,0,0,(patient.getGender().equals("Female")?1:0),(patie
nt.getGender().equals("Female")?0:1), patient.getAge(), 0,0,0,0,0,0);
      statisticsRepository.save(statistics);
   }else{
      Statistics statistics =
statisticsRepository.findStatisticsByDate(body.getDate().toLocalDate().toString());
      statistics.setAdmittances(statistics.getAdmittances()+1);
      if(patient.getGender().equals("Female")){
         statistics.setAdmittanceFemaleCount(statistics.getAdmittanceFemaleCount()+1);
      }else{
         statistics.setAdmittanceMaleCount(statistics.getAdmittanceMaleCount()+1);
      }
      statistics.setAdmittanceAgeSum(statistics.getAdmittanceAgeSum()+patient.getAge());
      statisticsRepository.save(statistics);
   }
   addAllTimeStatistics(1,patient.getGender(),patient.getAge());
```

```
    return patient;
}
```

Όταν καλείται το endpoint /patient/{amka}/addHospitalAdmittance , βρίσκουμε τον ασθενή στον οποίο αντιστοιχεί αυτό το amka με τη βοήθεια του PatientRepository , σώζουμε στη βάση το εισιτήριο με τη βοήθεια του HospitalAdmittanceRepository , προσθέτουμε αυτό το εισιτήριο στην ασθενή και αποθηκεύουμε αυτή τη σύνδεση στη βάση δεδομένων.

Έπειτα , προσθέτουμε στα στατιστικά αυτό το εισιτήριο χρησιμοποιώντας το StatisticsRepository καθώς και την μέθοδο addAllTimeStatistics.

Επιστρέφουμε την ανανεωμένη πληροφορία για τον ασθενή.

5.1.1.17.5 addHospitalExit:

```
@PostMapping("/patient/{amka}/addHospitalExit")
public Patient addHospitalExit(@PathVariable String amka, @RequestBody HospitalExit body){
    Patient patient = patientRepository.findPatientByAmka(amka);
    hospitalExitRepository.save(body);
    patient.getHospitalExitList().add(body);
    patientRepository.save(patient);
    if (statisticsRepository.findStatisticsByDate(body.getDate().toLocalDate().toString())==null){
        Statistics statistics = new Statistics(body.getDate().toLocalDate().toString(),0,1,0,0,0,0,
(patient.getGender().equals("Female")?1:0),(patient.getGender().equals("Female")?0:1),
patient.getAge(),0,0,0);
        statisticsRepository.save(statistics);
    }else{
        Statistics statistics =
statisticsRepository.findStatisticsByDate(body.getDate().toLocalDate().toString());
        statistics.setExits(statistics.getExits()+1);
        if(patient.getGender().equals("Female")){
            statistics.setExitFemaleCount(statistics.getExitFemaleCount()+1);
        }else{
            statistics.setExitMaleCount(statistics.getExitMaleCount()+1);
        }
        statistics.setExitAgeSum(statistics.getExitAgeSum()+patient.getAge());
        statisticsRepository.save(statistics);
    }
    addAllTimeStatistics(2,patient.getGender(),patient.getAge());
    return patient;
}
```

Όταν καλείται το endpoint /patient/{amka}/addHospitalExit , βρίσκουμε τον ασθενή στον οποίο αντιστοιχεί αυτό το amka με τη βοήθεια του PatientRepository , σώζουμε στη βάση το εισιτήριο με τη βοήθεια του HospitalExitRepository , προσθέτουμε αυτό το εισιτήριο στην ασθενή και αποθηκεύουμε αυτή τη σύνδεση στη βάση δεδομένων.

Έπειτα , προσθέτουμε στα στατιστικά αυτό το εισιτήριο χρησιμοποιώντας το StatisticsRepository καθώς και την μέθοδο addAllTimeStatistics.

Επιστρέφουμε την ανανεωμένη πληροφορία για τον ασθενή.

5.1.1.17.6 addPatientHistory:

```
@PostMapping("/patient/{amka}/addPatientHistory")
public Patient addPatientHistory(@PathVariable String amka, @RequestBody PatientHistory body){
    Patient patient = patientRepository.findPatientByAmka(amka);
    patientHistoryRepository.save(body);
    patient.getPatientHistoryList().add(body);
    patientRepository.save(patient);
    return patient;
}
```

Με την κλήση του endpoint /patient/{amka}/addPatientHistory , βρίσκουμε τον ασθενή στον οποίο αντιστοιχεί το amka αυτό με την χρήση του PatientRepository. Σώζουμε στη βάση δεδομένων το νέο ιστορικό που μας ήρθε στο body της κλήσης, το προσθέτουμε στον ασθενή και τον αποθηκεύουμε στη βάση δεδομένων.

Τελικά η κλήση επιστρέφει την ανανεωμένη πληροφορία για τον ασθενή.

5.1.1.17.7 addMedication:

```
@PostMapping("/patient/{amka}/addMedication")
public Patient addMedication(@PathVariable String amka, @RequestBody Medication body){
    Patient patient = patientRepository.findPatientByAmka(amka);
    List<Medicine> medicineList = body.getMedicineList();
    medicineRepository.saveAll(medicineList);
    medicationRepository.save(body);
    patient.getMedicationList().add(body);
    patientRepository.save(patient);
    return patient;
}
```

Όταν κάποιος client , κάνει κλήση στο /patient/{amka}/addMedication , το πρόγραμμα κάνει extract από το body τη λίστα με τα Medicine και την αποθηκέυει στη βάση δεδομένων με τη βοήθεια των MedicineRepository και MedicationRepository. Επίσης μέσω του PatientRepository , έχουμε ανακτήσει τη πληροφορία του ασθενούς στον οποίο ανήκει το amka που στάλθηκε μέσω του PathVariable amka . Προσθέτουμε την φαρμακευτική αγωγή στον ασθενή μας και μετά τον σώζουμε στη βάση δεδομένων.

Επιστρέφουμε την πληροφορία του ασθενή μετά τις τελευταίες τροποποιήσεις.

5.1.1.17.8 addBloodExam:

```
@PostMapping("/patient/{amka}/addBloodExam")
public Patient addBloodExam(@PathVariable String amka, @RequestBody BloodExamination body){
    Patient patient = patientRepository.findPatientByAmka(amka);
    List<BloodExaminationData> bloodExamsDataList = body.getBloodExaminationDataList();
    bloodExaminationDataRepository.saveAll(bloodExamsDataList);
    bloodExaminationRepository.save(body);
    patient.getBloodExamList().add(body);
    patientRepository.save(patient);
    return patient;
}
```

Το endpoint /patient/{amka}/addBloodExam , καλείται όταν θέλουμε να προσθέσουμε σε έναν ασθενή νέες εξετάσεις αίματος . Το πρόγραμμα μας αναζητεί τον ασθενή στον οποίο ανήκει το amka που υπάρχει διαθέσιμο σαν PathVariable στην κλήση . Έπειτα αποθηκεύει στη βάση δεδομένων τη λίστα από δεδομένα εξετάσεων αίματος (ζεύγη στοιχείων και αντίστοιχων τιμών στο αίμα) καθώς και κάθε ζεύγος ξεχωριστά . Το παραπάνω γίνεται με τη βοήθεια των BloodExaminationRepository και BloodExaminationDataRepository αντίστοιχα . Στο τέλος αποθηκεύουμε τις αλλαγές αυτές στον συγκεκριμένο ασθενή.

Η επιστροφή της κλήσης είναι η πληροφορία του ασθενούς για τον οποίο έγινε αυτή η κλήση

5.1.1.17.9 updatePatient:

```
@PutMapping("/patient/{amka}/updatePatient")
public Patient updatePatient(@PathVariable String amka , @RequestBody Patient body){
    Patient patient = patientRepository.findPatientByAmka(amka);
    patient.setName(body.getName());
    patient.setSurname(body.getSurname());
    patient.setGender(body.getGender());
    patient.setAge(body.getAge());
    patient.setWeight(body.getWeight());
    patient.setHeight(body.getHeight());
    patient.setWork(body.getWork());
    patient.setAddress(body.getAddress());
    patient.setPhoneNumber(body.getPhoneNumber());
    patient.setEmail(body.getEmail());
    patientRepository.save(patient);
    return patient;
}
```

Για να ενημερώσουμε τις πληροφορίες ενός ασθενή , πέρα από το αν ζει και την ώρα θανάτου , πρέπει να καλεστεί η /patient/{amka}/updatePatient. Το πρόγραμμα μας βρίσκει τον ασθενή του οποίου τα δεδομένα θέλουμε να ενημερώσουμε μέσω του PatientRepository και μετά ανανεώνει την τιμή στο κάθε πεδίο με την αντίστοιχη που έχει σταλεί στο body της κλήσης.

Έπειτα αποθηκεύονται οι αλλαγές στη βάση δεδομένων και επιστρέφεται ο ασθενής.

5.1.1.17.10 setPatientDeath:

```
@PostMapping("/patient/{amka}/setDeath")
public Patient setPatientDeath(@PathVariable String amka , @RequestBody Patient body){
    Patient patient = patientRepository.findPatientByAmka(amka);
    patient.setDeceased(true);
    patient.setTimeOfDeath(body.getTimeOfDeath());
    patientRepository.save(patient);
    if (statisticsRepository.findStatisticsByDate(body.getTimeOfDeath().toLocalDate().toString())==null){
        Statistics statistics = new Statistics(body.getTimeOfDeath().toLocalDate().toString(),0,0,1,0,0,0, 0,0,
0,(patient.getGender().equals("Female")?1:0),(patient.getGender().equals("Female")?0:1),
patient.getAge());
        statisticsRepository.save(statistics);
```

```
    }else{
        Statistics statistics =
statisticsRepository.findStatisticsByDate(body.getTimeOfDeath().toLocalDate().toString());
        statistics.setDeaths(statistics.getDeaths()+1);
        if(patient.getGender().equals("Female")){
            statistics.setDeathFemaleCount(statistics.getDeathFemaleCount()+1);
        }else{
            statistics.setDeathMaleCount(statistics.getDeathMaleCount()+1);
        }
        statistics.setDeathAgeSum(statistics.getDeathAgeSum()+patient.getAge());
        statisticsRepository.save(statistics);
    }
    addAllTimeStatistics(3,patient.getGender(),patient.getAge());
    return patient;
}
```

Για να δηλωθεί στο σύστημα ο θάνατος ενός ασθενή , χρειάζεται μια ξεχωριστή κλήση. Όταν γίνει κλήση στο endpoint /patient/{amka}/setDeath , αναζητούμε τον ασθενή στον οποίο αντιστοιχεί το δοσμένο στο PathVariable amka αμκα και αλλάζουμε την Boolean τιμή setDeceased σε true. Επίσης ενημερώνουμε την ώρα θανάτου με την ώρα που ήρθε στο body της κλήσης.

Ακολούθως , προσθέτουμε στα στατιστικά αυτόν τον θάνατο, χρησιμοποιώντας το StatisticsRepository καθώς και την μέθοδο addAllTimeStatistics.

Επιστρέφουμε την ανανεωμένη πληροφορία για τον ασθενή.

5.1.1.17.11 getRangeStatistics:

```
@RequestMapping("/statistics/range/{range}")
public Statistics getRangeStatistics(@PathVariable String range){
    int integerRange= Integer.parseInt(range);
    Statistics returnStatistics = new Statistics();
    Statistics tempStatistics = new Statistics();
    for (int i=0;i<integerRange;i++){
        tempStatistics =
statisticsRepository.findStatisticsByDate(LocalDateTime.now().minusDays(i).toLocalDate().toString());
        if(tempStatistics!=null){

returnStatistics.setAdmittances(returnStatistics.getAdmittances()+tempStatistics.getAdmittances());
            returnStatistics.setExits(returnStatistics.getExits()+tempStatistics.getExits());
            returnStatistics.setDeaths(returnStatistics.getDeaths()+tempStatistics.getDeaths());

returnStatistics.setAdmittanceFemaleCount(returnStatistics.getAdmittanceFemaleCount()+tempStatistics.
getAdmittanceFemaleCount());

returnStatistics.setAdmittanceMaleCount(returnStatistics.getAdmittanceMaleCount()+tempStatistics.getA
dmittanceMaleCount());

returnStatistics.setExitFemaleCount(returnStatistics.getExitFemaleCount()+tempStatistics.getExitFemale
Count());
```

```
returnStatistics.setExitMaleCount(returnStatistics.getExitMaleCount()+tempStatistics.getExitMaleCount());

returnStatistics.setDeathFemaleCount(returnStatistics.getDeathFemaleCount()+tempStatistics.getDeathF
emaleCount());

returnStatistics.setDeathMaleCount(returnStatistics.getDeathMaleCount()+tempStatistics.getDeathMaleC
ount());
            returnStatistics.setAdmittanceAgeSum(returnStatistics.getAdmittanceAgeSum()+
tempStatistics.getAdmittanceAgeSum());
            returnStatistics.setDeathAgeSum(returnStatistics.getDeathAgeSum()+
tempStatistics.getDeathAgeSum());
            returnStatistics.setExitAgeSum(returnStatistics.getExitAgeSum()+
tempStatistics.getExitAgeSum());
        }
    }
    return returnStatistics;
}
```

Για να τραβήξουμε τα στατιστικά των τελευταίων ημερών , χρησιμοποιούμε κλήσεις στο endpoint /statistics/range/{range}. Δημιουργούμε ένα αντικείμενο returnStatistics της κλάσης Statistics . Ξεκινώντας από τη σημερινή ημέρα και πηγαίνοντας προς τα πίσω τόσες ημέρες όσες μας υποδεικνύει το PathVariable range , προσθέτουμε κάθε φορά τα στατιστικά τις κάθε ημέρας στις αντίστοιχες τιμές του αντικειμένου returnStatistics .

Τέλος , επιστρέφουμε αυτό το αντικείμενο της Statistics , το returnStatistics.

5.1.1.17.12 addAllTimeStatistics:

```
private void addAllTimeStatistics(int type , String gender , int age){
    AllTimeStatistics allTimeStatistics = allTimeStatisticsRepository.findAllTimeStatisticsById(1);
    switch (type) {
        case 1 -> {
            allTimeStatistics.setAdmittances(allTimeStatistics.getAdmittances() + 1);
            allTimeStatistics.setAdmittanceAgeSum(allTimeStatistics.getAdmittanceAgeSum() + age);
            if (gender.equals("Female")) {
                allTimeStatistics.setAdmittanceFemaleCount(allTimeStatistics.getAdmittanceFemaleCount() +
1);
            } else {
                allTimeStatistics.setAdmittanceMaleCount(allTimeStatistics.getAdmittanceMaleCount() + 1);
            }
        }
        case 2 -> {
            allTimeStatistics.setExits(allTimeStatistics.getExits() + 1);
            allTimeStatistics.setExitAgeSum(allTimeStatistics.getExitAgeSum() + age);
            if (gender.equals("Female")) {
                allTimeStatistics.setExitFemaleCount(allTimeStatistics.getExitFemaleCount() + 1);
            } else {
                allTimeStatistics.setExitMaleCount(allTimeStatistics.getExitMaleCount() + 1);
            }
        }
```

```
            case 3 -> {
                allTimeStatistics.setDeaths(allTimeStatistics.getDeaths() + 1);
                allTimeStatistics.setDeathAgeSum(allTimeStatistics.getDeathAgeSum() + age);
                if (gender.equals("Female")) {
                    allTimeStatistics.setDeathFemaleCount(allTimeStatistics.getDeathFemaleCount() + 1);
                } else {
                    allTimeStatistics.setDeathMaleCount(allTimeStatistics.getDeathMaleCount() + 1);
                }
            }
        }
        allTimeStatisticsRepository.save(allTimeStatistics);
    }
```

Η addAllTimeStatistics, είναι μια private μέθοδος που χρησιμοποιείται από τις setHospitalAdmittance , setHospitalExit , setPatientDeath ώστε να προστεθούν στα στατιστικά από την αρχή της υλοποίησης οι αντίστοιχες τιμές . Η εκάστοτε μέθοδος που καλεί την addAllTimeStatistics , τροφοδοτεί την κλήση αυτή με πληροφορίες για το φύλο και την ηλικία του ασθενούς αλλά και για το αν πρόκειτε για εισιτήριο στο νοσοκομείο , εξιτήριο από το νοσοκομείο ή για θάνατο ασθενούς.

Με τη χρήση του AllTimeStatisticsRepository , ανακτούμε την μέχρι τωρα αποθηκευμένη στη βάση δεδομένων πληροφορία και την ανανεώνουμε βάση των παραμέτρων που μας ήρθα στην κλήση.

Η μέθοδος addAllTimeStatistics δεν επιστρέφει κάτι (void).

5.1.1.17.13 getAllStatistics:

```
@RequestMapping("/statistics/all")
public AllTimeStatistics getAllStatistics(){
    return allTimeStatisticsRepository.findAllTimeStatisticsById(1);
}
```

Για να ανακτηθεί η πληροφορία για τα στατιστικά από την αρχή της υλοποίησης , γίνεται μια κλήση στο endpoint /statistics/all. Αυτή η κλήση δεν χρειάζεται κάποιο PathVariable ή κάτι από το body της κλήσης.

Επιστρέφεται η πληροφορία για τα allTimeStatistics με τη βοήθεια του allTimeStatisticsRepository.

5.1.2 Δομή κώδικα GUI

Η υλοποίηση του GUI έχει γίνει με τη χρήση τεχνολογιών JavaScript και HTML5. Τα αρχεία που παρουσιάζονται σε αυτή την ενότητα , είναι σε ζευγάρια , ένα σε HTML και ένα σε JavaScript

- Login Page: Η αρχική σελίδα της εφαρμογής . Ο χρήστης χρειάζεται προσωπικό όνομα χρήστη και κωδικό για να εισέλθει στο σύστημα

- Main Page: Η σελίδα στην οποία μεταφέρεται ο χρήστης μόλις πιστοποιηθεί από το σύστημα. Σε αυτή τη σελίδα ο χρήστης μπορεί να αναζητήσει ασθενείς ή να δει στατιστικά στοιχεία.

- Add New Patient: Η σελίδα μέσω της οποία ο χρήστης μπορεί να εισάγει έναν νέο ασθενή στο σύστημα

- Patient Medical Info: Όλες οι πληροφορίες του ασθενούς εμφανίζονται στη σελίδα αυτή

- Update Patient: Η σελίδα στην οποία γίνονται αλλαγές στις βασικές πληροφορίες του ασθενούς
- Declare Death: Σελίδα στην οποία ο χρήστης καταχωρεί τον θάνατο του ασθενή
- Add Hospital Admittance: Τα εισιτήρια στην ιατρική μονάδα για τους ασθενείς καταχωρούνται σε αυτή τη σελίδα
- Add Hospital Exit: Ο χρήστης σε αυτή τη σελίδα καταχωρεί τα εξιτήρια ασθενών
- Add Medication: Σε αυτή τη σελίδα γίνεται η καταχώρηση φαρμακευτικής αγωγή ασθενή
- Add Patient History: Σελίδα στην οποία ο χρήστης μπορεί να προσθέσει το ιστορικό του ασθενή
- Add Blood Exams: Ο χρήστης καταχωρεί τις εξετάσεις αίματος του ασθενή

ΚΕΦΑΛΑΙΟ 6 Λειτουργικότητα

6.1 Οθόνη εισόδου χρήστη (MainPage.html)

Για την είσοδο του χρήστη στην εφαρμογή , χρειάζονται διαπιστευτήρια τα οποία τα εισάγει σε μια φόρμα.

Figure 17: Οθόνη εισόδου χρήστη

Τα στοιχεία που εισάγει ο χρήστης , με το πάτημα του κουμπιού «Login» , με τη χρήση της javaScript σε base64 encoded :

```
encodedData = btoa(`${username.value}:${password.value}`);
```

Και αποστέλλεται σαν Basic Authentication στο endpoint /token:

```
fetch('http://localhost:8080/token',{method: 'POST',headers: {'Accept':'*/*','Content-Type':
'application/json','Authorization': `Basic ${encodedData}`}})
```

Σε περίπτωση που ο χρήστης δεν έδωσε σωστά στοιχεία , τότε του εμφανίζεται ένα μήνυμα στον browser του :

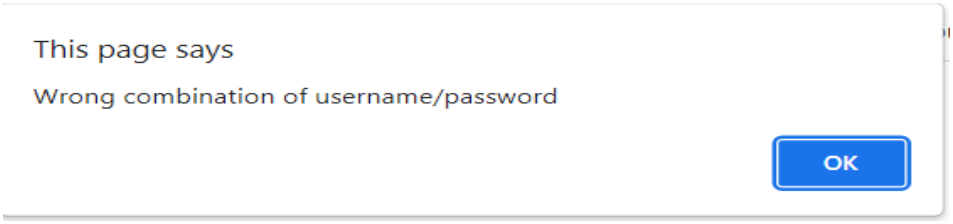

Figure 18: Alert εσφαλμένων στοιχείων

Όταν ο χρήστης δίνει σωστά στοιχεία , μεταφέρεται στην κεντρική σελίδα της εφαρμογής (**HomePage.html**)

6.2 HomePage.html

Στην κεντρική σελίδα της εφαρμογής, ο χρήστης μπορεί να αναζητήσει έναν ασθενή με βάση το Άμκα του , να δημιουργήσει έναν νέο φάκελο ασθενούς στο σύστημα είτε να αναζητήσει στατιστικά στοιχεία.

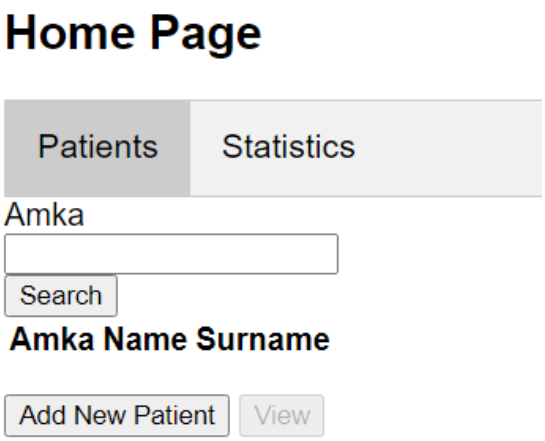

Figure 19: Κεντρική σελίδα , καρτέλα ασθενών

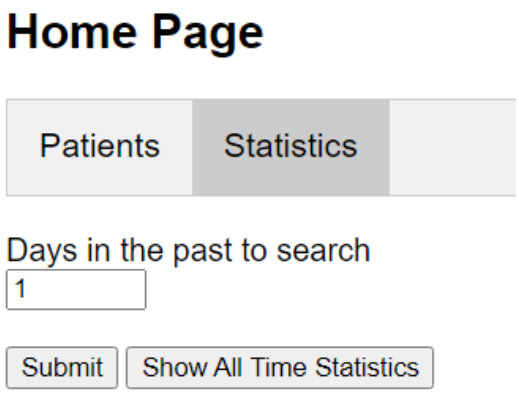

Figure 20: Κεντρική σελίδα , καρτέλα στατιστικών

6.2.1 Αναζήτηση ασθενούς βάση Άμκα

Ο Χρήστης εισάγει το Άμκα στο αντίστοιχο πεδίο και πατάει το κουμπί «search» .
Το event αυτό , κάνει κλήση στο endpoint /patient/{amka}

```
fetch(`http://localhost:8080/patient/${amka}`,{method: 'GET',headers: {'Accept':'*/*','Content-Type': 'application/json','Authorization': `Bearer ${bearerToken}`}})
```

Αν το Άμκα που εισήγαγε ο χρήστης δεν αντιστοιχεί σε κάποιον ασθενή , ο χρήστης ενημερώνετε με ένα σχετικό μήνυμα:

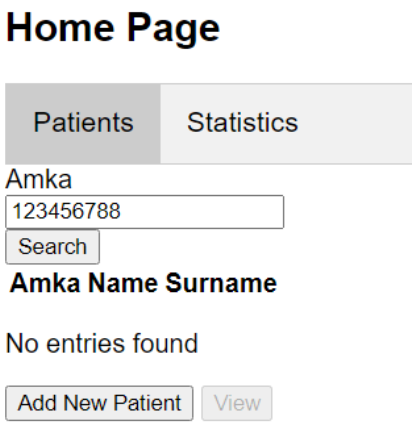

Figure 21: Αναζήτηση ασθενούς χωρίς επιτυχία

Αν το Άμκα αντιστοιχεί σε κάποιον ασθενή , εγγεγραμμένο στο σύστημα , τότε εμφανίζονται κάποιες βασικές πληροφορίες για αυτόν και ενεργοποιείται το κουμπί “view”

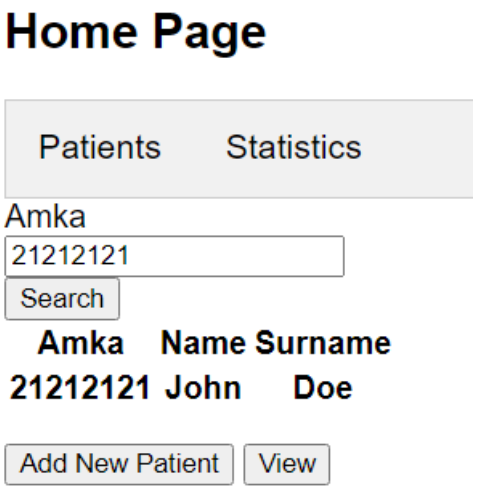

Figure 22: Επιτυχής αναζήτηση ασθενούς

Για να δούμε περισσότερα στοιχεία για τον ασθενή , χρειάζεται να πατήσουμε το κουμπί «View» , το οποίο θα μας οδηγήσει σε μια νέα σελίδα (**PatientMedicalInfo.html**)

6.2.2 Εγγραφή νέου ασθενούς στο σύστημα

Για να κάνουμε εγγραφή νέου ασθενούς στο σύστημα , ο χρήστης πρέπει να πατήσει το κουμπί «Add New Patient». Το σύστημα θα μας μεταφέρει στη σελίδα εγγραφής νέου ασθενούς (**AddNewPatient.html**)

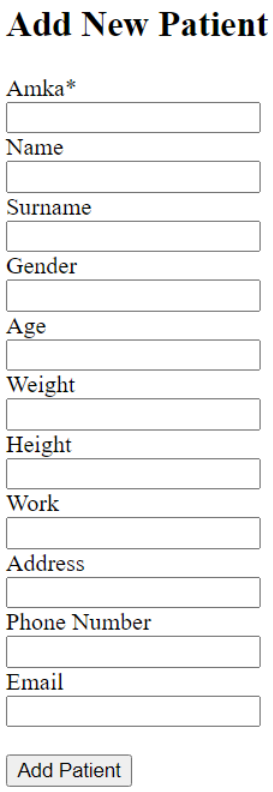

Figure 23: Φόρμα πρόσθεσης νέου ασθενούς στο σύστημα

Στη σελίδα εγγραφής νέου ασθενούς , ο χρήστης καλείται να συμπληρώσει όση περισσότερη πληροφορία μπορεί για τον ασθενή , με το Άμκα να είναι υποχρεωτικό πεδίο.

Με το πάτημα του κουμπιού «Add Patient» , το σύστημα κάνει μια κλήση στο /patient/addPatient :

```
fetch(`http://localhost:8080/patient/addPatient`,{method: 'POST',headers: {'Content-Type': 'application/json','Authorization': `Bearer ${bearerToken}`},body: data})
```

Στέλνοντας στο body , ότι πληροφορία παρείχε ο χρήστης στην φόρμα. Το πρόγραμμα μας στέλνει στην σελίδα με τα στοιχεία του ασθενούς **(PatientMedicalInfo.html**)

6.2.3 Αναζήτηση στατιστικών για συγκεκριμένο αριθμό ημερών

Ο χρήστης έχει τη δυνατότητα να ζητήσει από το σύστημα τα στατιστικά των τελευταίων ημερών. Τροφοδοτεί το σύστημα με τον αριθμό των ημερών που επιθυμεί και πατάει το κουμπί «Submit». Το πρόγραμμα κάνει κλήση στο endpoint /statistics/range/{range}

```
range = rangeOfDays.value
fetch(`http://localhost:8080/statistics/range/${range}`,{method: 'GET',headers: {'Accept':'*/*','Content-Type': 'application/json','Authorization': `Bearer ${bearerToken}`}})
```

Figure 24: Αναζήτηση στατιστικών στοιχείων για τις τελευταίες ημέρες

6.2.4 Αναζήτηση συνολικών στατιστικών στοιχείων ιατρικής μονάδας

Για την αναζήτηση των στατιστικών στοιχείων της ιατρικής μονάδας , ο χρήστης πατάει το κουμπί «Show All Time Statistics». Γίνεται τότε μια κλήση στο endpoint /statistics/all

```
fetch(`http://localhost:8080/statistics/all`,{method: 'GET',headers: {'Accept':'*/*','Content-Type':
'application/json','Authorization': `Bearer ${bearerToken}`}})
```

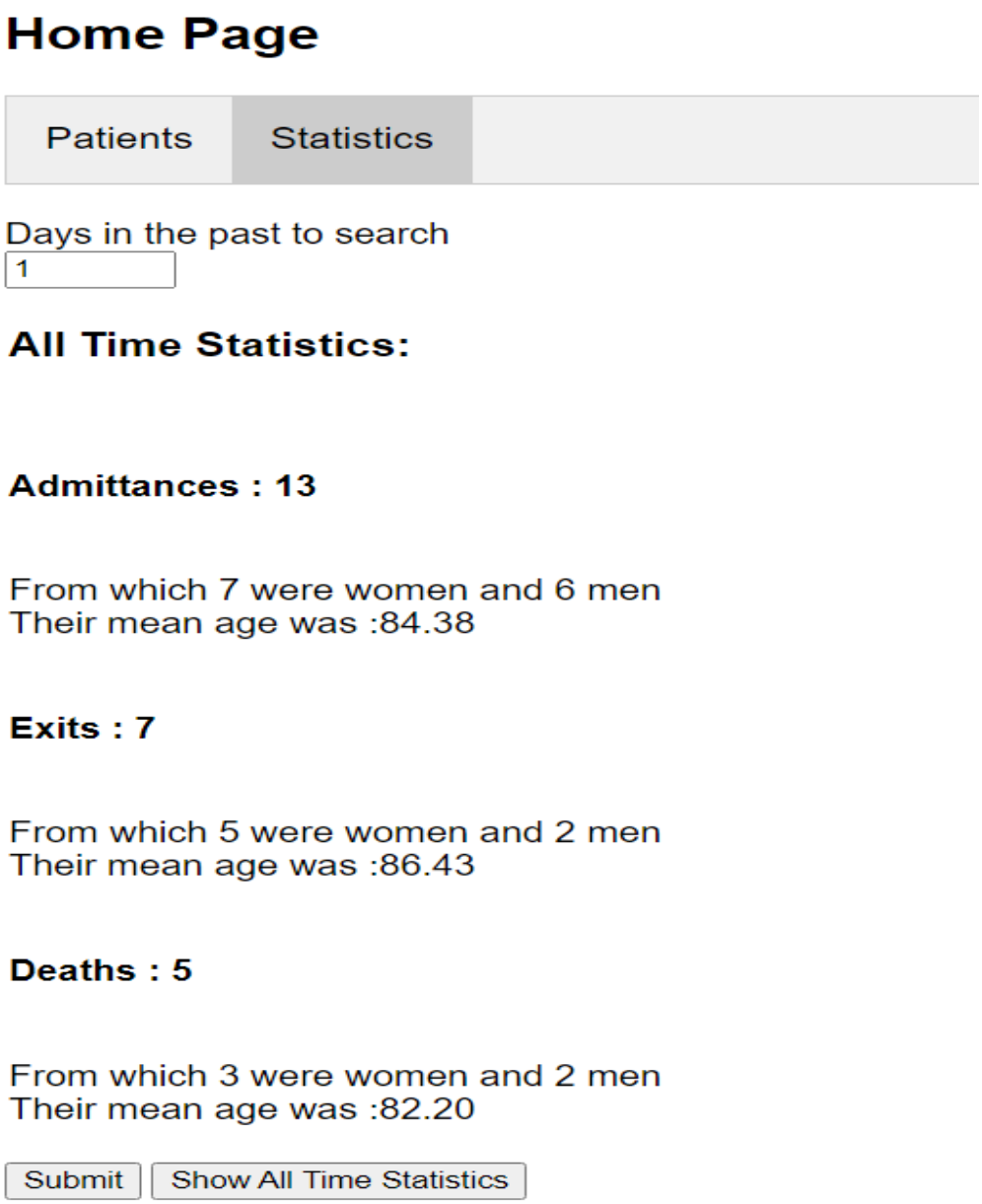

Figure 25: Αναζήτηση στατιστικών ιατρικής μονάδας

6.3 PatientMedicalInfo.html

Στη σελίδα PatientMedicalInfo.html ο χρήστης μπορεί να βρει όλη την πληροφορία σχετικά με τον ασθενή. Όταν ο χρήστης εισέρχεται στη σελίδα με τις πληροφορίες για τον ασθενή , καλείται το endpoint /patient/{amka}.

```
fetch(`http://localhost:8080/patient/${amka}`,{method: 'GET',headers: {'Accept':'*/*','Content-Type': 'application/json','Authorization': `Bearer ${bearerToken}`}})
```

Η σελίδα έχει 6 καρτέλες στις οποίες παρουσιάζει τις διάφορες πληροφορίες

6.3.1 Βασικές πληροφορίες ασθενούς

Στην καρτέλα των βασικών πληροφοριών , εμφανίζονται πληροφορίες όπως το Όνομα , ηλικία , email καθώς και αν ο ασθενής είναι εν ζωή.

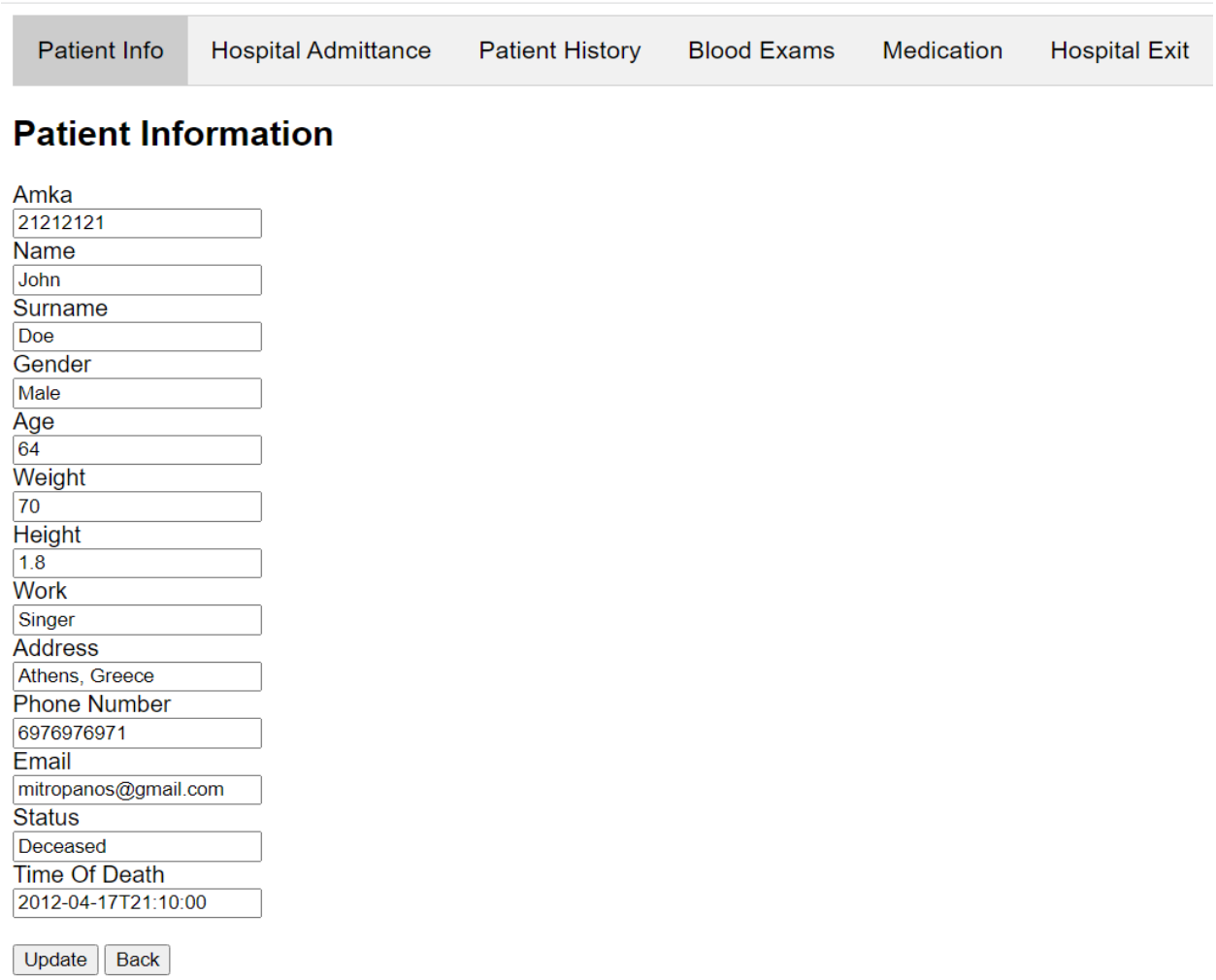

Figure 26: Φάκελος ασθενούς, Γενικές Πληροφορίες

Πατώντας το κουμπί «Update» οδηγούμαστε στην σελίδα **UpdatePatient.html .** Το κουμπί «Back» μας επιστρέφει στην **HomePage.html**

6.3.2 Εισιτήρια στην ιατρική μονάδα

Πατώντας στην καρτέλα «Hospital Admittance» , εμφανίζεται η πληροφορία σχετικά με τα εισιτήρια του ασθενή στην Ιατρική μονάδα

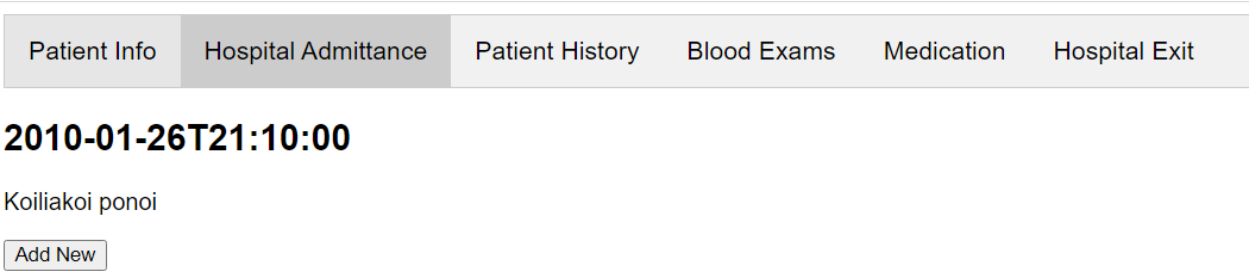

Figure 27: Φάκελος ασθενούς: Εισιτήρια στην ιατρική μονάδα

Το κουμπί «Add New» μας κατευθύνει στη σελίδα **AddHospitalAdmittance.html**

6.3.3 Ιστορικό ασθενούς

Η καρτέλα «Patient History» εμφανίζει το ιστορικό του ασθενούς που έχει καταχωρηθεί στην βάση δεδομένων

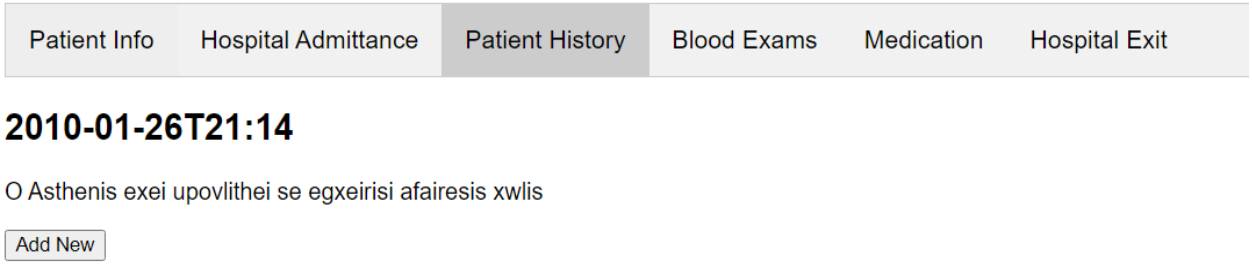

Figure 28: Φάκελος ασθενούς, Ιστορικό ασθενούς

Με την επιλογή «Add New» οδηγούμαστε στη σελίδα **AddPatientHistory.html**

6.3.4 Εξετάσεις Αίματος

Οι εξετάσεις αίματος για τον κάθε ασθενή , μπορούν να προβληθούν από την καρτέλα «Blood Exams»

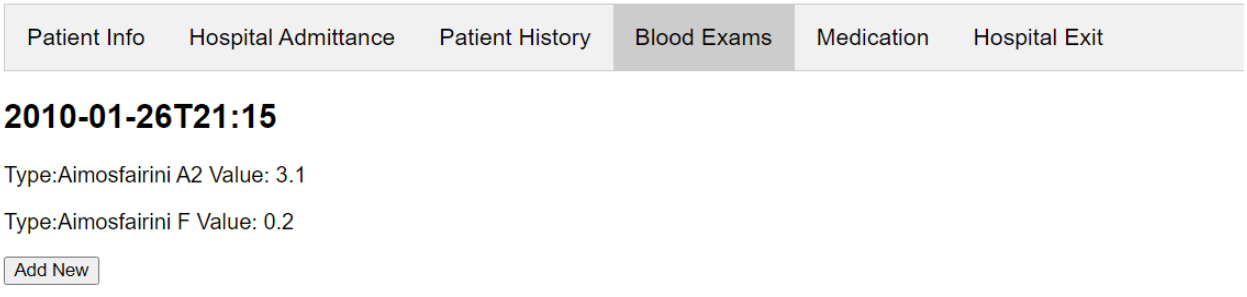

Figure 29: Φάκελος ασθενούς, Εξετάσεις αίματος

Για να προστεθούν νέες εξετάσεις αίματος , ο χρήστης πρέπει να πατήσει το κουμπί «Add New» για να μεταφερθεί στην σελίδας **AddBloodExams.html**

6.3.5 Φαρμακευτική αγωγή

Η φαρμακευτική αγωγή του ασθενούς είναι διαθέσιμη στην καρτέλα Medication

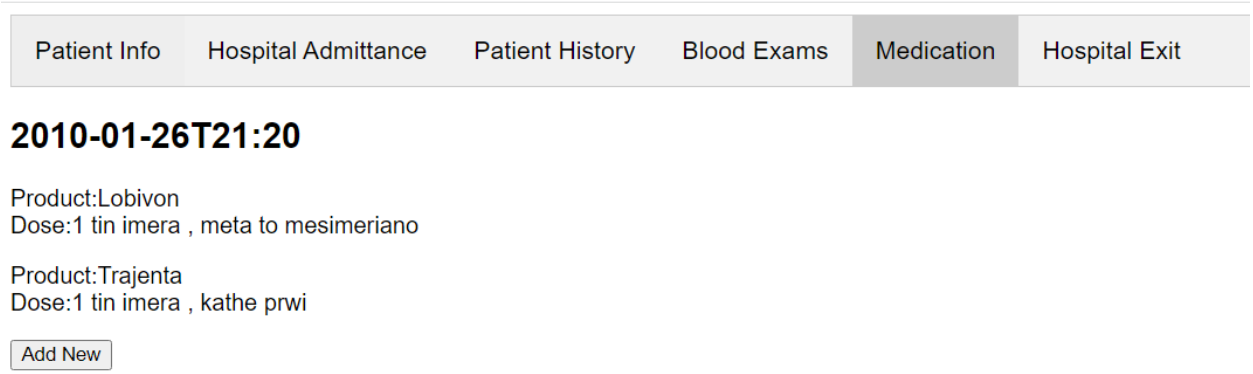

Figure 30: Φάκελος ασθενούς, Φαρμακευτική αγωγή

Το κουμπί «Add New» μας οδηγεί στην σελίδα **AddMedication.html**

6.3.6 Εξιτήρια από την ιατρική μονάδα

Ο Χρήστης μπορεί να βρει πληροφορίες για τα εξιτήρια του ασθενούς πατώντας στην καρτέλα «Hospital Exit»

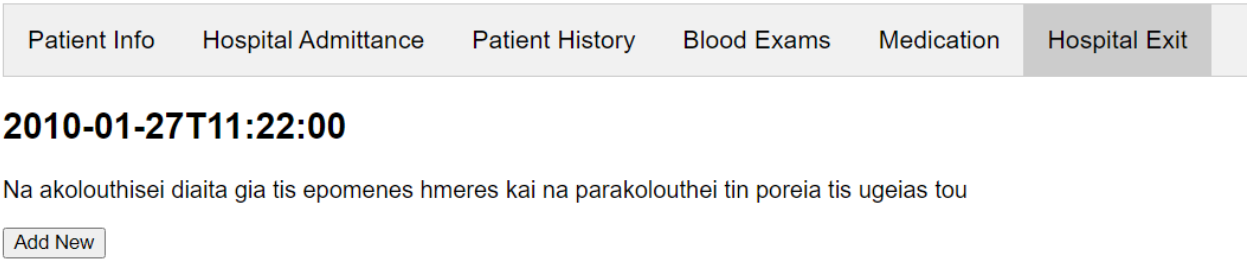

Figure 31: Φάκελος ασθενούς, Εξιτήρια από την ιατρική μονάδα

Νέα εξιτήρια μπορούν να προστεθούν πατώντας το κουμπί «Add New» που μας πηγαίνει στη σελίδα **AddHospitalExit.html**

6.4 UpdatePatient.html

Η σελίδα από την οποία ενημερώνονται οι βασικές πληροφορίες του ασθενούς. Ο χρήστης προσθέτει ή αλλάζει τις πληροφορίες που υπάρχουν στα διαθέσιμα πεδία.

Τα πεδία «Status» και «Time Of Death» τροποποιούνται από την σελίδα **DeclareDeath.html**

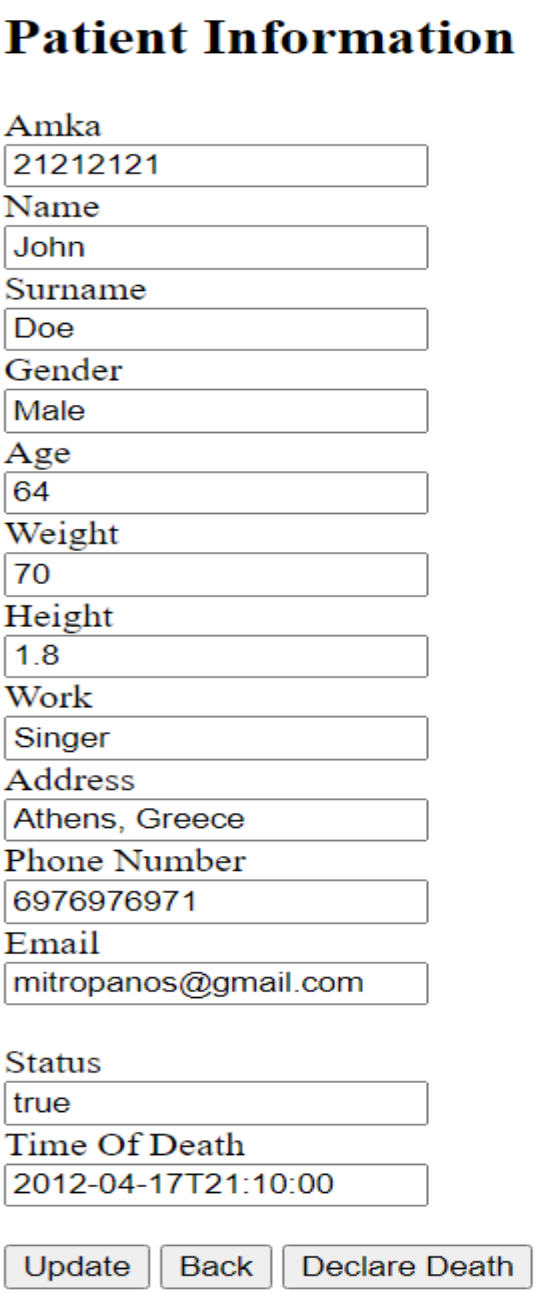

Figure 32: Φόρμα ενημέρωσης βασικών πληροφοριών ασθενούς

Με το πάτημα του κουμπιού «» , καλείται το endpoint /patient/{amka }/updatePatient

```
fetch(`http://localhost:8080/patient/${amka.value}/updatePatient`,{method: 'PUT',headers: {'Content-Type': 'application/json','Authorization': `Bearer ${bearerToken}`},body: data})
```

Στέλνοντας στο body , ότι πληροφορία παρείχε ο χρήστης στην φόρμα.

Το κουμπί «Declare Death» μας μεταφέρει στην σελίδα **DeclareDeath.html**, ενώ το «Back» μας επιστρέφει στην **PatientMedicalInfo.html**

ΚΕΦΑΛΑΙΟ 7 Συμπεράσματα

7.1 Οφέλη του ηλεκτρονικού ιατρικού φακέλου ασθενούς

Από τα παραπάνω κατέστη σαφές ότι ο χειρόγραφος ιατρικός φάκελος ασθενούς είναι αναποτελεσματικός τόσο κατά τη διάρκεια παρακολούθησης του ασθενούς όσο και τη θεραπευτική αγωγή που θα ακολουθήσει. Όπως είναι γνωστό, οι χειρόγραφοι ιατρικοί φάκελοι των νοσοκομείων βρίσκονται σε αποθήκες με κακή οργάνωση και αρχειοθέτηση, κακοσυντηρημένοι και πολλές φορές ο ιατρός ή το νοσηλευτικό προσωπικό αδυνατούν να τους βρουν για να εντοπίσουν τις πληροφορίες που χρειάζονται. Το αποτέλεσμα είναι να υπάρχει απογοήτευση και από τις δυο πλευρές, καθώς ούτε ο ιατρός μπορεί να επιτελέσει ορθά το λειτούργημά του, ούτε ο ασθενής να δεχτεί τη θεραπεία που του αξίζει.

Η ανάπτυξη ηλεκτρονικών φακέλων ασθενών μπορεί να συμβάλει στην βελτίωση των παροχών των υπηρεσιών υγείας.

Αναλυτικά, τα οφέλη που μπορούν να προκύψουν, είναι:

- **Ελαχιστοποίηση γραφειοκρατίας:** Αποφυγή χρονοβόρων διαδικασιών και προσκομίσεων δικαιολογητικών. Όλη η ζητούμενη πληροφορία μπορεί να ανακτηθεί από τον ηλεκτρονικό φάκελο του ασθενούς
- **Εξοικονόμηση αποθηκευτικού χώρου:** Μέχρι τώρα , οι ιατρικοί φάκελοι φυλάσσονταν στοιβαγμένοι σε δωμάτια αρχείου , καταλαμβάνοντας μεγάλή έκταση. Το σύνολο των ηλεκτρονικών ιατρικών φακέλων ασθενών , είναι αποθηκευμένοι σε βάσεις δεδομένων που καταλαμβάνουν ελάχιστο χώρο συγκριτικά με τους προκατόχους τους.
- **Ευκολία μεταφοράς δεδομένων:** Σε περιπτώσεις που είναι αναγκαία η μεταφορά δεδομένων του ασθενή , πχ ανάγκη μεταφοράς σε εξειδικευμένη ιατρική μονάδα, αποφεύγεται η μεταφορά ολόκληρων φυσικών φακέλων με την εξαγωγή και αποστολή της ζητούμενης πληροφορίας σε μορφή κρυπτογραφημένου ηλεκτρονικού αρχείου
- **Άμεση και γρήγορη πρόσβαση σε δεδομένα:** Ένας ιατρός , έχοντας ανά πάσα στιγμή , εύκολα και γρήγορα πρόσβαση στα ιατρικά δεδομένα του ασθενούς του , μπορεί να κάνει με μεγαλύτερη σιγουριά και πιο εμπεριστατωμένα την σωστή διάγνωση για τον ασθενή του.
- **Ασφάλεια δεδομένων:** Τα δεδομένα των ασθενών φυλάσσονται με ασφάλεια , μέσα σε μια κρυπτογραφημένη βάση δεδομένων
- **Έλεγχος πρόσβασης:** Πρόσβαση στα στοιχεία των ασθενών δίνεται σε εξειδικευμένο προσωπικό αφού για την προβολή τους απαιτείται είσοδος στο σύστημα με διαπιστευτήρια.
- **Ευκολία στην εξαγωγή στατιστικών στοιχείων:** Η διαδικασία δειγματοληψίας και διενέργεια στατιστικής έρευνας λαμβάνοντας δεδομένα από φυσικούς ηλεκτρονικούς φακέλους , είναι αρκετά χρονοβόρα και δεν εγγυάται ένα σωστό δείγμα. Η τήρηση των ιατρικών φακέλων σε ηλεκτρονική μορφή , μας δίνει τη δυνατότητα να εξάγουμε εύκολα στατιστικά στοιχεία και να κάνουμε συγκρίσεις όπου απαιτείται.

Για τους παραπάνω λόγους, ο ηλεκτρονικός ιατρικός φάκελος ασθενούς είναι ο μοναδικός τρόπος να λάβει ο ασθενής την θεραπευτική αγωγή που οφείλουν να του προσφέρουν. Η ευκολία δημιουργίας και χρήσης του σε συνδυασμό με την οργάνωσή του, καθώς -όπως ειπώθηκε- περιλαμβάνει τόσο το ιατρικό όσο και το φαρμακευτικό ιστορικό, συνηγορούν στη σωστή διάγνωση, αντιμετώπιση και παρακολούθηση της εξέλιξης της υγείας του ασθενούς, ώστε ο ιατρός να εξάγει βέβαια και ασφαλή συμπεράσματά.

Εν κατακλείδι, κρίνεται επιβεβλημένος ο άμεσος ψηφιακός μετασχηματισμός του Εθνικού Συστήματος Υγείας μας. Τα πληροφοριακά συστήματα των Νοσοκομείων χρήζουν ποιοτικής αναβάθμισης την ίδια στιγμή που το προσωπικό οφείλει να εκπαιδευτεί στις νέες ψηφιακές δεξιότητες. Σε αυτήν την κατεύθυνση προτείνουμε την υιοθέτηση και εφαρμογή του ηλεκτρονικού φακέλου ασθενούς. Φρονούμε ότι μπορεί να αποτελέσει το εργαλείο που θα εξασφαλίσει άμεσες και ποιοτικές ιατρικές υπηρεσίες στους πολίτες ενώ την ίδια στιγμή θα καταστήσει τα νοσοκομεία λειτουργικά και αποτελεσματικά. Το εργαλείο εκείνο που θα θέσει το ελληνικό Υγειονομικό Σύστημα σε τροχιά εκσυγχρονισμού ως προς την οργάνωση και διοίκηση του κεφαλαιώδους Τομέα της Υγείας.

7.2 Επεκτάσεις στην ανάπτυξη ηλεκτρονικού φακέλου με τη χρήση spring boot.

Η παρούσα διπλωματική εργασία, είναι μία προσπάθεια υλοποίησης ηλεκτρονικού ιατρικού φακέλου με τη τεχνολογία της spring boot. Σε πρώτη φάση , το εργαλείο αποθηκεύει και ανακτά ασθενείς, επιτρέπει την πρόσβαση σε βασικές πληροφορίες όπως η φαρμακευτική αγωγή και οι τιμές των βιοχημικών εξετάσεων καθώς και την εξαγωγή στατιστικών για εισιτήρια , εξιτήρια και θανάτους στην ιατρική μονάδα. Σε μελλοντική φάση, θα μπορούσε να υλοποιηθεί μια λύση η οποία επεκτείνει τις δυνατότητες αποθήκευσης πληροφορίας καθώς και την δυνατότητα εξαγωγής πιο σύνθετων στατιστικών στοιχείων.

Πιο συγκεκριμένα , μερικές από τις μελλοντικές προσθήκες θα μπορούσαν να είναι:

- Δυνατότητα αποθήκευσης αρχείων εικόνας και βίντεο με κάποιο πρωτόκολλο μεταφοράς αρχείων(FTP)
- Δυνατότητα εξαγωγής στατιστικών στοιχείων σχετικά με τη φαρμακευτική αγωγή των ασθενών που εισέρχονται στην ιατρική μονάδα
- Δυνατότητα γραφικής αναπαράστασης της πορείας των τιμών βιοχημικών εξετάσεων σε βάθος χρόνου για κάποιον ασθενή
- Δυνατότητα σύγκρισης τιμών βιοχημικών εξετάσεων μεταξύ ασθενών με παρόμοια πάθηση με στόχο την άμεση προσαρμογή της φαρμακευτικής αγωγής

ΚΕΦΑΛΑΙΟ 8 Βιβλιογραφία

[1]Σημαιόπουλος Βασίλειος, Ηλεκτρονικός Φάκελος Ασθενούς [Online]. Available: https://dione.lib.unipi.gr/xmlui/bitstream/handle/unipi/10282/Simaiopoulos_Vasileios.pdf?sequence=1&isAllowed=y

[2] Αγγελίδης Παντελής, Εφαρμογές πληροφορικής στην στοματική υγεία [Online]. Available: https://eclass.uowm.gr/modules/document/file.php/ICTE276/ΕΡΓΑΣΤΗΡΙΟ/ΠΑΡΟΥΣΙΑΣΕΙΣ/Εργαστήριο_1ο_Εισαγωγή_oc.pdf

[3] Προσπαθόπουλος Δημήτριος, Σαμαράς Βασίλειος, Ευέλικτος Απεικονιστικός Ιατρικός Φάκελος Ασθενή ως υποστηρικτικό εκπαιδευτικό μέσο στο χώρο της ιατρικής [Online]. Available: http://ikee.lib.auth.gr/record/290638/files/IatrikosFakelos.pdf

[4] Πετρούτσος Νικόλαος, Ηλεκτρονικός Φάκελος Ασθενή [Online]. Available: https://dione.lib.unipi.gr/xmlui/bitstream/handle/unipi/12721/Petroutsos_1835.pdf.pdf?sequence=1&isAllowed=y

[5]"Spring Framework", bmc.com. [Online]. Available: https://www.bmc.com/blogs/spring-framework/.

[6]"Spring Boot JPA", spring.io. [Online]. Available: https://spring.io/projects/spring-data-jpa

[7]"Spring Boot JPA", tutorialspoint.com. [Online]. Available: https://www.tutorialspoint.com/jpa/index.htm

[8]"Spring Boot", tutorialspoint.com. [Online]. Available: https://www.tutorialspoint.com/spring_boot/spring_boot_quick_guide.htm

[9]"Spring Data Rest", spring.io. [Online]. Available: https://spring.io/projects/spring-data-rest

[10]"Spring Boot OAuth02 resource server", docs.spring.io. [Online]. Available: https://docs.spring.io/spring-security/reference/servlet/oauth2/resource-server/index.html

[11]"JWT", jwt.io. [Online]. Available: https://jwt.io/introduction

[12]"H2 Database", h2database.com. [Online]. Available: https://www.h2database.com/html/main.html

[13]"JavaScript", en.wikipedia.org. [Online]. Available: https://en.wikipedia.org/wiki/JavaScript

[14]"HTML5", en.wikipedia.org. [Online]. Available: https://en.wikipedia.org/wiki/HTML