ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Σχεδιασμός, μελέτη και κατασκευή ενός διαγνωστικού συστήματος για την πορεία ενός ανθρώπινου οργανισμού

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

του

ΓΚΟΥΓΚΟΣΙΑ ΚΩΝΣΤΑΝΤΙΝΟΥ

(ΑΕΜ: 2293)

Επιβλέπων : Δ. Μιχαήλ Δόσης

Καστοριά Δεκέμβριος 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Σχεδιασμός, μελέτη και κατασκευή ενός διαγνωστικού συστήματος για την πορεία ενός ανθρώπινου οργανισμού

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

του

ΓΚΟΥΓΚΟΣΙΑ ΚΩΝΣΤΑΝΤΙΝΟΥ

(ΑΕΜ: 2293)

Επιβλέπων : Δρ. Μιχαήλ Δόσης

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την ημερομηνία εξέτασης

.....	.....	.....
Ον/μο Μέλους	Ον/μο Μέλους	Ον/μο Μέλους
Ιδίοτητα Μέλους	Ιδίοτητα Μέλους	Ιδίοτητα Μέλους

Καστοριά Δεκέμβριος 2022

Ευχαριστίες

Για την ολοκλήρωση αυτής της πτυχιακής εργασίας οφείλω να ευχαριστήσω τον επιβλέποντα καθηγήτη μου κύριο Μπάτο Παναγιώτη για τις χρήσιμες συμβουλές και τις υποδείξεις του , καθώς και για την καθοδήγηση που μου παρείχε.

Περίληψη

Ο σκοπός της πτυχιακής εργασίας είναι η ανάπτυξη ενός προγράμματος διαγνωστικού ελέγχου, το οποίο μπορεί να δώσει άμεσα πληροφορίες στον δυνητικό χρήστη της εφαρμογής. Η εφαρμογή θα προσφέρει μία ευρύτερη εικόνα της κατάστασης του ανθρώπινου οργανισμού σύμφωνα πάντα με την ποιότητα ζωής και την κατάσταση σε παθήσεις αυτού του ατόμου.

Υπάρχουν δεκάδες διαθέσιμες εφαρμογές στα κινητά τηλέφωνα αλλά και στους υπολογιστές, οι οποίες δεν είναι ακόμη σε θέση να αντικαταστήσουν την ιατρική επίσκεψη ή να αναγνωρίσουν σοβαρά ιατρικά ζητήματα που μόνος ένας ειδήμον μπορεί να κάνει. Συγκεφαλαιώνοντας, η ανάπτυξη της εφαρμογής για την πτυχιακή εργασία ξεδιπλώνει ένα πολύ μικρό κεφάλαιο της ιατρικής πρόληψης που σε σύντομο χρονικό διάστημα μπορεί να προσφέρει μια σφαιρική άποψη στους ενδιαφερόμενους χρήστες

Λέξεις κλειδιά:

Διαγνωστικό σύστημα,άνρθωπος,υγεία,ασθενής,λογισμικά,εφαρμογές, κώδικας,τεχνολογία,ιατρική επιστήμη,ανρθώπινο σώμα.

Abstract

The purpose of the thesis is the development of a diagnostic control program, which can provide immediate information to its potential user. The application will offer a broader picture of the state of the human organism according to the quality of life and the state of diseases of this person.

There are dozens of apps available on mobile phones as well as computers, which are still not able to replace a doctor's visit or identify serious medical issues that only a specialist can do. In summary, the application development for the thesis unfolds a very small chapter of preventive medicine that in a short period of time can provide a global view to interested users.

Key words:

Diagnostic system,human,health,patient,software,applications,code,technology, medical science,human body.

Πίνακας Περιεχομένων

Λίστα Εικόνων

Εισαγωγή

Με την πάροδο του χρόνου, η εξέλιξη της επιστήμης και της τεχνολογίας αυξάνεται με ραγδαίους ρυθμούς και φυσικά ο ρυθμός ανάπτυξης νέων μεθόδων και εφαρμογών στον κλάδο της πληροφορικής έχουν βελτιώσει δραστικά την καθημερινότητα των ανθρώπων. Αυτό όμως που δεν συναντούμε συχνά, είναι εμπορικές εφαρμογές που να στοχεύουν στην αυτοβελτίωση του ανθρώπου και του ανθρώπινου οργανισμού. Σχεδόν όλοι οι άνθρωποι παγκοσμίως χρησιμοποιούν καθημερινά ηλεκτρονικούς υπολογιστές με τη χρήση του ίντερνετ.

Πόσο μάλλον που πλέον ζούμε στην καμπή της τεχνολογίας που εκτός από καλούς υπολογιστές και ταχύτατο δίκτυο, έχουμε και ισχυρά τηλέφωνα. Η χρήση των συσκευών που προαναφέρθηκαν, δεν έχουν πάντα τον ρόλο ενός εργαλείου που μας εξυπηρετεί να λύσουμε προβλήματα αλλά χρησιμοποιούνται κυρίως ως πηγή διασκέδασης. Αυτό και μόνο δεν αρκεί. Είναι σημαντικό να υπάρχει προσιτό λογισμικό σε όλες τις πλατφόρμες που εκτός από τέτοιου είδους περιεχόμενο, να υπάρχουν και εφαρμογές που να υπενθυμίζουν στους ανθρώπους ότι πρέπει να παραμείνουν υγιείς.

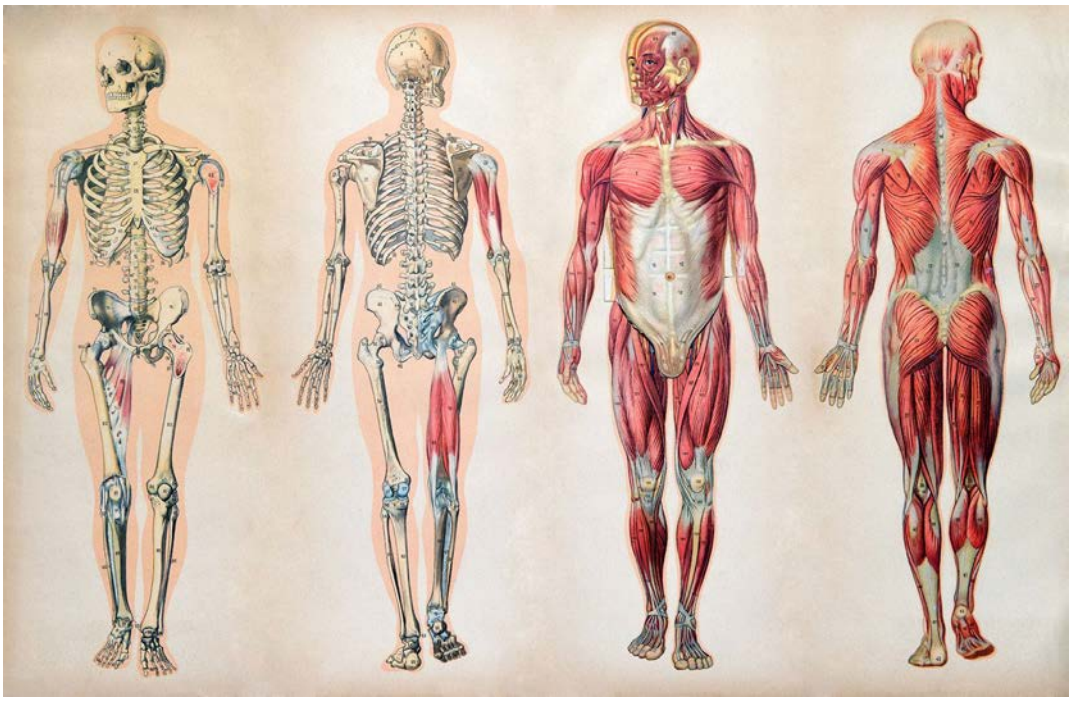
Φυσικά και υπάρχουν λογισμικά που να εφαρμόζουν τέτοιες πρακτικές αλλά από ότι έχει παρατηρηθεί δεν έχουν επιβεβαιωθεί από τους χρήστες αυτά που υπόσχονται ότι κάνουν. Ο άνθρωπος διαθέτει περιέργεια από τη φύση του και όταν θα χρειαστεί να αναζητήσει κάτι κρίσιμο που ενδεχομένως να αφορά την υγεία του, αυτομάτως απευθύνεται στην ωμή και ακατέργαστη πολλές φορές πληροφορία που υπάρχει στο διαδίκτυο. Την πληροφορία δε μπορεί πάντα να την φιλτράρει ο ανθρώπινος εγκέφαλος με λογική και ηρεμία.

Αυτή η διαδικασία μπορεί να παραπλανήσει τον άνθρωπο, να τον μπερδέψει .Σύμφωνα πάντα με το τι αναζητά! Αν για παράδειγμα ένας άνθρωπος πάσχει από μία νόσο, πρέπει να απευθυνθεί άμεσα σε κάποιον γιατρό και όχι σε μία μηχανή αναζήτησης.

Με αφορμή αυτό το σημαντικό θέμα, είναι απαραίτητη η ανάπτυξη εξειδικευμένου λογισμικού παράγοντας μελετημένα και σωστά προσεγμένα αποτελέσματα και ταυτόχρονα η χρήση του λογισμικού να είναι απλή για όλους τους ανθρώπους/χρήστες.

ΚΕΦΑΛΑΙΟ 1: Ανθρώπινος Οργανισμός

Όταν θέλουμε να μιλήσουμε για τον ανθρώπινο οργανισμό, αυτομάτως πρέπει να σκεφτόμαστε και να ταυτίζουμε τον οργανισμό με τον ανθρώπινο σώμα. Ο οργανισμός είναι μέρος αυτού, καθώς εάν θέλαμε με απλό τρόπο να δώσουμε έναν ορισμό, θα έλεγε κανείς το εξής: *Η φυσική ουσία του ανθρώπινου οργανισμού, αποτελείται από ζωντανά κύτταρα και εξωκυττάρια υλικά και οργανώνεται σε ιστούς,* ***όργανα*** *και συστήματα (Encyclopedia Britannica).* Εν ολίγοις, για να είναι ένας ανθρώπινος οργανισμός υγιής, πρέπει και τα όργανα να βρίσκονται σε υγιή κατάσταση.



Εικόνα 1: Ανθρώπινο σώμα και ανατομία.

Όσο όμοιοι κι αν είμαστε οι άνθρωποι (από πολλές απόψεις) με τα υπόλοιπα είδη, θα μπορούσαμε να αναφέρουμε πως είμαστε μοναδικοί μεταξύ των μορφών ζωής της γης στην ικανότητά μας να χρησιμοποιούμε τη γλώσσα και τη σκέψη. Εδώ και χιλιάδες χρόνια, ο άνθρωπος έχει εξελιχθεί και διαθέτει ένα πολύπλοκο και ταυτόχρονα εύθραυστο όργανο, τον **εγκέφαλο**. Το είδος μας έχει τη δυνατότητα να σκέφτεται, να φαντάζεται, να δημιουργεί και να μαθαίνει από την εμπειρία που ξεπερνά κατά πολύ αυτή οποιουδήποτε άλλου είδους. Χρησιμοποιήσαμε αυτή την ικανότητα για να δημιουργήσουμε τεχνολογίες και

λογοτεχνικά και καλλιτεχνικά έργα σε τεράστια κλίμακα και να αναπτύξουμε μια επιστημονική κατανόηση του εαυτού μας και του κόσμου.[4]

1.1 Η ανθρώπινη ταυτότητα

Από τις περισσότερες βιολογικές απόψεις, οι άνθρωποι είναι σαν άλλους ζωντανούς οργανισμούς. Για παράδειγμα, αποτελούνται από κύτταρα όπως αυτά άλλων ζώων, έχουν σχεδόν την ίδια χημική σύνθεση, έχουν συστήματα οργάνων και φυσικά χαρακτηριστικά όπως πολλά άλλα, αναπαράγονται με παρόμοιο τρόπο, φέρουν το ίδιο είδος συστήματος γενετικών πληροφοριών και αποτελούν μέρος ενός τροφικού ιστού.

Απολιθώματα και μοριακά στοιχεία υποστηρίζουν ότι το ανθρώπινο είδος εξελίχθηκε από άλλους οργανισμούς. Τα στοιχεία συνεχίζουν να συσσωρεύονται και οι επιστήμονες συνεχίζουν να συζητούν για τις ημερομηνίες και την καταγωγή, αλλά σε γενικότερο πλαίσιο οι περισσότερες πληροφορίες ιστορικά έγιναν αποδεκτές.

- **Ανθρωποειδής πίθηκος**: η ταξινόμηση παρόμοιων οργανισμών που περιλαμβάνει ανθρώπους, πιθήκους, και πολλά άλλα είδη θηλαστικών- άρχισε να εξελίσσεται από άλλα θηλαστικά πριν από λιγότερο από 100 εκατομμύρια χρόνια. Αρκετά είδη πρωτευόντων που μοιάζουν με τον άνθρωπο άρχισαν να εμφανίζονται και να διακλαδίζονται πριν από περίπου 5 εκατομμύρια χρόνια, αλλά όλα εκτός από ένα εξαφανίστηκαν. Το είδος που επέζησε οδήγησε στο σύγχρονο ανθρώπινο είδος.

Όπως και άλλοι πολύπλοκοι οργανισμοί, οι άνθρωποι διαφέρουν ως προς το μέγεθος και το σχήμα, το χρώμα του δέρματος, τις αναλογίες του σώματος, τις τρίχες του σώματος, τα χαρακτηριστικά του προσώπου, τη μυϊκή δύναμη, τις χειρονομίες και ούτω καθεξής. Αλλά αυτές οι διαφορές είναι μικρές σε σύγκριση με την εσωτερική ομοιότητα όλων των ανθρώπων, όπως αποδεικνύεται από το γεγονός ότι οι άνθρωποι από οπουδήποτε στον κόσμο μπορούν να αναμειχθούν φυσικά με βάση την αναπαραγωγή, τις μεταγγίσεις αίματος και τις μεταμοσχεύσεις οργάνων.

Ένα από τα πιο σημαντικά γεγονότα στην ιστορία του ανθρώπινου είδους πριν από περίπου 10.000 ήταν η μετάβαση από το κυνήγι και τη συλλογή στη

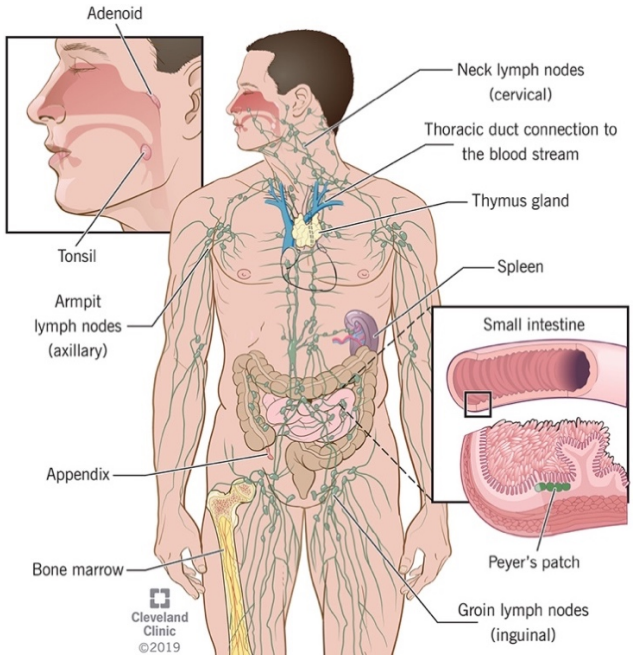
γεωργία, γεγονός που κατέστησε δυνατή την ταχεία αύξηση του πληθυσμού. Κατά τη διάρκεια αυτής της πρώιμης περιόδου ανάπτυξης, η κοινωνική εφευρετικότητα του ανθρώπινου είδους άρχισε να παράγει χωριά και στη συνέχεια πόλεις. Πρόσφατα, η μεγαλύτερη αποτελεσματικότητα της γεωργίας και ο έλεγχος των **μολυσματικών ασθενειών** έχει επιταχύνει περαιτέρω την ανάπτυξη του ανθρώπινου πληθυσμού, ο οποίος σήμερα ξεπερνά τα επτά δισεκατομμύρια. Όπως το είδος μας είναι βιολογικό, κοινωνικό και πολιτιστικό, έτσι είναι και τεχνολογικό. Σε σύγκριση με άλλα είδη, δεν είμαστε τίποτα ιδιαίτερο όσον αφορά την ταχύτητα, την ευκινησία, τη δύναμη, την αντοχή, την όραση, την ακοή ή την ικανότητα να αντέχουμε ακραίες περιβαλλοντικές συνθήκες. Μια ποικιλία τεχνολογιών, ωστόσο, βελτιώνει την ικανότητά μας να αλληλοεπιδρούμε με τον φυσικό κόσμο. Κατά μία έννοια, οι εφευρέσεις μας μάς βοήθησαν να καλύψουμε τα βιολογικά μας μειονεκτήματα. Παραδείγματος χάρη οι προσθετικές συσκευές και η χημική και χειρουργική επέμβαση δίνουν τη δυνατότητα στα άτομα με σωματικές αναπηρίες να λειτουργούν αποτελεσματικά στο περιβάλλον τους.[4]

1.1.1 Βασικές λειτουργίες του οργανισμού

Το ανθρώπινο σώμα είναι ένα πολύπλοκο σύστημα κυττάρων, τα περισσότερα από τα οποία ομαδοποιούνται σε συστήματα οργάνων που έχουν εξειδικευμένες λειτουργίες. Αυτά τα συστήματα μπορούν να κατανοηθούν καλύτερα από την άποψη των βασικών λειτουργιών που εξυπηρετούν: παραγωγή ενέργειας από τα τρόφιμα, προστασία από τραυματισμούς, εσωτερικός συντονισμός και αναπαραγωγή.

Η συνεχής ανάγκη για ενέργεια δεσμεύει τις αισθήσεις και τους σκελετικούς μύες στην απόκτηση τροφής, το πεπτικό σύστημα στη διάσπαση της τροφής σε χρησιμοποιήσιμες ενώσεις και στην απόρριψη άπεπτων υλικών τροφίμων, τους πνεύμονες στην παροχή οξυγόνου για την καύση των τροφών και την εκκένωση του διοξειδίου του άνθρακα, το ουροποιητικό σύστημα για την απόρριψη άλλων διαλυμένων υπολειμμάτων της κυτταρικής δραστηριότητας, το δέρμα και τους πνεύμονες για την απαλλαγή από την υπερβολική θερμότητα (στην οποία τελικά αποδομείται το μεγαλύτερο μέρος της ενέργειας των τροφίμων) και το κυκλοφορικό σύστημα για τη μεταφορά όλων αυτών των ουσιών προς ή από τα κύτταρα όπου χρειάζονται ή παράγονται.

Όπως όλοι οι οργανισμοί, οι άνθρωποι έχουν τα μέσα να προστατεύονται. Η αυτοπροστασία περιλαμβάνει τη χρήση των αισθήσεων για την ανίχνευση κινδύνου, το ορμονικό σύστημα για την τόνωση της καρδιάς και την απόκτηση πρόσβασης σε εφόδια ενέργειας έκτακτης ανάγκης και των μυών σε διαφυγή ή άμυνα. Το δέρμα παρέχει ασπίδα ενάντια σε επιβλαβείς ουσίες και οργανισμούς, όπως βακτήρια και παράσιτα. Το **ανοσοποιητικό σύστημα** παρέχει προστασία από τις ουσίες που εισχωρούν στο σώμα και από τα καρκινικά κύτταρα που αναπτύσσονται αυθόρμητα στο σώμα. Το νευρικό σύστημα παίζει ιδιαίτερα σημαντικό ρόλο στην επιβίωση. Παίζει σημαντικό ρόλο στη μάθηση που χρειάζονται οι άνθρωποι για να αντιμετωπίσουν τις αλλαγές στο περιβάλλον τους.



Εικόνα 2: Το ανοσοποιητικό σύστημα

Ο εσωτερικός έλεγχος που απαιτείται για τη διαχείριση και τον συντονισμό αυτών των πολύπλοκων συστημάτων πραγματοποιείται από τον εγκέφαλο και το νευρικό σύστημα σε συνδυασμό με τους αδένες που εκκρίνουν ορμόνες. Τα ηλεκτρικά και χημικά σήματα που μεταφέρονται από τα νεύρα και τις ορμόνες ενσωματώνουν το σώμα ως σύνολο. Οι πολλές διασταυρούμενες επιρροές μεταξύ των ορμονών και των νεύρων δημιουργούν ένα σύστημα συντονισμένων κύκλων σε όλες σχεδόν τις λειτουργίες του σώματος.

Τα νεύρα μπορούν να διεγείρουν ορισμένους αδένες για να εκκρίνουν ορμόνες, ορισμένες ορμόνες επηρεάζουν τα εγκεφαλικά κύτταρα, ο ίδιος ο εγκέ-

φαλος απελευθερώνει ορμόνες που επηρεάζουν την ανθρώπινη συμπεριφορά και οι ορμόνες εμπλέκονται στη μετάδοση σημάτων μεταξύ των νευρικών κυττάρων. Ορισμένα φάρμακα —νόμιμα και παράνομα— μπορούν να επηρεάσουν το ανθρώπινο σώμα και τον εγκέφαλο μιμούμενοι ή μπλοκάροντας τις ορμόνες και τους νευροδιαβιβαστές που παράγονται από το ορμονικό και το νευρικό σύστημα.

Η αναπαραγωγή εξασφαλίζει τη συνέχιση του είδους. Η σεξουαλική παρόρμηση καθοδηγείται βιολογικά, αλλά το πώς αυτή η ορμή εκδηλώνεται στους ανθρώπους καθορίζεται από ψυχολογικούς και πολιτισμικούς παράγοντες. Εμπλέκονται τα αισθητήρια όργανα και οι ορμόνες, καθώς και τα ίδια τα εσωτερικά και εξωτερικά γεννητικά όργανα. Το γεγονός ότι η σεξουαλική αναπαραγωγή παράγει μια μεγαλύτερη γενετική παραλλαγή με την ανάμειξη των γονιδίων των γονέων παίζει βασικό ρόλο στην εξέλιξη.[4]

1.1.2 Υγεία του σώματος

Για να παραμείνει σε καλή λειτουργική κατάσταση το ανθρώπινο σώμα απαιτεί ποικιλία τροφών και εμπειριών. Η ποσότητα της τροφικής ενέργειας (θερμίδες) που χρειάζεται ένα άτομο ποικίλλει ανάλογα με το μέγεθος του σώματος, την ηλικία, το φύλο, το επίπεδο δραστηριότητας και τον μεταβολικό ρυθμό. Πέρα από την ενέργεια, η κανονική λειτουργία του σώματος απαιτεί ουσίες που πρέπει να προστεθούν ή να αντικαταστήσουν τα υλικά από τα οποία είναι φτιαγμένο: ακόρεστα λίπη, ίχνη δώδεκα στοιχείων των οποίων τα άτομα παίζουν βασικούς ρόλους και μερικά ίχνη ουσιών που τα ανθρώπινα κύτταρα δεν μπορούν να συνθέσουν—συμπεριλαμβανομένων ορισμένων αμινοξέα και βιταμίνες.

Η φυσιολογική κατάσταση των περισσότερων συστημάτων του σώματος απαιτεί να εκτελούν την προσαρμοστική τους λειτουργία: Για παράδειγμα, οι μύες πρέπει να ασκούν κίνηση, τα οστά πρέπει να φέρουν φορτία και η καρδιά πρέπει να αντλεί αίμα αποτελεσματικά. Η τακτική άσκηση, επομένως, είναι σημαντική για τη διατήρηση ενός υγιούς συστήματος καρδιάς/πνεύμονα, για τη διατήρηση του μυϊκού τόνου και για την προστασία των οστών από το να γίνουν εύθραυστα.

Η καλή υγεία εξαρτάται επίσης από την αποφυγή της υπερβολικής έκθεσης σε **ουσίες** που παρεμβαίνουν στη λειτουργία του οργανισμού. Τα κυριότε-

ρα από αυτά που μπορεί να ελέγξει κάθε άτομο είναι το κάπνισμα (που εμπλέκεται στον καρκίνο του πνεύμονα, το εμφύσημα και οι καρδιακές παθήσεις), τα εθιστικά **φάρμακα** (που εμπλέκονται σε ψυχικό αποπροσανατολισμό και διαταραχές του νευρικού συστήματος) και η υπερβολική ποσότητα **αλκοόλ** (που έχει αρνητικές επιπτώσεις στο ήπαρ, στον εγκέφαλο και στην καρδιά). Επιπλέον, το περιβάλλον μπορεί να περιέχει επικίνδυνα επίπεδα ουσιών (όπως μόλυβδος, ορισμένα φυτοφάρμακα και ραδιενεργά ισότοπα) που μπορεί να είναι επιβλαβή για τον άνθρωπο.

Άλλοι οργανισμοί μπορούν επίσης να επηρεάσουν την κανονική λειτουργία του ανθρώπινου σώματος. Ορισμένα είδη βακτηρίων ή μυκήτων μπορεί να μολύνουν το σώμα για να σχηματίσουν αποικίες σε προτιμώμενα όργανα ή ιστούς. Οι ιοί εισβάλλουν στα υγιή κύτταρα και τα αναγκάζουν να συνθέσουν περισσότερους ιούς, σκοτώνοντας συνήθως αυτά τα κύτταρα στη διαδικασία. Μολυσματική ασθένεια μπορεί επίσης να προκληθεί από ζωικά παράσιτα, τα οποία μπορεί να εγκατασταθούν στα έντερα, στην κυκλοφορία του αίματος ή στους ιστούς.

Η πρώτη γραμμή άμυνας του ίδιου του οργανισμού έναντι των μολυσματικών παραγόντων είναι να τους εμποδίζει να εισέλθουν ή να καθιζάνουν στο σώμα. Οι προστατευτικοί μηχανισμοί περιλαμβάνουν το δέρμα για να τα μπλοκάρει, τα δάκρυα και το σάλιο για τη διεξαγωγή τους και τις στομαχικές και κολπικές εκκρίσεις για να τα σκοτώσουν. Τα σχετικά μέσα προστασίας από τους εισβολείς οργανισμούς περιλαμβάνουν τη διατήρηση του δέρματος καθαρό, τη σωστή διατροφή, την αποφυγή μολυσμένων τροφίμων και υγρών και γενικά την αποφυγή της άσκοπης έκθεσης σε ασθένειες.

Η επόμενη γραμμή άμυνας του οργανισμού είναι το ανοσοποιητικό σύστημα. Τα λευκά αιμοσφαίρια δρουν τόσο για να περιβάλλουν τους εισβολείς όσο και για να παράγουν συγκεκριμένα αντισώματα που θα τους επιτεθούν (ή θα διευκολύνουν την επίθεση από άλλα λευκά αιμοσφαίρια). Εάν το άτομο επιβιώσει από την εισβολή, ορισμένα από αυτά τα αντισώματα παραμένουν - μαζί με την ικανότητα να παράγει γρήγορα πολλά περισσότερα. Για χρόνια μετά, ή ακόμα και για μια ζωή, το ανοσοποιητικό σύστημα θα είναι έτοιμο για αυτόν τον τύπο οργανισμού και θα είναι σε θέση να περιορίσει ή να αποτρέψει την ασθένεια. Ένα άτομο μπορεί να «κρυώσει» πολλές φορές γιατί υπάρχουν πολλές ποικιλίες μικροβίων που προκαλούν παρόμοια συμπτώματα.

Οι αλλεργικές αντιδράσεις προκαλούνται από ασυνήθιστα ισχυρές ανοσολογικές αντιδράσεις σε ορισμένες περιβαλλοντικές ουσίες, όπως αυτές που βρίσκονται στη γύρη, στις τρίχες ζώων ή σε ορισμένα τρόφιμα. Μερικές φορές το ανθρώπινο ανοσοποιητικό σύστημα μπορεί να δυσλειτουργήσει και να επιτεθεί ακόμη και σε υγιή κύτταρα. Ορισμένες ιογενείς ασθένειες, όπως το AIDS, καταστρέφουν κρίσιμα κύτταρα του ανοσοποιητικού συστήματος, αφήνοντας τον οργανισμό αβοήθητο στην αντιμετώπιση πολλαπλών μολυσματικών παραγόντων και καρκινικών κυττάρων.

Ωστόσο, οι μολυσματικές ασθένειες δεν είναι η μόνη απειλή για την ανθρώπινη υγεία. Τα μέρη ή τα συστήματα του σώματος μπορεί να αναπτύξουν μειωμένη λειτουργία για εντελώς εσωτερικούς λόγους. Ορισμένες λανθασμένες λειτουργίες των διεργασιών του σώματος είναι γνωστό ότι προκαλούνται από αποκλίνοντα γονίδια. Μπορεί να έχουν ένα άμεσο, προφανές αποτέλεσμα, όπως η πρόκληση εύκολης αιμορραγίας ή μπορεί μόνο να αυξήσουν την ευαισθησία του σώματος στην ανάπτυξη συγκεκριμένων ασθενειών, όπως βουλωμένες αρτηρίες ή ψυχική κατάθλιψη.

Τέτοια γονίδια μπορεί να είναι κληρονομικά ή μπορεί να προκύψουν από μετάλλαξη σε ένα κύτταρο ή σε λίγα κύτταρα κατά τη διάρκεια της ανάπτυξης του ίδιου του ατόμου. Επειδή ένα γονίδιο ενός ζεύγους που λειτουργεί σωστά μπορεί να είναι αρκετό για να εκτελέσει τη λειτουργία του γονιδίου, πολλές γενετικές ασθένειες δεν εμφανίζονται εκτός εάν μια ελαττωματική μορφή του γονιδίου κληρονομηθεί και από τους δύο γονείς (οι οποίοι, για τον ίδιο λόγο, μπορεί να μην είχαν συμπτώματα του η ίδια η ασθένεια).

Το γεγονός ότι οι περισσότεροι άνθρωποι ζουν τώρα σε σωματικά και κοινωνικά περιβάλλοντα που είναι πολύ διαφορετικά από εκείνα στα οποία είχε προσαρμοστεί η ανθρώπινη φυσιολογία πριν από πολύ καιρό είναι ένας παράγοντας που καθορίζει την υγεία του πληθυσμού γενικά. Μια σύγχρονη «ανωμαλία» στις βιομηχανικές χώρες είναι η διατροφή, η οποία κάποτε περιελάβανε κυρίως ακατέργαστα φυτικά και ζωικά υλικά, αλλά τώρα περιλαμβάνει υπερβολικές ποσότητες ραφιναρισμένης ζάχαρης, κορεσμένων λιπαρών και αλατιού, καθώς και καφεΐνη, αλκοόλ, νικοτίνη και άλλα φάρμακα. Η έλλειψη άσκησης είναι μια άλλη αλλαγή από τον πολύ πιο ενεργό τρόπο ζωής της προϊστορίας.

Υπάρχουν επίσης περιβαλλοντικοί ρύποι και το ψυχολογικό άγχος της ζωής σε ένα πολυσύχναστο, ταραχώδες και ταχέως μεταβαλλόμενο κοινωνικό

περιβάλλον. Από την άλλη πλευρά, οι νέες ιατρικές τεχνικές, τα αποτελεσματικά συστήματα παροχής υγειονομικής περίθαλψης, η βελτιωμένη υγιεινή και η πληρέστερη κατανόηση της φύσης της νόσου από το κοινό δίνουν στους σημερινούς ανθρώπους καλύτερες πιθανότητες να παραμείνουν υγιείς από ότι είχαν οι πρόγονοί τους.[4]

1.1.3 Ψυχική υγεία

Η καλή ψυχική υγεία περιλαμβάνει την αλληλεπίδραση ψυχολογικών, βιολογικών, φυσιολογικών, κοινωνικών και πολιτισμικών συστημάτων. Γενικά θεωρείται ως η ικανότητα αντιμετώπισης των συνηθισμένων περιστάσεων που αντιμετωπίζουν οι άνθρωποι στην προσωπική, επαγγελματική και κοινωνική τους ζωή.

Ωστόσο, οι ιδέες για το τι συνιστά καλή ψυχική υγεία ποικίλλουν από τον έναν πολιτισμό στον άλλο και από τη μια χρονική περίοδο στην άλλη. Η συμπεριφορά που μπορεί να θεωρηθεί ως απόλυτη παράνοια σε έναν πολιτισμό μπορεί να θεωρηθεί σε έναν άλλο ως απλώς εκκεντρικότητα ή ακόμη και ως θεϊκή έμπνευση. Σε ορισμένους πολιτισμούς, τα άτομα μπορεί να ταξινομηθούν ως ψυχικά άρρωστα εάν εκφράζουν επίμονα διαφωνία με θρησκευτικές ή πολιτικές αρχές. Οι ιδέες για το τι συνιστά σωστή θεραπεία για μη φυσιολογικές ψυχικές καταστάσεις διαφέρουν επίσης. Τα στοιχεία μη φυσιολογικής σκέψης που θα τιμωρούνταν εσκεμμένα σε έναν πολιτισμό μπορεί να αντιμετωπίζονται σε άλλους πολιτισμούς με κοινωνική συμμετοχή, απομόνωση, αυξημένη κοινωνική υποστήριξη, προσευχές, εκτενείς συνεντεύξεις ή ιατρικές διαδικασίες.

Τα άτομα διαφέρουν πολύ ως προς την ικανότητά τους να αντιμετωπίζουν στρεσογόνα περιβάλλοντα. Το άγχος στην παιδική ηλικία μπορεί να είναι ιδιαίτερα δύσκολο να αντιμετωπιστεί και, επειδή μπορεί να διαμορφώσει τη μετέπειτα εμπειρία και σκέψη του παιδιού, μπορεί να έχει μακροχρόνιες επιπτώσεις στην ψυχολογική υγεία και την κοινωνική προσαρμογή του ατόμου. Οι άνθρωποι διαφέρουν επίσης στο πόσο καλά μπορούν να αντιμετωπίσουν την ψυχολογική διαταραχή όταν αυτή εμφανίζεται.

Συχνά, οι άνθρωποι αντιδρούν στην ψυχική δυσφορία αρνούμενοι ότι έχουν κάποιο ψυχολογικό πρόβλημα. Ακόμη και όταν οι άνθρωποι αναγνωρίζουν ότι έχουν ένα τέτοιο πρόβλημα, μπορεί να μην έχουν τα χρήματα, τον χρόνο ή την κοινωνική υποστήριξη που απαιτείται για να αναζητήσουν βοήθει-

α. Η παρατεταμένη διαταραχή της συμπεριφοράς μπορεί να οδηγήσει σε έντονες αντιδράσεις από τις οικογένειες, τους προϊστάμενους της εργασίας και τις δημόσιες αρχές που αυξάνουν το άγχος στο άτομο.

Η διάγνωση και η θεραπεία των ψυχικών διαταραχών μπορεί να είναι ιδιαίτερα δύσκολη επειδή μεγάλο μέρος της ψυχικής ζωής των ανθρώπων δεν είναι συνήθως προσβάσιμο ούτε σε αυτούς. Όταν θυμόμαστε το όνομα κάποιου, για παράδειγμα, το όνομα φαίνεται απλώς να μας έρχεται—το συνειδητό μυαλό δεν έχει ιδέα για το ποια ήταν η διαδικασία αναζήτησης. Ομοίως, μπορεί να βιώσουμε θυμό ή φόβο ή κατάθλιψη χωρίς να ξέρουμε γιατί. Σύμφωνα με ορισμένες θεωρίες ψυχικής διαταραχής, τέτοια συναισθήματα μπορεί να προκύψουν από εξαιρετικά ενοχλητικές σκέψεις ή αναμνήσεις που εμποδίζονται να γίνουν συνειδητές. Στη θεραπεία που βασίζεται σε τέτοιες θεωρίες, ενδείξεις σχετικά με ανησυχητικές ασυνείδητες σκέψεις μπορεί να αναζητηθούν στα όνειρα ή στα γλωσσικά του ασθενούς και ο ασθενής ενθαρρύνεται να μιλήσει μακροχρόνια και ελεύθερα για να βγει οι ιδέες ανοιχτά, όπου μπορούν να αντιμετωπιστούν.

Ορισμένα είδη σοβαρής ψυχολογικής διαταραχής που κάποτε θεωρούνταν καθαρά πνευματική ή ψυχική έχουν βάση σε βιολογικές ανωμαλίες. Η καταστροφή του εγκεφαλικού ιστού από όγκους ή σπασμένα αιμοφόρα αγγεία μπορεί να προκαλέσει μια ποικιλία συμπτωμάτων συμπεριφοράς, ανάλογα με τις θέσεις στον εγκέφαλο που επηρεάζονται. Για παράδειγμα, οι εγκεφαλικές κακώσεις μπορεί να επηρεάσουν την ικανότητα να συνδυάζουν λέξεις με κατανοητό τρόπο ή να κατανοούν την ομιλία των άλλων ή μπορεί να προκαλούν ανούσιες συναισθηματικές εκρήξεις. Η ανεπάρκεια ή η περίσσεια ορισμένων χημικών ουσιών που παράγονται στον εγκέφαλο μπορεί να οδηγήσει σε παραισθήσεις και χρόνια κατάθλιψη.

Η ψυχική επιδείνωση που εμφανίζεται μερικές φορές στους ηλικιωμένους μπορεί να προκαλείται από πραγματική ασθένεια του εγκεφάλου. Η βιολογική ανωμαλία δεν προκαλεί απαραίτητα την ψυχολογική δυσλειτουργία από μόνη της, αλλά μπορεί να κάνει τα άτομα εξαιρετικά ευάλωτα σε άλλες αιτίες διαταραχής.

Αντίθετα, οι έντονες συναισθηματικές καταστάσεις έχουν κάποια διακριτά βιοχημικά αποτελέσματα. Ο φόβος και ο θυμός προκαλούν την απελευθέρωση ορμονών στην κυκλοφορία του αίματος που προετοιμάζουν το σώμα για δρά-

ση - μάχη ή φυγή. Η ψυχολογική δυσφορία μπορεί επίσης να επηρεάσει την ευπάθεια ενός ατόμου σε βιολογικές ασθένειες. Υπάρχουν κάποιες ενδείξεις ότι οι έντονες ή χρόνιες συναισθηματικές καταστάσεις μπορεί μερικές φορές να προκαλέσουν αλλαγές στο νευρικό, στο σπλαχνικό και στο ανοσοποιητικό σύστημα.

Για παράδειγμα, ο φόβος, ο θυμός, η κατάθλιψη ή ακόμα και η απλή απογοήτευση μπορεί να οδηγήσουν στην ανάπτυξη πονοκεφάλων, ελκών και λοιμώξεων. Τέτοιες επιδράσεις μπορούν να κάνουν το άτομο ακόμη πιο ευάλωτο στο ψυχολογικό στρες – δημιουργώντας έναν φαύλο κύκλο δυσλειτουργίας. Από την άλλη πλευρά, υπάρχουν στοιχεία ότι οι κοινωνικές επαφές και η υποστήριξη μπορεί να βελτιώσουν την ικανότητα ενός ατόμου να αντιστέκεται σε ορισμένες ασθένειες ή μπορεί να ελαχιστοποιήσουν τις επιπτώσεις τους.[5]

1.2 Το λογισμικό βελτιώνει την ιατρική

Τα *big data*, η τεχνητή νοημοσύνη, το *cloud computing*, το «*internet of things*» και οι φορητές συσκευές - όλα εμφανίστηκαν τις τελευταίες δύο δεκαετίες. Πράγματα που ήταν αδύνατο να φανταστούμε στο παρελθόν αλλάζουν τώρα τη ροή εργασίας σχεδόν σε κάθε κλάδο, συμπεριλαμβανομένου του ιατρικού τομέα.

Μπορείτε να φανταστείτε να ερμηνεύετε μια ιατρική εικόνα σε χιλιοστά του δευτερολέπτου; Ή τον εντοπισμό του καρκίνου πιο γρήγορα και με μεγαλύτερη ακρίβεια; Η τεχνολογία εξελίσσεται ραγδαία και οι εφαρμογές της συμβάλλουν σημαντικά στην ανάπτυξη της υγειονομικής περίθαλψης. Με αυξανόμενο αριθμό διαθέσιμων λύσεων ιατρικού λογισμικού στην αγορά, προέκυψαν νέες δυνατότητες στον κλάδο της υγειονομικής περίθαλψης.

Υπάρχουν επί του παρόντος πολλά παραδείγματα εφαρμογών τεχνητής νοημοσύνης που βοηθούν στην καταπολέμηση του COVID-19. Με την εφαρμογή του σε λογισμικό ιατρικών συσκευών, οι επαγγελματίες υγείας είναι σε θέση να ανιχνεύσουν, να προβλέψουν και να αντιμετωπίσουν αυτήν την παγκόσμια πρόκληση όχι μόνο πιο γρήγορα, αλλά και με μεγαλύτερη ακρίβεια. Η πανδημία δεν είναι η μόνη κατάσταση όπου αυτές οι τεχνολογίες μπορούν να χρησιμοποιηθούν στο μέγιστο των δυνατοτήτων τους.

Σε αυτή την παράγραφο, θα διερευνήσουμε τις εφαρμογές της τεχνολογίας σε διάφορα προϊόντα ιατρικού λογισμικού, θα ρίξουμε φως σε μερικές από τις πιο καινοτόμες νεοφυείς επιχειρήσεις και εταιρείες σε όλο τον κόσμο και θα δούμε ποιες είναι οι προκλήσεις που αντιμετωπίζει ο κλάδος της υγείας για την ανάπτυξη αυτών των τεχνολογιών.[1]

1.2.1 Δεν υπάρχει χώρος για τεχνοφοβία στην υγεία

Οι πιο πρόσφατες τεχνολογίες όπως η μικτή πραγματικότητα, η μηχανική όραση και η βαθιά μάθηση μπορεί να φαίνονται αρκετά τρομακτικές στους επαγγελματίες του ιατρικού τομέα. Κάποιοι θα μπορούσαν να φοβούνται ότι τα ρομπότ και οι μηχανές θα τα αντικαταστήσουν στο εγγύς μέλλον. Είναι όμως αλήθεια;

Η ανάπτυξη της πληροφορικής στον κλάδο της υγειονομικής περίθαλψης έδωσε τη δυνατότητα στους επαγγελματίες υγείας να κάνουν πολλά πράγματα που πριν ήταν αδύνατα. Για παράδειγμα, η τεχνολογία βοηθά τους γιατρούς να εμπλουτίσουν τις διαγνώσεις τους με αποφάσεις που βασίζονται σε δεδομένα. Τους δίνει επίσης τη δυνατότητα να αναλύουν έναν μεγάλο αριθμό ιατρικών εικόνων ταχύτερα και με μεγαλύτερη ακρίβεια. Σήμερα, οι γιατροί μπορούν να είναι καλύτερα προετοιμασμένοι για χειρουργική επέμβαση και να ελαχιστοποιήσουν τους κινδύνους χάρη στην τεχνολογία και τις νέες λύσεις ιατρικού λογισμικού.

Η ιατρική έχει να κάνει με τους ανθρώπους που βοηθούν άλλα ανθρώπινα όντα να κατανοήσουν και να ξεπεράσουν δύσκολες καταστάσεις, δηλαδή ασθένειες. Το ιατρικό λογισμικό είναι ένα απλό εργαλείο που βοηθά τους επαγγελματίες υγείας να αφιερώνουν περισσότερο ποιοτικό χρόνο στους ασθενείς τους. Δεν μπορούν ούτε να αντικαταστήσουν την ανθρώπινη κρίση ούτε να υποκαταστήσουν την ανθρώπινη λήψη αποφάσεων. Αντί για σενάρια όπου τα ρομπότ αντικαθιστούν τους ανθρώπους, ας διερευνήσουμε πράγματα που η τεχνολογία κατέστησε δυνατά και πραγματικά στον τομέα του ιατρικού λογισμικού στον κλάδο της υγείας.[1]

1.2.2 Λογισμικά Ιατρικής Διάγνωσης

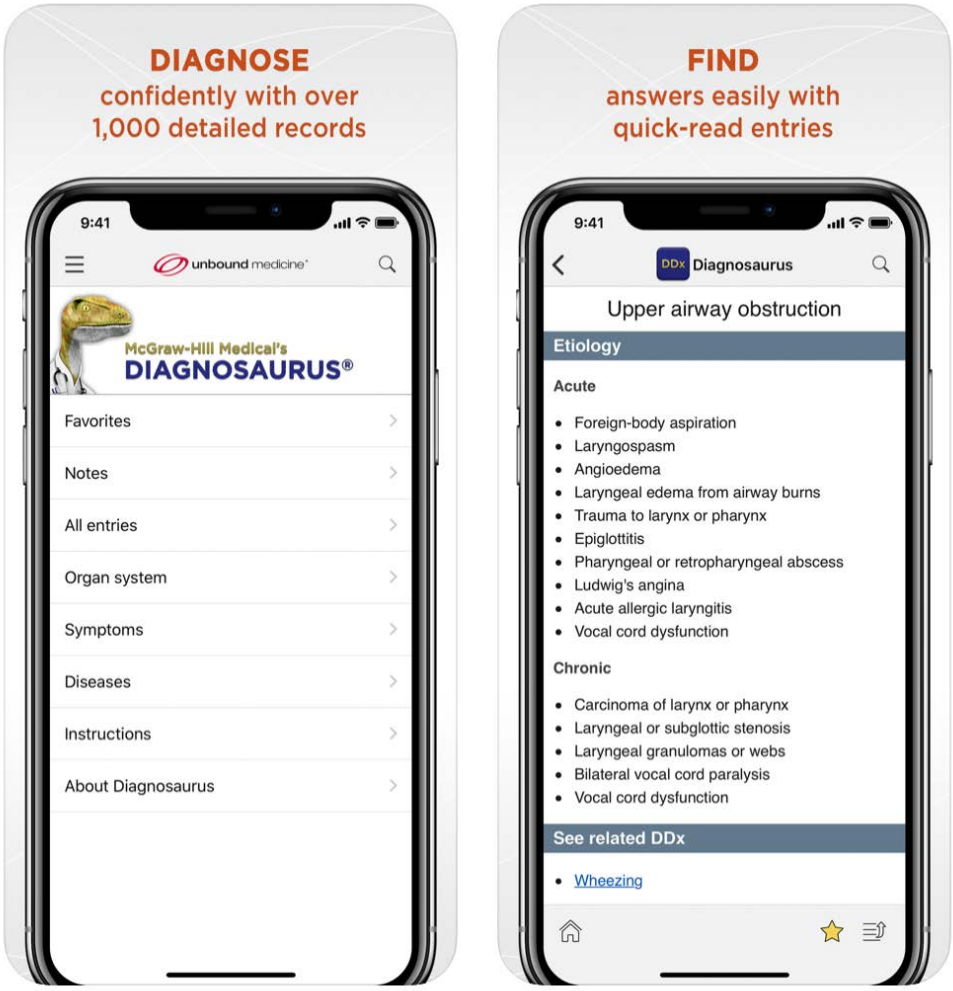
Οι αναδυόμενες λύσεις ιατρικού λογισμικού στον κλάδο της υγειονομικής περίθαλψης παρέχουν ένα μεγάλο φάσμα προηγουμένως αδιανόητων αποτε-

λεσμάτων. Βοηθούν τους επαγγελματίες υγείας να συλλέξουν δεδομένα προκειμένου να λάβουν αποφάσεις που βασίζονται σε δεδομένα, ενώ τους βοηθούν επίσης να ερμηνεύσουν μεγάλο όγκο δεδομένων.

Το ιατρικό λογισμικό επιτρέπει στους επαγγελματίες υγείας να λαμβάνουν τις αποφάσεις τους πιο γρήγορα και με μεγαλύτερη ακρίβεια. Επιτρέπουν επίσης την εμφάνιση νέων δεδομένων και εμπλουτίζουν τις διαδικασίες λήψης αποφάσεων, κάτι που είναι ιδιαίτερα σημαντικό να λαμβάνεται υπόψη όταν πρόκειται για λογισμικό ιατρικής διάγνωσης.

Η πληροφορική δεν είναι κάτι νέο στον κλάδο της υγείας. Το πρώτο βήμα στο οποίο συνέβαλε ήταν να καταστήσει διαθέσιμη τη γνώση προσβάσιμη στο διαδίκτυο. Στη συνέχεια, το έκανε συστηματοποιημένο και προσβάσιμο πιο γρήγορα. Αυτό που ακολούθησε ήταν η προσαρμογή αυτών των βημάτων στις αναδυόμενες τεχνολογίες.

Μπορούμε να το δούμε σε ένα παράδειγμα του Diagnosaurus - μια εφαρμογή υγειονομικής περίθαλψης για κινητά που εξελίχθηκε την τελευταία δεκαετία. Η πρώην λίστα απλής διάγνωσης δίνει πλέον τη δυνατότητα στους γιατρούς να εξερευνούν με ένα πάτημα στο τηλέφωνό τους περισσότερες από 1.000 διαφορικές διαγνώσεις ανά όργανο, σύμπτωμα ή ασθένεια. Ξεχωρίζει επειδή το χαρακτηριστικό του "*See Related DDx*" βοηθά τους γιατρούς να εξετάσουν εναλλακτικές διαγνώσεις.



Εικόνα 3: Στιγμιότυπο από την εφαρμογή *Diagnosaurus*

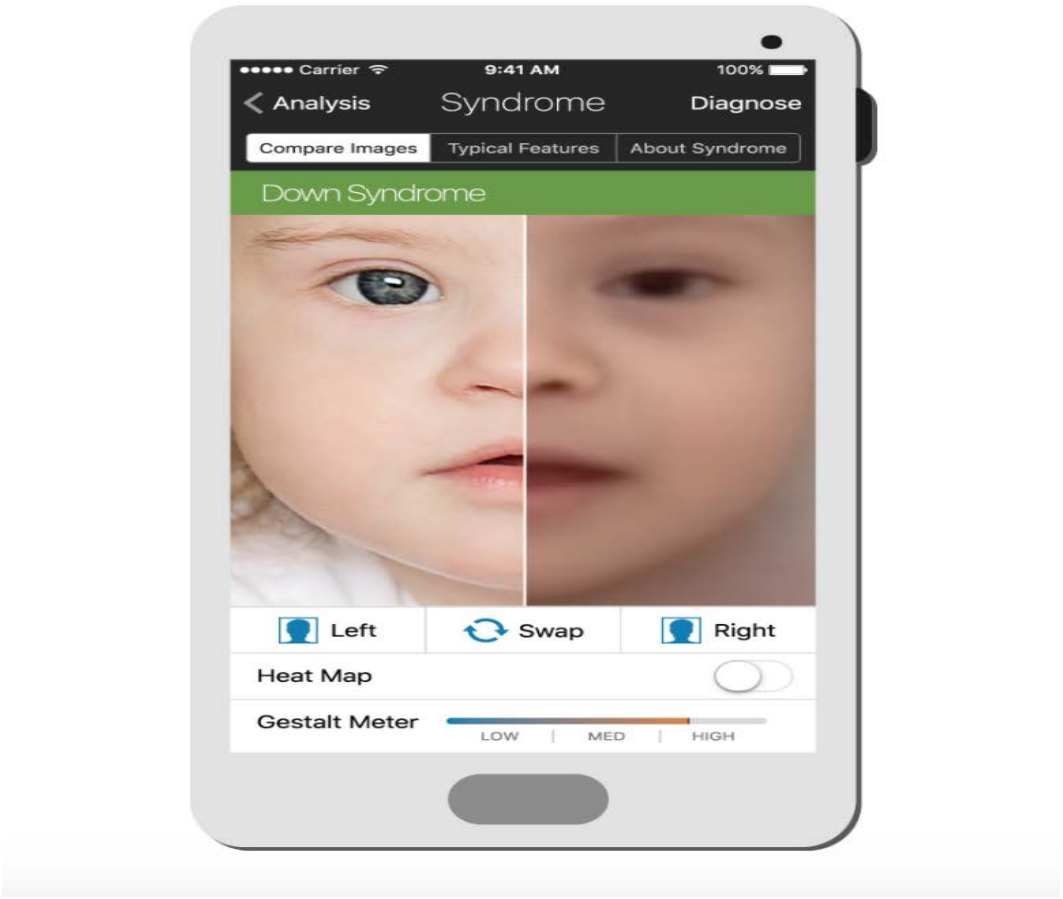
Σήμερα, η πληροφορική προσφέρει καινοτόμες λύσεις. Ένα από τα παραδείγματα για το πώς η τεχνολογία βοηθά τους γιατρούς στα διαγνωστικά προέρχεται από την εταιρεία FDNA, που ειδικεύεται στον προσδιορισμό φαινοτύπων επόμενης γενιάς. Ιδρύθηκε το 2011, η εταιρεία δημιούργησε το Face2Gene - ένα λογισμικό αναγνώρισης προσώπου σε συνδυασμό με μηχανική εκμάθηση.

Το Face2Gene είναι «ένα εργαλείο αναζήτησης και αναφοράς για σπάνιες ασθένειες και επιτρέπεται η χρήση μόνο από επαγγελματίες υγείας. Βοηθά τους κλινικούς γιατρούς να φτάσουν πιο γρήγορα σε πιθανές διαφορικές διαγνώσεις και βοηθά τα εργαστήρια να συσχετίσουν τα αποτελέσματα της αλληλουχίας τους με τον φαινότυπο του ασθενούς», *Η τεχνολογία AI τους επί του παρόντος «επιτρέπει την ανάλυση φωτογραφιών προσώπου και την ανάλυση*

κλινικών σημειώσεων (κειμένου) [...] να βοηθήσουν τα εργαστήρια να συσχετίσουν τα αποτελέσματα της αλληλουχίας τους με τον φαινότυπο του ασθενούς.

(Nicole Fleischer, Αντιπρόεδρος Επιστημονικής Έρευνας του FDNA.)

Επί του παρόντος, υπάρχουν περισσότεροι από 400 εκατομμύρια ασθενείς παγκοσμίως που πάσχουν από σπάνιες ασθένειες, εκ των οποίων οι μισοί υπολογίζεται ότι είναι παιδιά. Οι επαγγελματίες υγείας χρησιμοποιούν την εφαρμογή Face2Gene για πολλούς από αυτούς. Η προστιθέμενη αξία της τεχνολογίας του Face2Gene έχει αποδειχθεί κατά τη διάρκεια της πανδημίας COVID-19. *Η τεχνητή νοημοσύνη του FDNA παρέχει στα εργαστήρια λεπτομερή φαινότυπο ασθενών και «επιτρέπει στους κλινικούς γιατρούς που εργάζονται με την τηλεϊατρική, να λαμβάνουν με ασφάλεια πληροφορίες ασθενών πριν από τη συνεδρία τηλευγείας, επιτρέποντας έτσι μια πιο εστιασμένη και ακριβή συνάντηση*, (Nicole Fleischer, Αντιπρόεδρος Επιστημονικής Έρευνας του FDNA.)

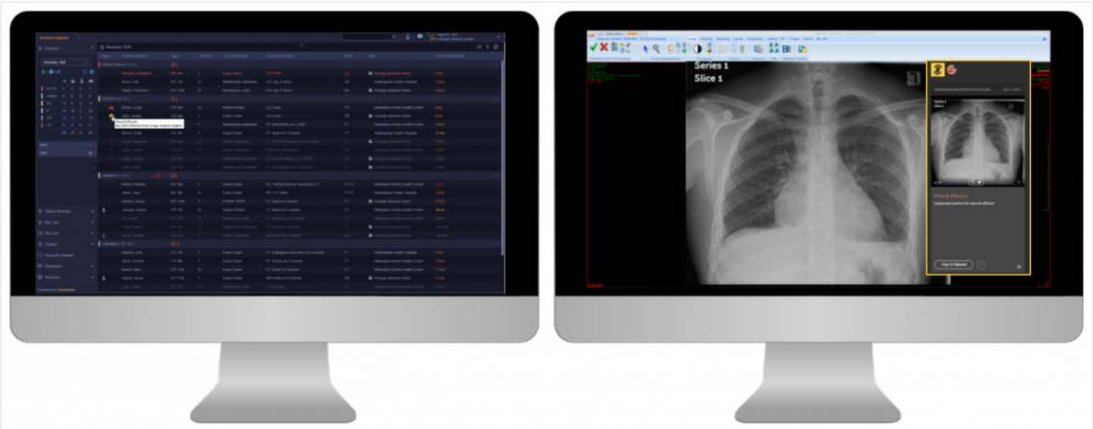


Εικόνα 4: Στιγμιότυπο από την εφαρμογή Face2Gene

Επειδή η MedTech καθιστά δυνατή την παραγωγή και τη διαχείριση μεγάλων ποσοτήτων δεδομένων, προέκυψαν πολλές λύσεις για να βοηθήσουν τους γιατρούς να εντοπίζουν πιο γρήγορα χρήσιμες πληροφορίες από σύνολα δε-

δομένων για τον καθορισμό διαγνώσεων. Ειδικά στον τομέα της ακτινολογίας, όπου η ανάλυση πολλών απεικονιστικών σαρώσεων μπορεί να είναι πολύ δύσκολη, προκειμένου να αποφευχθεί το λάθος, μεγιστοποιώντας παράλληλα την ακρίβεια. Κάποιο ιατρικό λογισμικό έχει κατασκευαστεί για να κάνει ακριβώς αυτό - να αναλύει τις ιατρικές εικόνες των ασθενών με μεγαλύτερη ταχύτητα και ακρίβεια.

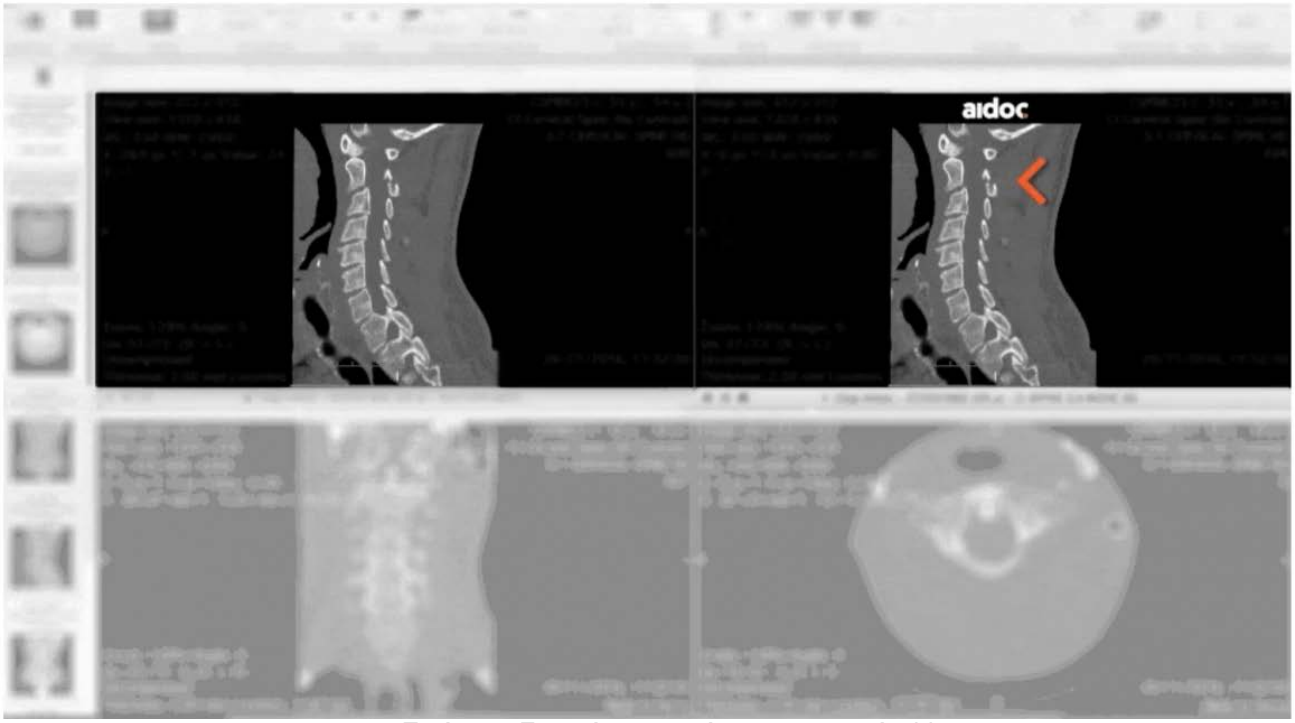
Αυτός είναι ο λόγος για τον οποίο η Zebra Medical Vision βρήκε μια λύση AI1 "όλα σε ένα" για ακτινολόγους. Το Imaging Analytics Engine τους λαμβάνει σαρώσεις απεικόνισης και τις αναλύει αυτόματα για διάφορα κλινικά ευρήματα που έχει μελετήσει. Η τεχνολογία του βασίζεται σε μια «βάση δεδομένων εκατομμυρίων σαρώσεων απεικόνισης, μαζί με μηχανήματα και εργαλεία βαθιάς μάθησης» που τους βοήθησε να «δημιουργήσουν λογισμικό που αναλύει δεδομένα σε πραγματικό χρόνο με ακρίβεια σε ανθρώπινο επίπεδο - παρέχοντας στους ακτινολόγους τη βοήθεια που χρειάζονται για να διαχειριστείτε συνεχώς αυξανόμενους φόρτους εργασίας, χωρίς να θυσιάσετε την ποιότητα», σύμφωνα με την ιστοσελίδα της εταιρείας.



Εικόνα 5: Στιγμιότυπο από την εφαρμογή Zebra AI1

Πολλές νεοφυείς επιχειρήσεις εμφανίστηκαν με στόχο να κάνουν μια πιο ακριβή διάγνωση εφαρμόζοντας τεχνολογία AI. Ένα από αυτά είναι η Aidoc, μια ισραηλινή start-up που χρησιμοποιεί τεχνητή νοημοσύνη για να αναλύσει ιατρικές σαρώσεις κεφαλής, σπονδυλικής στήλης, κοιλιάς και θώρακα. Η εταιρεία δηλώνει πώς τα προϊόντα της «βοηθούν στον εντοπισμό και τον εντοπισμό κρίσιμων ανωμαλιών για τους ακτινολόγους μέσω αλγορίθμων βαθιάς μάθησης και τεχνητής νοημοσύνης που αναλύουν ιατρικές εικόνες και δεδομένα

ασθενών». Ένα από τα μεγαλύτερα πλεονεκτήματά τους είναι ένα εργαλείο προτεραιότητας που επισημαίνει κρίσιμα ευρήματα, όπως η ενδοκρανιακή αιμορραγία σε αξονικές τομογραφίες κεφαλής χωρίς σκιαγραφικό. Έτσι, οι ακτινολόγοι μπορούν να προσαρμόσουν τη λίστα εργασιών τους εάν είναι απαραίτητο.



Εικόνα 6: Στιγμιότυπο από την εφαρμογή Aidoc

Priority	STAT	Patient Name	Patient MRN	Modality	Study Time	Hospital Location	Procedure
a	STAT	Rick, Johnson	81945487	CT	7:55am (74 min ago)	Inpatient	Cervical-spine
	STAT	Rice, Michelle	12859436	CT	6:39am (2 hr ago)	ED	Head
a		Lopez, Luis	28427001	CT	8:46 (23 min ago)	Inpatient	Abdomen
a		Patton, Simon	85340275	CT	8:57 (12 min ago)	Outpatient	Chest
		Garcia, Howard	81945487	CT	8:28am (41 min ago)	Inpatient	Cervical-spine
		West, Sandy	54132180	MR	8:39 (30 min ago)	Outpatient	Cervical-spine

Εικόνα 7: Στιγμιότυπο από την εργαλείο προτεραιοποίησης της εφαρμογής Aidoc

Η απεικόνιση παίζει καθοριστικό ρόλο στη διάγνωση και την τεκμηρίωση ασθενειών όπως ο καρκίνος. Μια μελέτη που δημοσιεύθηκε από την NVIDIA έδειξε ότι η βαθιά μάθηση μειώνει το ποσοστό σφαλμάτων για διαγνώσεις καρκίνου του μαστού κατά 85%. Αυτός ήταν ένας από τους λόγους για τους οποίους οι Jeet Samarth Raut, Peter Wakahiu Njenga και Simon Rasalingham ίδρυσαν την ιατρική πλατφόρμα απεικόνισης AI Behold.ai το 2016. Βρήκαν έναν αλγόριθμο red dot® που βασίζεται σε μοντέλα βαθιάς μάθησης.

Αυτό σημαίνει ότι μπορεί να ταξινομήσει μια ακτινογραφία θώρακος και να εντοπίσει τα ευρήματά της ως χάρτες θερμότητας. Σύμφωνα με τον ιστότοπο της εταιρείας, ο αλγόριθμός τους έχει αναπτυχθεί «χρησιμοποιώντας περισσότερες από 30 χιλιάδες παραδείγματα εικόνων, οι οποίες όλες έχουν ελεγχθεί και αναφερθεί από έμπειρους συμβούλους ακτινολόγους κλινικούς γιατρούς». Ως αποτέλεσμα, ο αλγόριθμός τους έχει πάνω από 90% ακρίβεια για την ανίχνευση ανωμαλιών μέσα σε δευτερόλεπτα.[1]

1.2.3 Λογισμικά Ιατρικής Οπτικοποίησης

Οι νέες τεχνολογίες που σχετίζονται με τρισδιάστατα μοντέλα, εικονική πραγματικότητα ή προσομοιώσεις έδωσαν τη δυνατότητα σε γιατρούς και χειρουργούς να αποκτήσουν νέες πληροφορίες που προηγουμένως δεν μπορούσαν να κάνουν. Τέτοιες τεχνολογίες δίνουν τη δυνατότητα στους επαγγελματίες υγείας να εκτελούν καλύτερα τη δουλειά τους με λύσεις ιατρικού λογισμικού για οπτικοποίηση ή προετοιμασία χειρουργικής επέμβασης.

Στον τομέα των συσκευών οπτικοποίησης, μια από τις γνωστές εταιρείες είναι η AccuVein. Παρήγαγαν μια συσκευή οπτικοποίησης φλεβών που βοηθά τους επαγγελματίες υγείας να βρίσκουν εύκολα τις φλέβες των ασθενών για αιμοληψίες. Η εταιρεία δηλώνει ότι η φορητή συσκευή επαυξημένης πραγματικότητας χρησιμοποιεί «ένα σύστημα τρισδιάστατης απεικόνισης εγγύς υπέρυθρο (NIR) για την ανίχνευση φλεβών. Καθώς η συσκευή εστιάζει στην περιοχή στόχο του ασθενούς, το πρώτο IR ανιχνεύει την αιμοσφαιρίνη στις φλέβες μέσω του δέρματος του ασθενούς» και την προβάλλει ως εικόνα αντίθεσης. «Καθώς η ένταση του πρώτου λέιζερ αυξάνεται και πηγαίνει βαθύτερα στο δέρμα, ο φωτοανιχνευτής λαμβάνει αρκετές εικόνες φλεβών με βάση το βάθος. ”

Μια άλλη λύση έρχεται από την εταιρεία EchoPixel. Ανυπομονούσαν να βρουν μια λύση ώστε οι χειρουργοί να εξασκηθούν και να προετοιμαστούν για επεμβάσεις με τρισδιάστατα μοντέλα, αντί για εικόνες 2D. Το λογισμικό True 3D καθιστά δυνατή την αλληλεπίδραση με τρισδιάστατες εικόνες μιας «εικονικής ανατομίας ειδικά για τον ασθενή χωρίς την ανάγκη για ογκώδη ακουστικά VR/AR». Απαιτούνται ειδικά γυαλιά για προβολή και γραφίδα για τον χειρισμό των εικόνων. Σύμφωνα με την εταιρεία, το λογισμικό χρησιμοποιεί μηχανική εκμάθηση για να καταγράψει τον τρόπο με τον οποίο οι γιατροί αλληλοεπιδρούν με την παραγόμενη τρισδιάστατη εικόνα. Κατά συνέπεια, άλλοι γιατροί μπορούν να αλληλοεπιδράσουν με την εικόνα με τον ίδιο τρόπο που είχαν οι προηγούμενοι γιατροί.

Η τεχνολογία τρισδιάστατης εκτύπωσης είναι επίσης το θεμέλιο για τη βελγική εταιρεία Materialise. Η σουίτα Mimics Innovation σχεδιάστηκε για μηχανικούς και ερευνητές «για να κάνει τη χρήση δεδομένων ιατρικών εικόνων για μηχανικούς σκοπούς (3D ανάλυση, σχεδιασμός και εξατομίκευση συσκευών) όσο το δυνατόν πιο εύκολη και ικανοποιητική». Κατασκεύασαν επίσης τη σουίτα Materialize Mimics Care για επαγγελματίες υγείας για σχεδιασμό εικόνας και ιατρική τρισδιάστατη εκτύπωση σε νοσοκομεία.[2]

1.2.4 Λογισμικά Θεραπευτικού Σκοπού

Το λογισμικό μπορεί να έχει πολλές εφαρμογές στον τομέα της υγείας και πολλές εταιρείες και νεοφυείς επιχειρήσεις εργάζονται για την ανάπτυξη νέων θεραπευτικών εργαλείων για τους γιατρούς. Ποια είναι η κατάσταση αυτού του λογισμικού όταν συνοδεύεται από ιατρικές συσκευές, κάτι που συμβαίνει συχνά στον τομέα της αποκατάστασης; Πρέπει να δηλωθεί ως λογισμικό ως ιατρική συσκευή; Είναι λογισμικό που αποτελεί μέρος μιας ιατρικής συσκευής; Ή είναι λογισμικό ως μη ιατρική υπηρεσία; Το ερώτημα εξακολουθεί να εκκρεμεί σε πολλές χώρες που δεν σταματούν να κυκλοφορούν λύσεις λογισμικού στην αγορά.

Ένα από τα παραδείγματα προγραμμάτων λογισμικού που χρησιμοποιούνται για την αποκατάσταση γνωστικών διαταραχών προέρχεται από τη γερμανική εταιρεία HASOMED GmbH. Βρήκαν το RehaCom - ένα «σύστημα λογισμικού για τη γνωστική αποκατάσταση με τη βοήθεια υπολογιστή». Σύμφωνα με την εταιρεία, το σύστημα παρέχει «εννέα ενότητες προσυμπτωματικού ελέγ-

χου για να υποστηρίξει τον θεραπευτή να επιλέξει τις πιο αποτελεσματικές ενότητες θεραπείας». Το RehaCom αποτελείται από 28 ενότητες παιχνιδιοποιημένης θεραπείας για να «βοηθήσει τους ασθενείς να βελτιώσουν τη γνωστική λειτουργία και τις αντισταθμιστικές δεξιότητες στην προσοχή, τη μνήμη, τις εκτελεστικές λειτουργίες και το οπτικό πεδίο. Η εταιρεία τονίζει ότι η Rehacom έχει χρησιμοποιηθεί στο 95% των γερμανικών κλινικών αποκατάστασης. Παρέχει επίσης «μια λύση εκτός σύνδεσης υπολογιστή για εσωτερικούς ασθενείς καθώς και μια λύση τηλεαποθεραπείας μέσω Διαδικτύου για εξωτερικούς ασθενείς».

Εικόνα 8: Στιγμιότυπο του λογισμικού RehaCom

Η εμφάνιση των εννοιών του βιντεοπαιχνιδιού (ή του *gamification*) στον κλάδο της υγειονομικής περίθαλψης είναι μια προσπάθεια βελτίωσης των κλινικών αποτελεσμάτων των ασθενών με ελκυστικά μέσα. Μια τέτοια τάση έχει επίσης θέσει την «*ανάγκη για έναν «ψηφιακό επαγγελματία» που διοχετεύει αυτά τα παιχνίδια, παρακολουθεί την πρόοδο και επιλέγει τα καταλληλότερα για έναν δεδομένο ασθενή*», (Eli G. Phillips Jr, Chadi Nabhan, και Bruce A. Feinberg).

Η τάση του gamification είναι ιδιαίτερα σημαντική όταν πρόκειται για θεραπευτικές λύσεις για παιδιά.

Για να τους κρατήσει όσο το δυνατόν περισσότερο αφοσιωμένους και να αυξήσει τα κλινικά τους αποτελέσματα, η γερμανική εταιρεία Caterna ανέπτυξε το λογισμικό Caterna Vision Therapy για παιδιά με αμβλυωπία ή «τεμπέλικο μάτι». Τα σενάρια εννέα παιχνιδιών τους είναι προσβάσιμα τόσο στο διαδίκτυο όσο και ως εφαρμογή για κινητά τηλέφωνα για παιδιά ηλικίας 4-12 ετών. Σύμφωνα με την εταιρεία, «ένα ειδικό μοτίβο κυμάτων στο παρασκήνιο των παιχνι-

διών διεγείρει τον εγκέφαλο να ενεργοποιήσει ξανά το αδύναμο μάτι» και είναι αποτελεσματικό μετά από 90 ημέρες θεραπείας. Είναι επίσης η πρώτη ευρωπαϊκή θεραπεία που βασίζεται στο Διαδίκτυο που καλύπτεται από ασφάλιση υγείας ως ιατρική συσκευή. Έτσι, μόνο ένας οφθαλμίατρος μπορεί να το συνταγογραφήσει ως μέρος της θεραπείας απόφραξης.[2]

Εικόνα 9: Στιγμιότυπο από το λογισμικό Caterna Vision

1.2.5 Λογισμικά παρακολούθησης ασθενούς

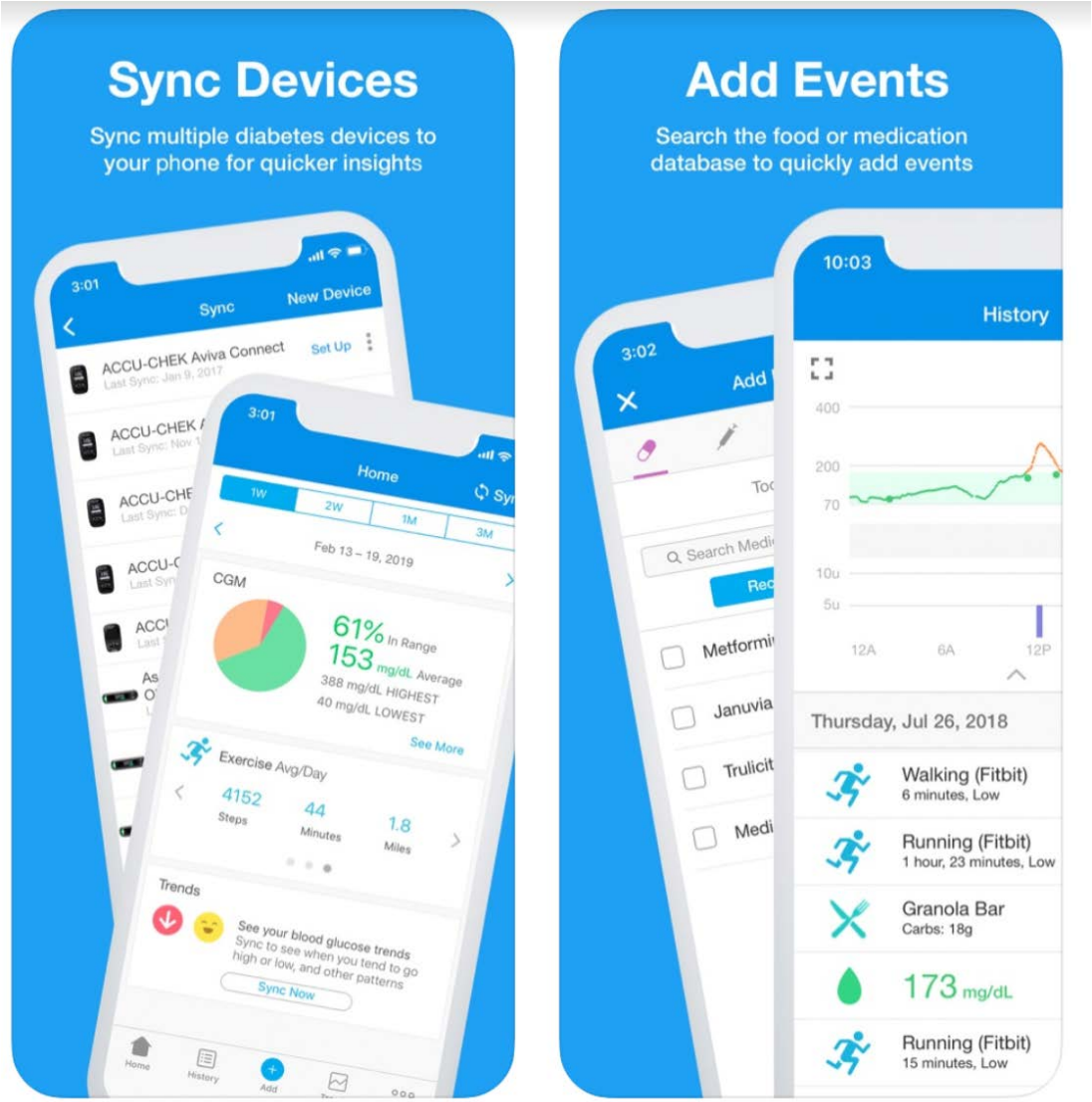
Η ανάπτυξη της τεχνολογίας στον κλάδο της υγειονομικής περίθαλψης είχε ως αποτέλεσμα έναν αυξανόμενο αριθμό κινητής τεχνολογίας υγείας ή αγοράς mHealth. Το 2019, η παγκόσμια αγορά mHealth υπολογίστηκε σε 37 δισεκατομμύρια δολάρια. Με αυτό, εμφανίστηκαν επίσης φορητές συσκευές, δίνοντας τη δυνατότητα στους επαγγελματίες υγείας και τους ασθενείς τους να διατηρούν επαφή και να αλληλοεπιδρούν εξ αποστάσεως.

Για παράδειγμα, έχουν γίνει πολλά στον τομέα των διαβητικών που χρησιμοποιεί ως επί το πλείστων συστήματα διαχείρισης δεδομένων. Σε αυτήν την περίπτωση, τα συστήματα διαχείρισης δεδομένων είναι απλά προγράμματα που σας επιτρέπουν αυτόματα ή μη αυτόματα να καταγράφετε δεδομένα όπως τα επίπεδα γλυκόζης στο αίμα. Αυτά τα δεδομένα μπορούν στη συνέχεια να εμφανιστούν ως γράφημα για τον προσδιορισμό των τάσεων ή ακόμη και να σταλούν ως ειδοποιήσεις σε έναν επαγγελματία υγείας για να βοηθήσουν τους ασθενείς να διαχειριστούν τον διαβήτη τους.

Μία από τις λύσεις λογισμικού παρακολούθησης προέρχεται από την αμερικανική εταιρεία Glooko. Ιδρύθηκε το 2010 για να «μεταμορφώσει τον τρόπο με τον οποίο λαμβάνονται οι καθημερινές αποφάσεις αξιοποιώντας τη δύναμη του κινητού, του cloud και των αναλυτικών στοιχείων» για άτομα με διαβήτη.

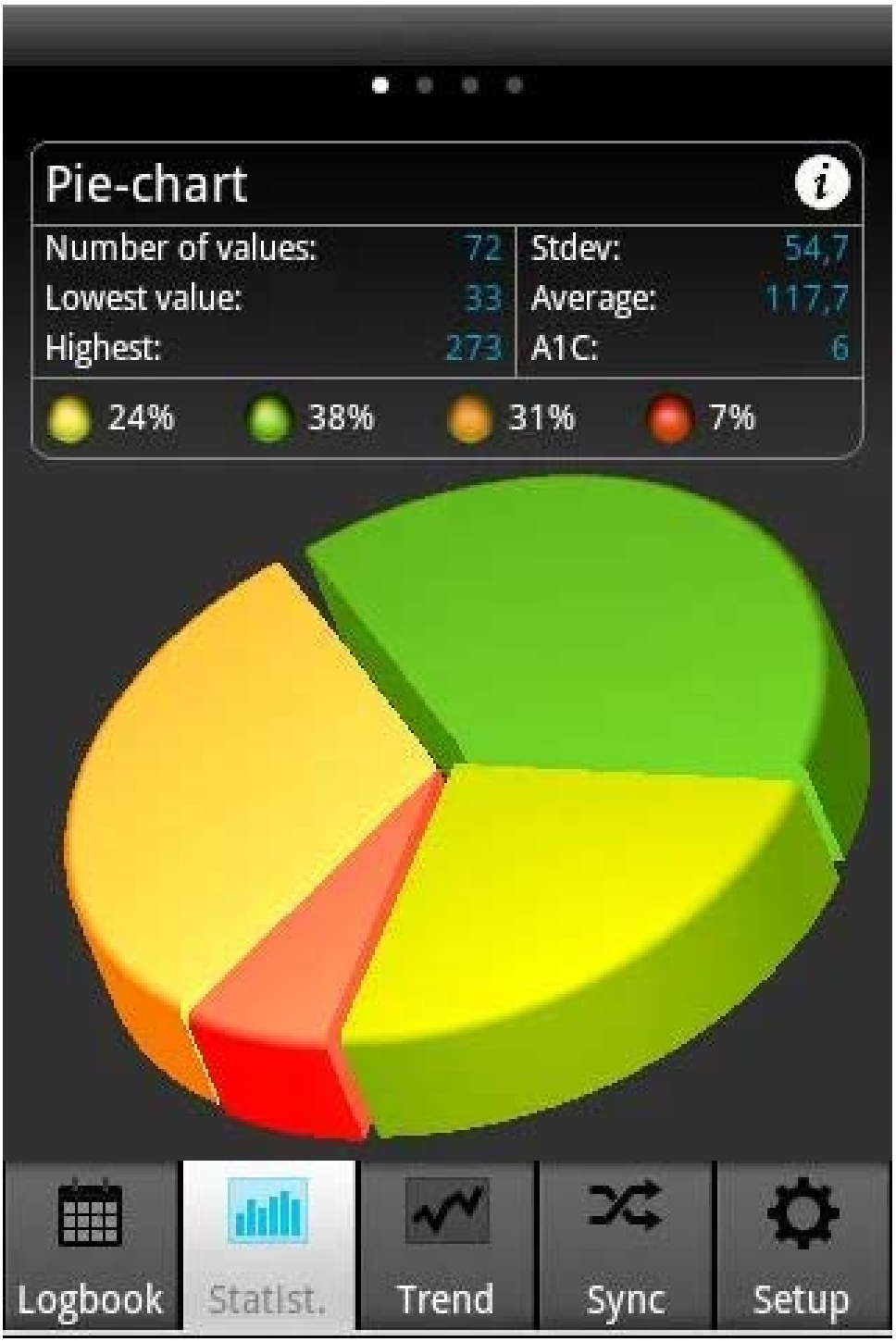
Σήμερα, οι πλατφόρμες λογισμικού της εταιρείας, Glooko® (στη Βόρεια Αμερική) και diasend® (στην EMEA/APAC), ενισχύουν τη διαχείριση του διαβήτη συλλέγοντας δεδομένα από μετρητές γλυκόζης αίματος, CGM, αντλίες ινσουλίνης, στυλό και ανιχνευτές δραστηριότητας. Τα δεδομένα μεταφορτώνονται εύκολα - απομακρυσμένα μέσω εφαρμογής ή εντός κλινικής, μοιράζονται με ασφάλεια και οπτικοποιούνται σε γραφήματα και γραφήματα με δυνατότητα δράσης.

Οι πλατφόρμες είναι συμβατές με τη συντριπτική πλειονότητα των διαθέσιμων συσκευών διαβήτη, δίνοντας στα άτομα με διαβήτη και στις ομάδες φροντίδας τους την ελευθερία επιλογής. Κατά τη διάρκεια της πανδημίας, η Glooko προσφέρει μια δωρεάν Λύση Απομακρυσμένης Φροντίδας σε άτομα με διαβήτη και παρόχους υγειονομικής περίθαλψης για να επιτρέψει την ασφαλή και συνδεδεμένη φροντίδα ασθενών.



Εικόνα 10: Στιγμιότυπο από την εφαρμογή Glooko

Το λογισμικό διαχείρισης διαβήτη SiDiary προέρχεται από τη γερμανική εταιρεία Sinovo. Παρόμοια με το Glooko, επιτρέπει επίσης στους ασθενείς να εισάγουν τα δεδομένα τους από μετρητές και αντλίες στην έκδοση υπολογιστή στην οποία μπορούν να έχουν πρόσβαση και να αναλύουν από κινητά τηλέφωνα και tablet. Οι ασθενείς μπορούν επίσης να στείλουν τα στοιχεία τους στους γιατρούς τους. Από την πλευρά τους, οι γιατροί μπορούν να χρησιμοποιήσουν την SiDiary Healthcare Professional Version, η οποία τους επιτρέπει να διαβάζουν τα δεδομένα των ασθενών, να λαμβάνουν πιο λεπτομερή δεδομένα για τους ασθενείς τους και να τους παρέχουν πιο προσεκτική παρακολούθηση, χωρίς να εγκαταστήσουν διαφορετικό λογισμικό από διαφορετικούς κατασκευαστές.

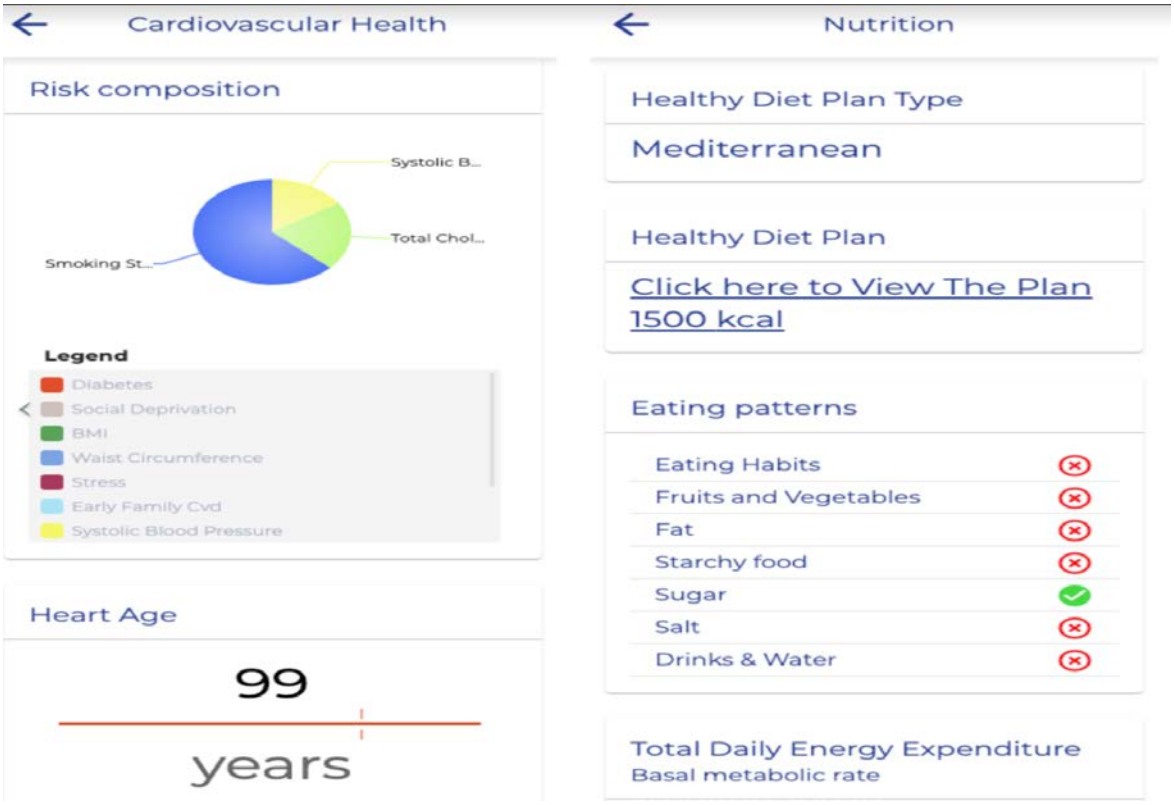


Εικόνα 11: Στιγμιότυπο από την εφαρμογή SiDiary

Παρόμοιες λύσεις λογισμικού εμφανίζονται σε όλους τους άλλους ιατρικούς τομείς εκτός από τον διαβήτη. Ένα από τα παραδείγματα προέρχεται από μια νεαρή κροατοαμερικανική start-up Luxheal. Η ομάδα των γιατρών τους ανέπτυξε το Luana, μια εφαρμογή για κινητά τηλέφωνα που λειτουργεί ως ψηφιακός βοηθός για την πρόληψη και την αποκατάσταση καρδιοπαθών. «Συλλέγει

και αναλύει τα δεδομένα του τρόπου ζωής τους, τους συνδέει με νοσοκομεία και γιατρούς που μπορούν να βελτιστοποιήσουν τη θεραπεία τους με βάση τα παρεχόμενα δεδομένα». Ο Διευθύνων Σύμβουλος της Luxeal, Marino Sabijan, τονίζει πώς η Luana δηλώνεται επί του παρόντος ως «βοήθεια για ασθενείς».

Τα προϊόντα της Luxheal είναι συμβατά με το GDPR και βρίσκονται σε διαδικασία συμμόρφωσης με το HIPAA. Αλλά είναι μόνο το πρώτο βήμα για αυτή τη φιλόδοξη εταιρεία. Περιμένουν τον Κανονισμό Ιατρικών Συσκευών, που έχει προγραμματιστεί για τα τέλη Μαΐου 2020, για να ταξινομηθεί η Luana ως «Λογισμικό ως Ιατρικό Συσκευή». Η εφαρμογή αυτή τη στιγμή χρησιμοποιείται στις ΗΠΑ (Maryland Urgent Care νοσοκομείο) και στην Κροατία.[3]



Εικόνα 12: Στιγμιότυπο από την εφαρμογή Luxheal Luana

1.2.6 Οι νέες ευκαιρίες που προσφέρουν οι λύσεις ιατρικού λογισμικού

Ο κατάλογος των λύσεων ιατρικού λογισμικού που αλλάζουν τον κλάδο της υγειονομικής περίθαλψης θα μπορούσε να συνεχιστεί. Μερικές από τις νέες ευκαιρίες που προσφέρουν στους επαγγελματίες υγείας περιλαμβάνουν την αυτοματοποίηση εργασιών, την ανάλυση συνόλων μεγάλων δεδομένων, τη μείωση του φόρτου εργασίας και τη δυνατότητα να επικεντρωθούν περισσότε-

ρο στους ασθενείς. Ορισμένες εταιρείες λογισμικού ισχυρίζονται ότι το προϊόν τους μπορεί να εξοικονομήσει χρήματα στα συστήματα υγειονομικής περίθαλψης.

Η εμφάνιση της τεχνολογίας υγειονομικής περίθαλψης που μπορεί να φορεθεί και του mHealth συμβάλλει επίσης στην επίτευξη αυτού του στόχου. Οι γιατροί μπορούν πλέον να λαμβάνουν εύκολα δεδομένα παρακολούθησης ασθενών από τέτοιες συσκευές. Βοηθά τους επαγγελματίες υγείας να βελτιστοποιήσουν τον χρόνο εργασίας τους και να αφιερώνουν περισσότερο χρόνο σε ασθενείς υψηλού κινδύνου και να αλληλοεπιδρούν με αφοσιωμένους και ενημερωμένους ασθενείς. Η εμφάνιση πυλών ασθενών επιτρέπει επίσης στους γιατρούς να συνδέονται με τους ασθενείς τους και να εξηγούν τις διαδικασίες πιο διεξοδικά.

Ωστόσο, δεν πρέπει να ξεχνάμε ότι όλες οι τεχνολογικές εξελίξεις βασίζονται σε δεδομένα που παρέχονται από τον άνθρωπο. *Αυτό σημαίνει ότι υπάρχει «κίνδυνος τα σύνολα δεδομένων να περιέχουν ασυνείδητη προκατάληψη». Απομένει να δούμε αν και πώς οι νέοι κανόνες δεοντολογίας θα αντιμετωπίσουν αυτά τα ζητήματα*. (Business Insider).[3]

1.2.7 Ταχέως αναδυόμενη τεχνολογία έναντι αργών κανονισμών

Η τεχνολογία αναπτύσσεται με απίστευτο ρυθμό, αλλάζοντας τον ιατρικό τομέα με απίστευτους τρόπους. Αλλά για να συνεχίσει, η νομοθεσία αγωνίζεται να συμβαδίσει με αυτή την ταχύτητα. Παρόλο που υπάρχουν ήδη νόμοι σχετικά με την προστασία της ιδιωτικής ζωής, όπως η HIPAA στις ΗΠΑ και ο GDPR στην ΕΕ, μπορούμε να δούμε στο παράδειγμα της Luana της Luxheal - οι κανονισμοί διαφέρουν μεταξύ ηπείρων, ακόμη και μεταξύ ευρωπαϊκών χωρών.

Το GDPR και το HIPAA μπορούν επίσης να προκαλέσουν πολλά προβλήματα στους επιχειρηματίες και τους μηχανικούς επιβραδύνοντας τη διαδικασία διάθεσης λύσεων λογισμικού στην αγορά. Ωστόσο, το απόρρητο δεν είναι το μόνο ζήτημα που πρέπει να επιλυθεί όσον αφορά τη ρύθμιση του ιατρικού λογισμικού. Υπάρχουν δύο μεγάλα ερωτήματα που πρέπει να απαντηθούν - πώς πρέπει να ορίσουμε και πώς πρέπει οι κυβερνήσεις να ρυθμίζουν τις νέες ιατρικές τεχνολογίες;

Στην ΕΕ, φαίνεται ότι μένει να δούμε πώς θα απαντηθούν αυτά τα ερωτήματα. Λόγω της πανδημίας του COVID-19, η εφαρμογή του Κανονισμού για τα

ιατροτεχνολογικά προϊόντα αναβλήθηκε για ένα χρόνο. Πώς θα αντιδράσει η αγορά ιατρικού λογισμικού σε αυτό; Ένα πράγμα είναι σίγουρο - θα είναι ενδιαφέρον να δούμε πώς η τεχνολογία βοήθησε στην αντιμετώπιση και (ελπίζουμε) να ξεπεράσουμε το ξέσπασμα του COVID-19.[3]

ΚΕΦΑΛΑΙΟ 2: Ανάλυση και σχεδιασμός εφαρμογής

Σε αυτό το κεφάλαιο θα γίνει αναλυτική προσέγγιση για τον σχεδιασμό της εφαρμογής και οι πρακτικές που εφαρμόστηκαν.

2.1 Λεκτική Περιγραφή

Η εφαρμογή που υλοποιήθηκε για την πτυχιακή εργασία χρησιμοποιεί τη γλώσσα προγραμματισμού Java η οποία είναι μια γλώσσα αντικειμενοστραφή προγραμματισμού. Στις αντικειμενοστραφής γλώσσες έχουμε τη δυνατότητα να κατασκευάσουμε κλάσεις και να δίνουμε έννοια σε αυτές περιγράφοντας τις οντότητες που θα απαρτίζεται η εφαρμογή.

Όπως προαναφέρθηκε στο πρώτο κεφάλαιο, η εφαρμογή θα χρησιμοποιείται από κάποιους δυνητικούς ασθενείς (χρήστες) οι οποίοι θα μπορούν να έχουν εγκατεστημένη την εφαρμογή στον υπολογιστή τους. Η υλοποίηση της εφαρμογής βασίζεται σε μία desktop προσέγγιση. Οι desktop εφαρμογές επικρατούσαν στο παρελθόν πριν έρθουν στο προσκήνιου οι web-based cloud εφαρμογές. Ωστόσο με τη χρήση σύγχρονων μέσων και με την χρήση της Java μπορούμε να αναπτύξουμε ενδιαφέρον λογισμικά που μπορούν να μας λύσουν τα χέρια ακόμη και αν δεν ανήκουν στην κατηγορία των τάσεων.

Παρόλαυτα, όταν γίνεται ανάπτυξη λογισμικού, εκτός από τις σύγχρονες τάσεις που μας ελκύουν να γράψουμε σύγχρονο κώδικα, μας ενδιαφέρει ακόμα περισσότερο να γίνει σωστή σχεδίαση των εξαρτημάτων του κώδικα, η διασύνδεση μεταξύ τους, η χρήση ορισμένων προτύπων (design patterns). Ο σκοπός αυτής της παραγράφου είναι να γίνει μια περιγραφή στις τεχνικές που αξιοποιήθηκαν:

- Dependency Injection (έγχυση εξάρτησης)
    - Το dependency injection είναι ένα πρότυπο στην ανάπτυξη λογισμικού που μας επιτρέπει να μεταβιβάσουμε το αντικείμενο μίας κλάσης σε μία άλλη δημιουργώντας έτσι με αυτό τον τρόπο μία “σύμβαση” μεταξύ των κλάσεων. Αυτό μπορεί να πραγματοποιηθεί μέσω του κατασκευαστή (Constructor).

    - Ας το κάνουμε λίγο πιο σαφές. Αν υποθέσουμε ότι έχουμε κατασκευάσει μία κλάση **A**, η οποία έχει έναν συγκεκριμένο ρόλο, δη-

λαδή να πραγματοποιήσει μία σύνδεση στο ίντερνετ και να κατεβάσει δεδομένα ή να εκτελέσει μία ρουτίνα εγγραφής δεδομένων στον δίσκο (I/O). Η κλάση **A** όπως φαίνεται πραγματοποιεί ένα βαρύ φορτίο, ωστόσο εμείς θέλουμε τα αποτελέσματα αυτής να τα μεταβιβάσουμε σε μία κλάση **B** μέσω το αντικειμένου της κλάσης **A**. Το αντικείμενο της κλάσης **A** θα περάσει ως παράμετρος στον κατασκευαστή της κλάσης **B** και αντίστοιχα θα το αρχικοποιήσουμε.

    - Με αυτόν τον τρόπο δημιουργήσαμε μία εξάρτηση μεταξύ των κλάσεων η οποία ονομάζεται ***loosely coupled*** δηλαδή δεν είναι στενά συνδεδεμένες μεταξύ τους. Αν δεν γινόταν χρήση αυτού του προτύπου θα ήμασταν αναγκασμένοι κάθε φορά που θα θέλαμε να χρησιμοποιήσουμε το αντικείμενο της κλάσης **A** θα έπρεπε να το αρχικοποιούμαι συνέχεια σε πολλά μέρη μέσα στον κώδικα.

- Singleton Pattern
    - Το πρότυπο Singleton Pattern μας δίνει τη δυνατότητα να περιορίσουμε την αρχικοποίηση ενός αντικειμένου σε ένα και μοναδικό. Με αυτό τον τρόπο μπορούμε να διαχειριστούμε εύκολα την αρχικοποίηση του και μπορούμε να έχουμε πρόσβαση στις καθολικές μεταβλητές.[7]

2.2 UML Διαγράμματα

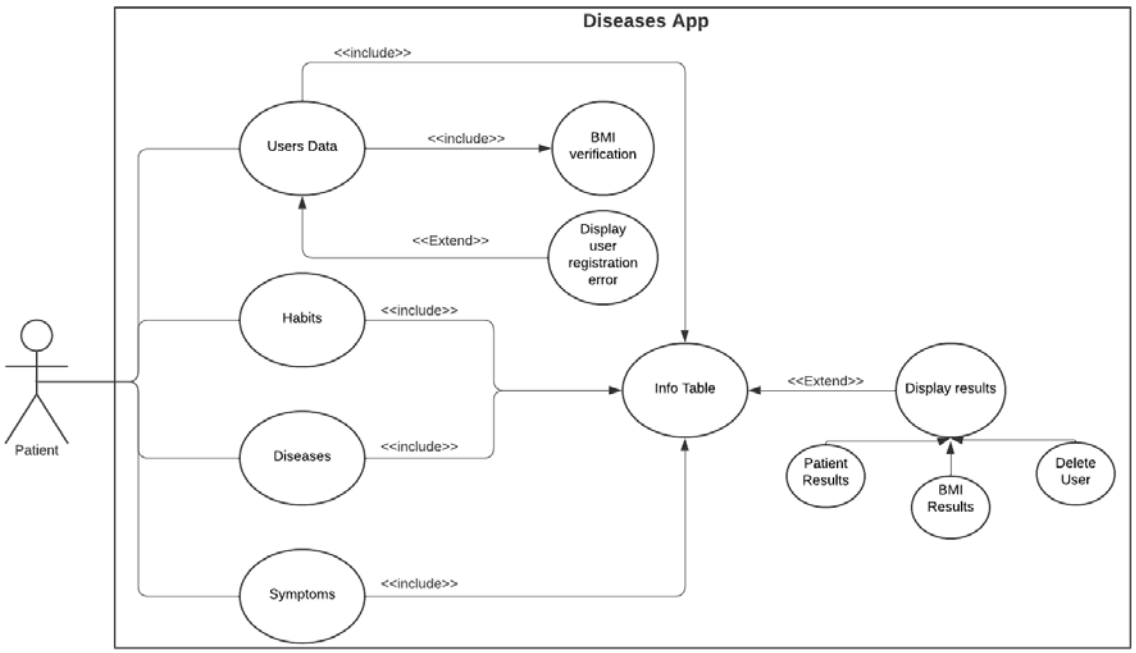
Ένα διάγραμμα UML είναι ένα διάγραμμα που βασίζεται στην UML (Unified Modeling Language) με σκοπό την οπτική αναπαράσταση ενός συστήματος μαζί με τους κύριους παράγοντες, τους ρόλους, τις ενέργειες, ή τις κλάσεις του, με σκοπό την καλύτερη κατανόηση, τροποποίηση, διατήρηση ή τεκμηρίωση πληροφοριών σχετικά με το σύστημα μας. Ωστόσο ένα τα σημαντικότερα διαγράμματα που πρέπει να υλοποιηθούν για να περιγράψουν τον τρόπο λειτουργίας της εφαρμογής είναι τα εξής διαγράμματα.[7]

2.2.1 Διαγράμματα Περιπτώσεων

Ένα διάγραμμα περιπτώσεων (Use Case Diagrams) είναι μια γραφική απεικόνιση των πιθανών αλληλεπιδράσεων ενός χρήστη με ένα σύστημα. Ένα

διάγραμμα περίπτωσης χρήσης δείχνει διάφορες περιπτώσεις χρήσης και διαφορετικούς τύπους χρηστών που διαθέτει το σύστημα και συχνά θα συνοδεύεται από άλλους τύπους διαγραμμάτων επίσης. (Wikipedia)

Για την εφαρμογή της πτυχιακής εργασίας δημιουργήθηκε το εξής διάγραμμα περιπτώσεων:



Εικόνα 13: Διάγραμμα περιπτώσεων

Όπως μπορούμε να δούμε, στο διάγραμμα υπάρχει ένας actor όπου στη συγκεκριμένη περίπτωση ονομάστηκε Patient. Ο actor προσδιορίζει τον ρόλο του χρήστη στην εφαρμογή και τις ενέργειες που μπορεί να κάνει. Όπως απεικονίζεται παραπάνω, ο χρήστης έχει τη δυνατότητα να επιλέξει τέσσερις επιλογές που θα συναντήσει στην οθόνη του υπολογιστή του με βάση το διάγραμμα.

- Η οντότητα Users Data είναι η οθόνη που μπορεί ο χρήστης να εισάγει τα προσωπικά του δεδομένα
    - Επίσης για να είναι δυνατή η εγγραφή του χρήστη στο σύστημα, πρέπει οπωσδήποτε να γίνει ο υπολογισμός του BMI για να ολοκληρωθεί η διαδικασία.
    - Για κάθε άλλη περίπτωση εμφανίζεται έναν μήνυμα σφάλματος κατά την εγγραφή του χρήστη
- Η οντότητα Habits προσδιορίζει τις συνήθειες του χρήστη
- Η οντότητα Diseases προσδιορίζει τις παθήσεις του χρήστη

- Η οντότητα Symptoms προσδιορίζει τα συμπτώματα του χρήστη

Όλες αυτές οι οντότητες καταλήγουν στο να ενωθούν με την οντότητα Info Table. Ουσιαστικά το Info Table είναι η αρχική οθόνη που συναντάει ο χρήστη όπου αφού συμπληρωθούν τα δεδομένα θα απεικονίζονται με μορφή πίνακα. Επιπρόσθετα, ο πίνακας μπορεί και διεξάγει τρεις διαφορετικές ενέργειες με βάση τα συμβάντα που καλεί ο χρήστης. Το ένα σενάριο είναι η απεικόνιση του πορίσματος, το δεύτερο σενάριο η απεικόνιση του αποτελέσματος του BMI και το τρίτο και τελευταίο είναι η διαγραφή των δεδομένων και του ίδιου του χρήστη.[7]

2.2.2 Διαγράμματα Κλάσεων

Στη μηχανική λογισμικού, ένα διάγραμμα κλάσης στην ενοποιημένη γλώσσα μοντελοποίησης είναι ένας τύπος διαγράμματος στατικής δομής που περιγράφει τη δομή ενός συστήματος δείχνοντας τις κλάσεις του συστήματος, τα χαρακτηριστικά τους, τις λειτουργίες και τις σχέσεις μεταξύ αντικειμένων. (Wikipedia)

Το διάγραμμα κλάσεων είναι το κύριο δομικό στοιχείο της αντικειμενοστραφούς μοντελοποίησης. Χρησιμοποιείται για γενική εννοιολογική μοντελοποίηση της δομής της εφαρμογής και για λεπτομερή μοντελοποίηση, μετατρέποντας τα μοντέλα σε κώδικα προγραμματισμού. Τα διαγράμματα κλάσεων επίσης μπορούν να χρησιμοποιηθούν για την μοντελοποίηση δεδομένων. Οι κλάσεις σε ένα διάγραμμα κλάσεων αντιπροσωπεύουν τόσο τα κύρια στοιχεία και τις αλληλεπιδράσεις στην εφαρμογή, όσο και τις κλάσεις που πρόκειται να προγραμματιστούν.

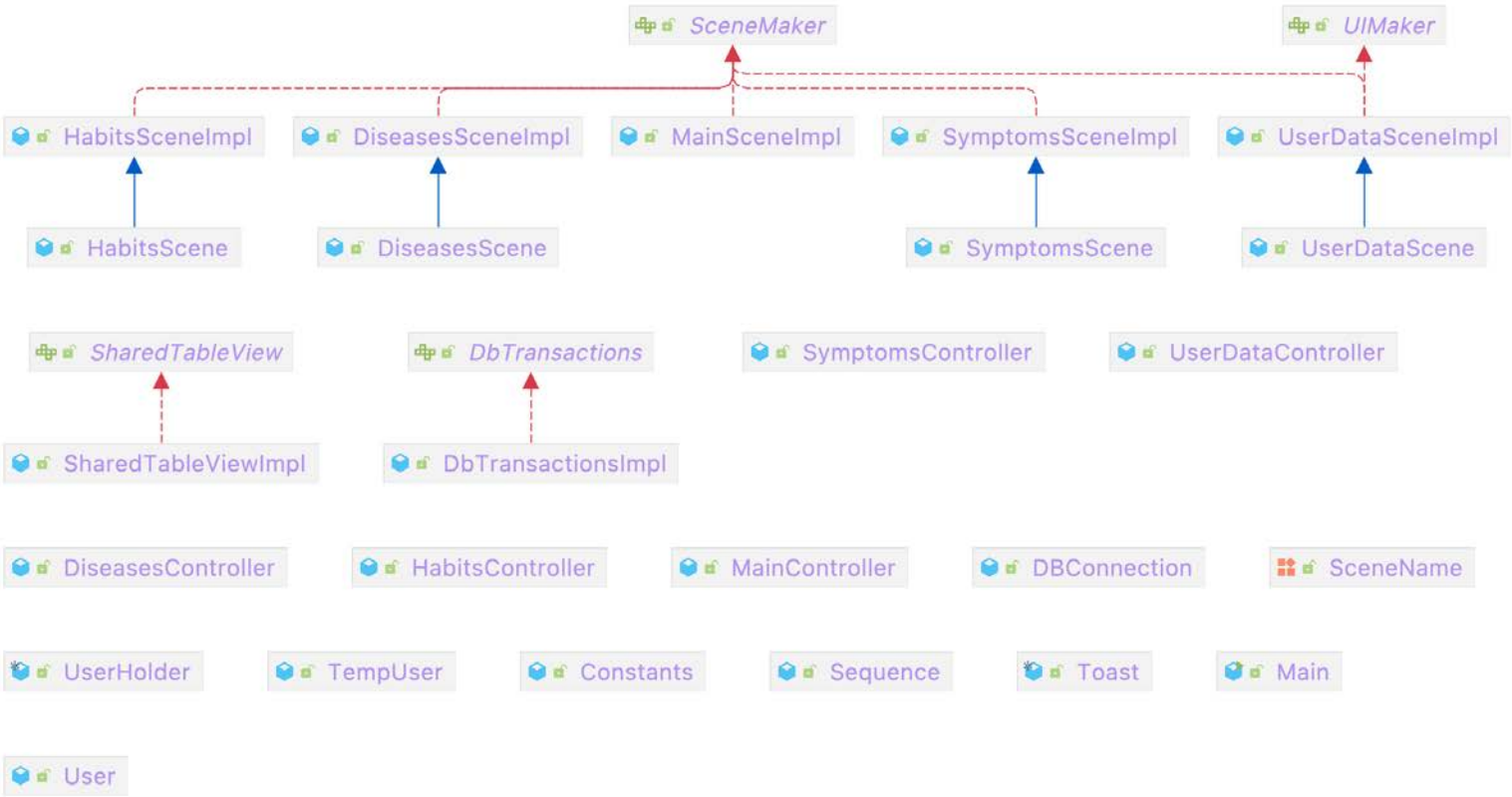
Στο διάγραμμα, οι κλάσεις αντιπροσωπεύονται με κουτιά που περιέχουν τρία διαμερίσματα:

- Το επάνω διαμέρισμα περιέχει το όνομα της τάξης. Είναι τυπωμένο με έντονους και κεντραρισμένους χαρακτήρες και το πρώτο γράμμα είναι κεφαλαίο.
- Το μεσαίο διαμέρισμα περιέχει τα χαρακτηριστικά της κλάσης. Είναι στοίχιση αριστερά και το πρώτο γράμμα είναι πεζό.

- Το κάτω διαμέρισμα περιέχει τις λειτουργίες που μπορεί να εκτελέσει η κλάση. Είναι επίσης αριστερή στοίχιση και το πρώτο γράμμα είναι πεζό.

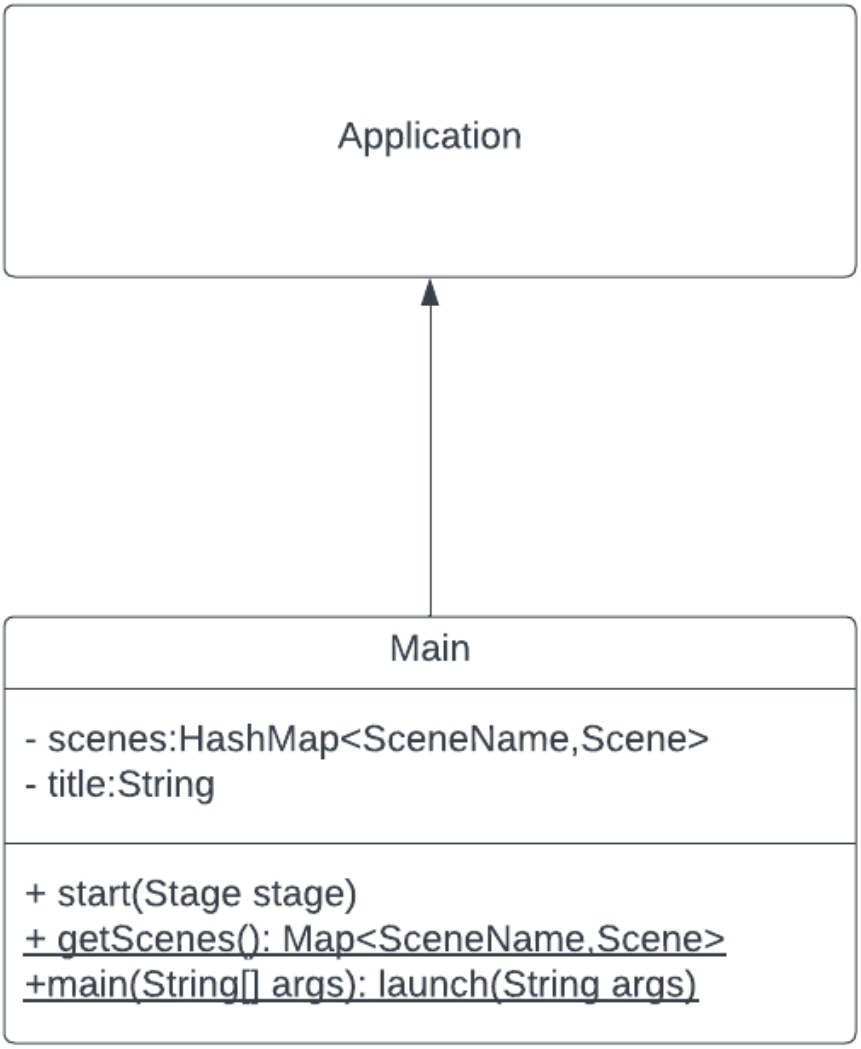
Στο σχεδιασμό ενός συστήματος, οι κλάσεων χαρακτηρίζονται και ομαδοποιούνται σε ένα διάγραμμα που βοηθά στον προσδιορισμό των στατικών σχέσεων μεταξύ τους. Στη λεπτομερή μοντελοποίηση, οι κατηγορίες του εννοιολογικού σχεδιασμού συχνά χωρίζονται σε υποκατηγορίες.

Προκειμένου να περιγραφεί περαιτέρω η συμπεριφορά των συστημάτων, αυτά τα διαγράμματα κλάσεων μπορούν να συμπληρωθούν από ένα διάγραμμα καταστάσεων ή μια μηχανή καταστάσεων UML όπως εξηγήθηκε παραπάνω. Όπως ήδη έχει παρατηρηθεί η εφαρμογή χωρίζεται σε διάφορα components.



Εικόνα 14: Συνολικό διάγραμμα της εφαρμογής

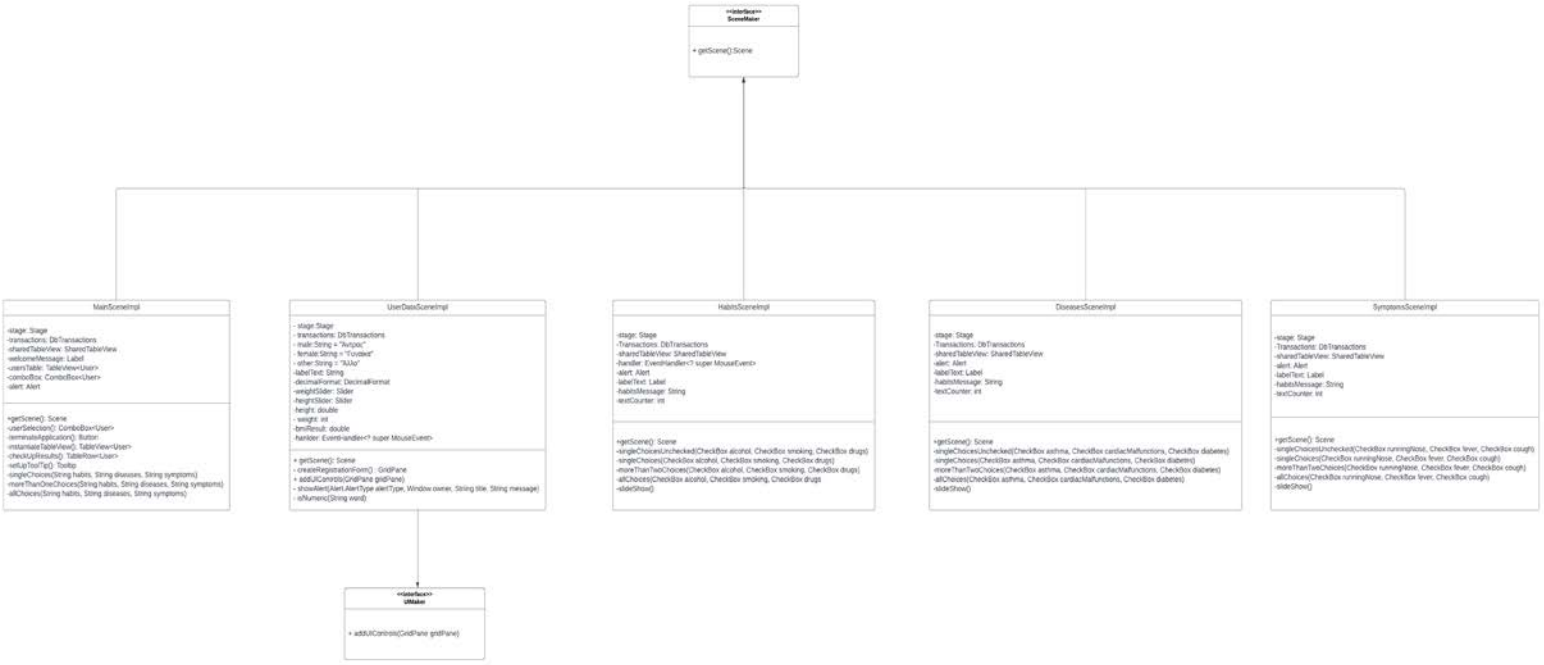
Ωστόσο παρακάτω θα γίνει η ανάλυση ενός βασικού διαγράμματος που περιγράφει την συγγραφή του κώδικα από την άποψη του προγραμματιστή με την σύνδεση ενός component της JavaFX.



Εικόνα 15: Διάγραμμα κλάσεων

Η Main κλάση έχει κατασκευαστεί από τη δική μας την πλευρά και όπως μπορούμε να δούμε στο διάγραμμα, η Main επεκτείνει την κλάση Application από την οποία κληρονομεί κάποια χαρακτηριστικά της για να μπορέσει εξολοκλήρου να γίνει η διασύνδεση της εφαρμογής με την JavaFX. Ουσιαστικά η JavaFX περιλαμβάνει μία υπερκλάση με το όνομα Application από την οποία μπορούμε να καλέσουμε ορισμένες μεθόδους. Αυτή που μας ενδιαφέρει είναι η μέθοδος start(). Η μέθοδος start περιέχει ως όρισμα την κλάση Stage.

Η κλάση Stage είναι υπεύθυνη για την δημιουργία σκηνικών στην εφαρμογή στα οποία περιλαμβάνονται γραφικά στοιχεία απεικόνισης. Αυτό που μας ενδιαφέρει είναι ότι κατά την εκκίνηση της εφαρμογής από τον χρήστη, το πρώτο πράγμα που πρέπει να πραγματοποιηθεί είναι η δημιουργία του Stage και αντίστοιχα με τη βοήθεια της μεθόδου getScenes() καλούμε τα σκηνικά που κατασκευάσαμε εμείς για να απεικονίσουμε στον χρήστη τις λειτουργίες που θα μπορέσει να χρησιμοποιήσει. [8]



Εικόνα 16: Στο παραπάνω διάγραμμα κλάσεων παρατηρούμε την διασύνδεση των κυριότερων μοντέλων στην εφαρμογή καθώς η κάθε κλάση υλοποιεί την ίδια διεπαφή.

Εικόνα 17: Στο παραπάνω διάγραμμα κλάσεων παρατηρούμε την διασύνδεση των εξαρτημάτων που πραγματοποιούν τα transactions στη βάση δεδομένων

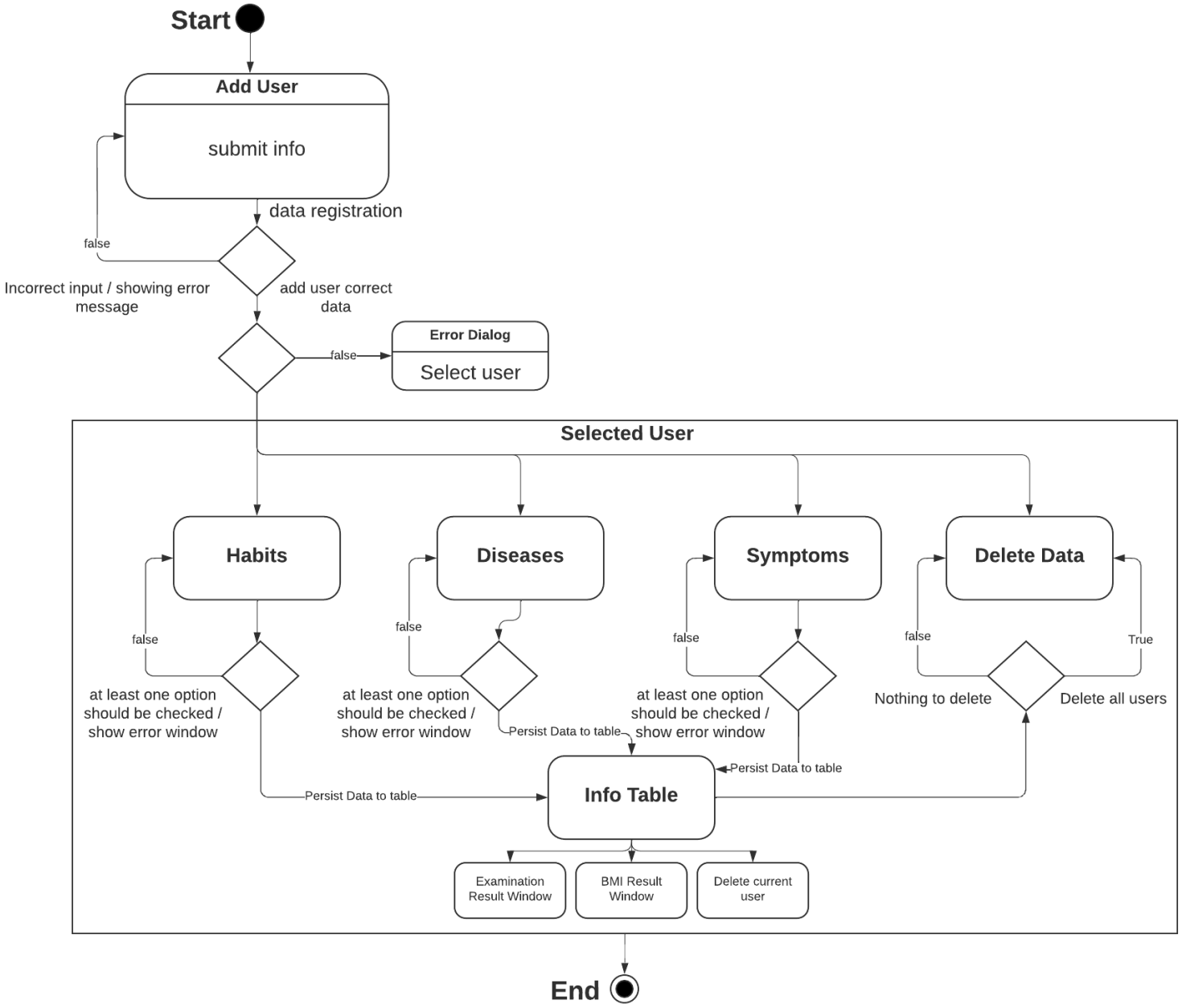
2.2.3 Διάγραμμα Καταστάσεων

Τα διαγράμματα μηχανής καταστάσεων (state machine diagrams) χρησιμοποιούνται για τη μοντελοποίηση της διακριτής συμπεριφοράς κάποιου συστήματος. Η συμπεριφορά του συστήματος μοντελοποιείται ως η μετάβαση από μία κατάσταση σε κάποια άλλη. Τα διαγράμματα μηχανής καταστάσεων

μπορούν να χρησιμοποιηθούν και για τη μοντελοποίηση της συμπεριφοράς των αντικειμένων.

Υπάρχουν περιπτώσεις όμως που η κατάσταση των αντικειμένων έχει ιδιαίτερο ενδιαφέρον για την ανάλυση. Το ενδιαφέρον αυτό είναι εντονότερο, όταν η κατάσταση ενός αντικειμένου επηρεάζει τη συμπεριφορά του. Σε αντίθεση με τα διαγράμματα επικοινωνίας που μοντελοποιούν τη συμπεριφορά και την επικοινωνία πολλών αντικειμένων που συνεργάζονται, τα διαγράμματα μηχανής καταστάσεων μοντελοποιούν τις καταστάσεις ενός αντικειμένου

Τα διαγράμματα καταστάσεων έχουν αρκετά πλούσιο συμβολισμό και χρησιμοποιούνται κυρίως για εξειδικευμένο λογισμικό, όπως λογισμικό ελεγκτών συσκευών και λογισμικό πραγματικού χρόνου. Με τα διαγράμματα μηχανής καταστάσεων (state machine diagrams) αντιμετωπίζουμε τις απαιτήσεις του συστήματος ή κάποιων μερών του ως τη μετάβαση από μία κατάσταση σε κάποια άλλη όταν το σύστημα ανταποκρίνεται σε ερεθίσματα του περιβάλλοντός του.



Εικόνα 18: Διάγραμμα καταστάσεων

Όπως παρατηρούμε στην παραπάνω εικόνα, διακρίνονται οι καταστάσεις της εφαρμογής που πραγματοποιούνται όσο η εφαρμογή βρίσκεται σε κατάσταση εκτέλεσης. Διακρίνεται η έναρξη της εφαρμογής με την μαύρη κουκίδα, η οποία καθιερώνει την πρωταρχική ενέργεια που προϋποθέτει η εφαρμογή για να λειτουργήσει σωστά και να συνεχίσει στα επιμέρους βήματα των καταστάσεων που περιμένουν να εκτελεστούν.

Στη συνέχεια, έπειτα από μια ροή συνθηκών (οι ρόμβοι), καθορίζουν τις σωστές ή τις λανθάνουσες αποφάσεις που διενεργούνται με βάση τους ελέγχους εγκυρότητας της εφαρμογής που έχουμε αναπτύξει. Εάν δεν υπήρχαν οι έλεγχοι εγκυρότητας και οι συνθήκες που καθορίζουν τη ροή της εφαρμογής, αφενός θα υπήρχε λειτουργικότητα της εφαρμογής μέχρι ένα ποσοστό, αφετέρου θα προέκυπταν μεταγενέστερα προβλήματα ως προς την αυθεντικότητα των δεδομένων που θα εκχωρούνταν στην βάση δεδομένων και πόσο μάλλον η καταγραφή δεδομένων θα γινόταν με ελλιπή τρόπο κάτι το οποίο δεν ακολουθεί τα *best practices* που πρέπει να εφαρμόζονται κατά την ανάπτυξη ενός λογισμικού .[7]

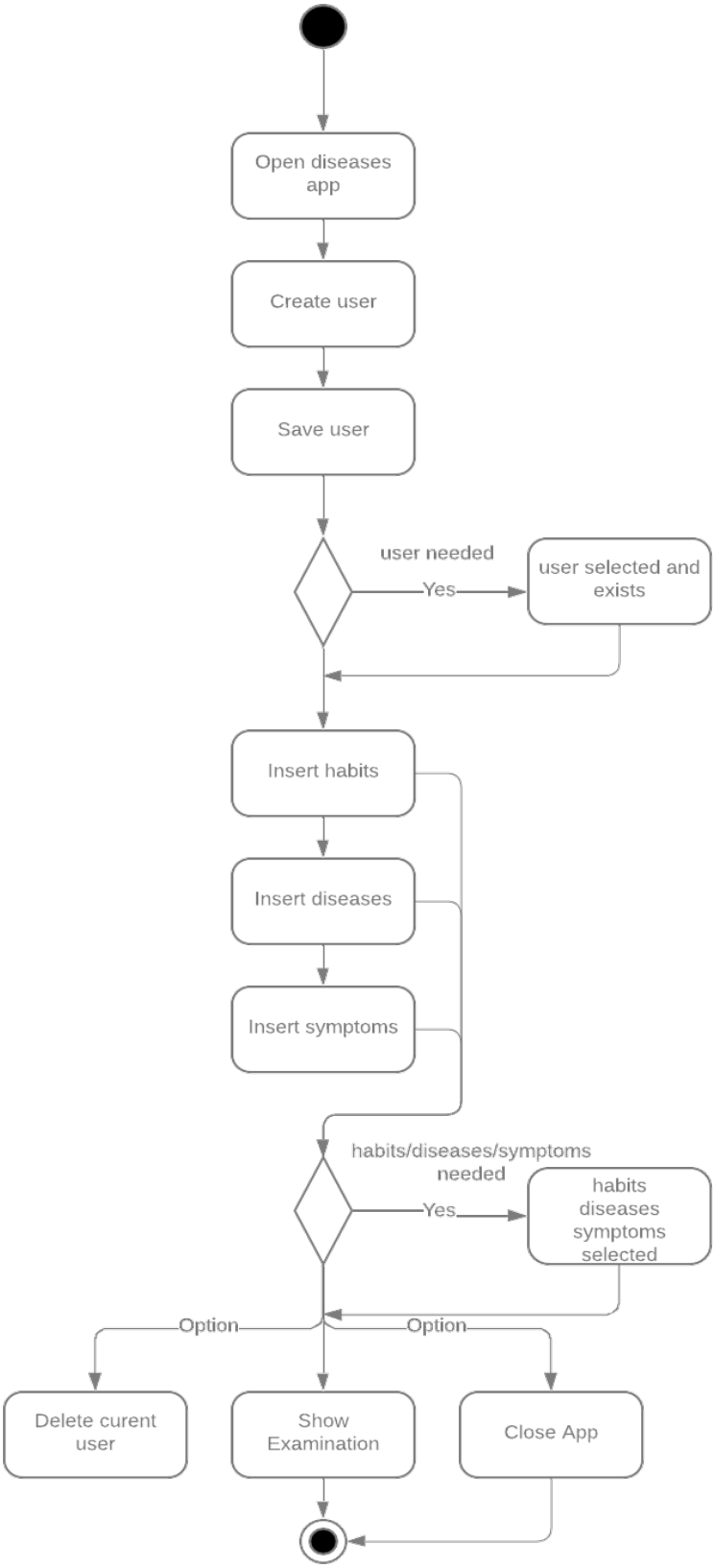
2.2.4 Διάγραμμα Δραστηριοτήτων

Τα διαγράμματα δραστηριότητας είναι διαγράμματα της UML που παρουσιάζουν ακολουθιακή ή και παράλληλη εκτέλεση δραστηριοτήτων. Μία δραστηριότητα (activity) αναπαριστά μία σύνθετη συμπεριφορά η οποία αναλύεται σε άλλες δραστηριότητες ή σε απλούστερη συμπεριφορά που είναι οι ενέργειες (actions).

Η χρήση διαγραμμάτων δραστηριότητας αξιοποιείται για

- Επιχειρησιακή μοντελοποίηση (business modeling). Περιγραφή διαδικασιών που ακολουθούνται στη λειτουργία ενός οργανισμού
- Οπτική αναπαράσταση της ροής των βημάτων μίας περίπτωσης χρήσης
- Μοντελοποίηση της λογικής του λογισμικού

Όπως βλέπουμε στο παρακάτω διάγραμμα, φαίνονται ξεκάθαρα όλες οι δραστηριότητες που εκτελούνται για να παράξει η εφαρμογή τον τελικό της στόχο. Η συγκεκριμένη ακολουθία προσδιορίζει ακριβώς τα βήματα που θα εκτελούνταν από την προοπτική του χρήστη με αρχή μέση και τέλος.[8]



Εικόνα 19: Διάγραμμα δραστηριοτήτων

2.3 Βάση Δεδομένων

2.3.1 Τι είναι μία βάση δεδομένων

Στην πληροφορική, μια βάση δεδομένων είναι μια οργανωμένη συλλογή δεδομένων που αποθηκεύονται ηλεκτρονικά. Οι μικρές βάσεις δεδομένων μπορούν να αποθηκευτούν σε ένα σύστημα αρχείων, ενώ οι μεγάλες βάσεις δεδομένων φιλοξενούνται σε συμπλέγματα υπολογιστών ή στο cloud. Ο σχεδιασμός των βάσεων δεδομένων καλύπτει τεχνικά και πρακτικά ζητήματα, συμπεριλαμβανομένης της μοντελοποίησης, της αποτελεσματικής αναπαράστασης και αποθήκευσης δεδομένων, των ερωτημάτων (queries), της ασφάλειας του απορρήτου των ευαίσθητων δεδομένων και ζητημάτων κατανεμημένων υπολογιστών, συμπεριλαμβανομένης της υποστήριξης ταυτόχρονης πρόσβασης και ανοχής σφαλμάτων.

Ένα σύστημα διαχείρισης βάσεων δεδομένων (DBMS) είναι το λογισμικό που αλληλοεπιδρά με τους τελικούς χρήστες, τις εφαρμογές και την ίδια τη βάση δεδομένων για τη συλλογή και ανάλυση των τους. Οι επιστήμονες υπολογιστών μπορούν να ταξινομήσουν συστήματα διαχείρισης βάσεων δεδομένων σύμφωνα με τα μοντέλα που υποστηρίζουν. Οι σχεσιακές βάσεις δεδομένων έγιναν κυρίαρχες τη δεκαετία του 1980. Αυτά τα δεδομένα μοντελοποιούνται ως σειρές και στήλες σε μια σειρά πινάκων και η συντριπτική πλειοψηφία χρησιμοποιεί SQL για τη σύνταξη και την αναζήτηση δεδομένων. Στη δεκαετία του 2000, οι μη σχεσιακές βάσεις δεδομένων έγιναν δημοφιλείς, οι οποίες συλλογικά αναφέρονται ως NoSQL, επειδή χρησιμοποιούν διαφορετικές γλώσσες ερωτημάτων. Ωστόσο στην πτυχιακή εργασία θα γίνει εστίαση σε σχεσιακή βάση δεδομένων χρησιμοποιώντας την MySQL.

2.3.2 Διάγραμμα οντοτήτων συσχετίσεων

Το μοντέλο οντοτήτων-συσχετίσεων (μοντέλο Ο/Σ - ER model) είναι ένα (αφαιρετικό/ιδεατό) μοντέλο δεδομένων, τα οποία έχουν καθορισμένη δομή. Στη μηχανική λογισμικού χρησιμοποιείται για να παρέχει ένα εννοιολογικό σχήμα κατά τη σχεδίαση βάσεων δεδομένων, ως μοντέλο δεδομένων ενός συστήματος και των απαιτήσεών του με top-down προσέγγιση. Ένα διάγραμμα που δημιουργείται με αυτή τη διαδικασία σχεδίασης καλείται διάγραμμα οντοτήτων-συσχετίσεων, ή διάγραμμα Ο/Σ ή ΟΣΔ εν συντομία. Προτάθηκε

αρχικά το 1976 από τον Peter Chen, ωστόσο στη συνέχεια επινοήθηκαν πολλές παραλλαγές της διαδικασίας.

Χρησιμοποιείται στο πρώτο στάδιο σχεδίασης ενός συστήματος πληροφοριών, κατά την ανάλυση των απαιτήσεών του. Σκοπός του είναι να περιγράφει τις αναγκαίες πληροφορίες οι οποίες πρόκειται να αποθηκευτούν στη βάση δεδομένων ή τον τύπο τους. Η μοντελοποίηση δεδομένων γίνεται για την περιγραφή των χρησιμοποιούμενων όρων και των σχέσεών τους σε έναν ορισμένο τομέα ενδιαφέροντος.

Στην περίπτωση σχεδιασμού ενός συστήματος πληροφοριών, που στηρίζεται σε μια βάση δεδομένων, το εννοιολογικό μοντέλο δεδομένων χαρτογραφείται σε προχωρημένο στάδιο σε ένα λογικό μοντέλο δεδομένων, όπως το σχεσιακό μοντέλο δεδομένων. Το στάδιο αυτό ονομάζεται συνήθως στάδιο λογικού σχεδιασμού. Ύστερα, κατά τη διάρκεια του φυσικού σχεδιασμού το λογικό μοντέλο χαρτογραφείται σε κάποιο φυσικό μοντέλο. Ορισμένες φορές και οι δύο φάσεις αναφέρονται ως "φυσικός σχεδιασμός".

Βάση για των μοντέλων Ο/Σ είναι η κατηγοριοποίηση αντικειμένων και των σχέσεών τους μεταξύ τους. Οντότητα Οι διάφοροι τύποι οντοτήτων παριστάνονται στο διάγραμμα Ο/Σ με ένα ορθογώνιο.

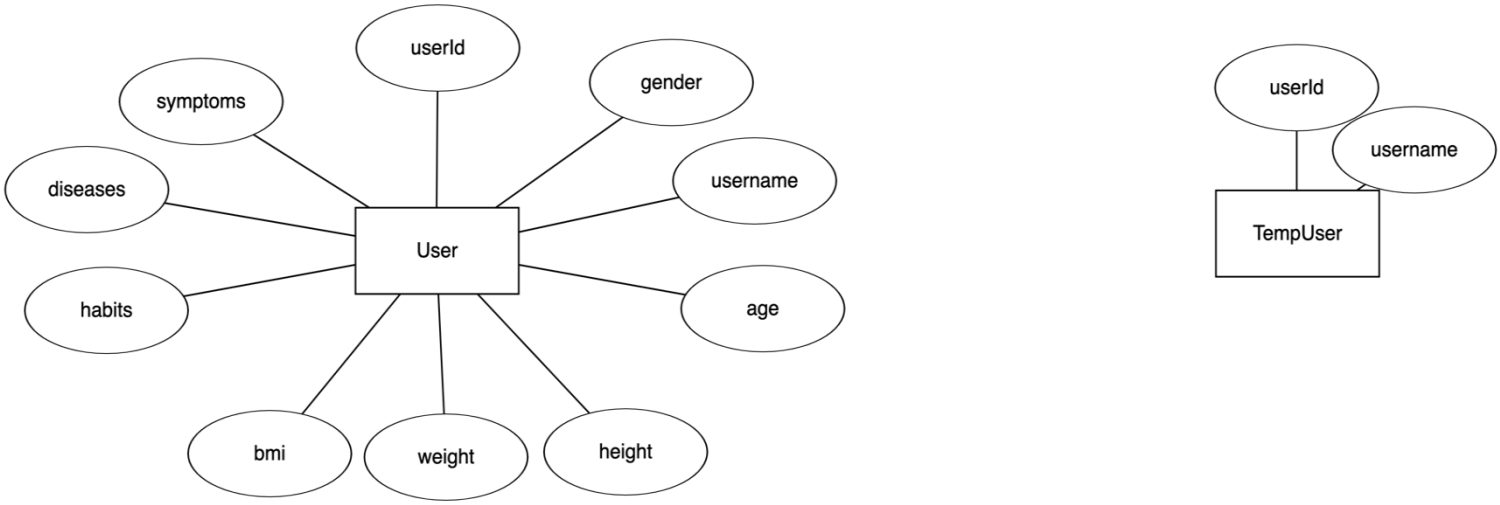
- Οντότητα (entity) είναι ένα αντικείμενο ενδιαφέροντος στον πραγματικό κόσμο το οποίο ξεχωρίζει από τα υπόλοιπα. Μια οντότητα λειτουργεί αφαιρετικά σε έναν πολύπλοκο τομέα. Οντότητες μπορεί να είναι άνθρωποι, μέρη, αντικείμενα, γεγονότα, έννοιες κλπ. Στιγμιότυπο (instance) μιας οντότητας είναι μια συγκεκριμένη περίπτωση ενός τύπου οντότητας.
- Τύπος Οντότητας: Ο τύπος της οντότητας είναι μια συλλογή χαρακτηριστικών που περιγράφουν την οντότητα. Οι διάφοροι τύποι οντοτήτων (π.χ. ΑΣΘΕΝΗΣ, ΦΟΙΤΗΤΗΣ) παριστάνονται στο διάγραμμα Ο/Σ με ένα ορθογώνιο

Χαρακτηριστικό Ένας φοιτητής μπορεί να έχει το πεδίο ΜΑΘΗΜΑ, το οποίο όμως δέχεται ως τιμές ένα σύνολο μαθημάτων που παρακολουθεί. Κάθε οντότητα έχει διάφορα στοιχεία που την προσδιορίζουν. Ένα τέτοιο στοιχείο ονομάζεται ιδιότητα (attribute), χαρακτηριστικό ή πεδίο της οντότητας.

Τα χαρακτηριστικά χωρίζονται σε μονότιμα (single valued), τα οποία έχουν μόνο μια τιμή και πλειότιμα (multi-valued), τα οποία έχουν σύνολο από

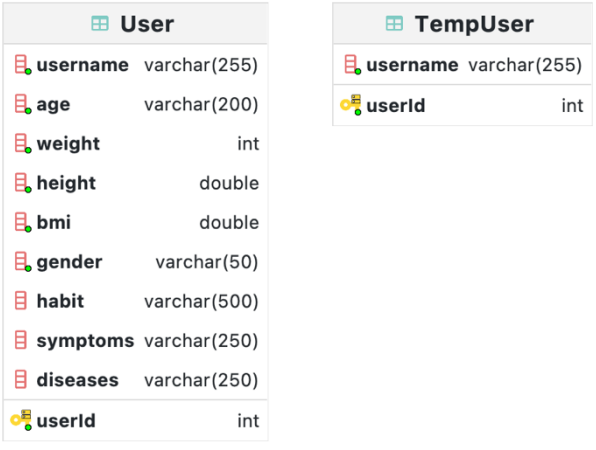
τιμές Στο διάγραμμα Ο/Σ οι ιδιότητες που έχει μια οντότητα παριστάνονται μέσα σε έλλειψη, με υπογραμμισμένο το πρωτεύον κλειδί. Τα πλειότιμα χαρακτηριστικά μιας οντότητας παριστάνονται μέσα σε έλλειψη με διπλό περίγραμμα.

Το διάγραμμα της εφαρμογής είναι σχετικά απλό. Δεν περιλαμβάνει πολλαπλές οντότητες με συσχετίσεις για λόγους ευκολίας. Στο μέλλον θα μπορούσε να διαχωριστούν κάποια χαρακτηριστικά σε πολλαπλές οντότητες.



Εικόνα 20: Διάγραμμα οντοτήτων συσχετίσεων

Ουσιαστικά υπάρχουν δύο οντότητες οι οποίες δεν συσχετίζονται μεταξύ τους. Κάποια από τα χαρακτηριστικά που περιγράφουν την οντότητα του χρήστη μπορούν να σπάσουν σε άλλες οντότητες όποιος για παράδειγμα οι συνήθειες, οι παθήσεις και τα συμπτώματα. Επιπλέον υπάρχει μία ακόμη οντότητα του «προσωρινού» χρήστη. Η χρησιμότητα και ο λόγος ύπαρξης αυτής της οντότητας θα αναλυθεί στο επόμενο κεφάλαιο.



Εικόνα 21: Διάγραμμα οντοτήτων συσχετίσεων με τύπους δεδομένων

ΚΕΦΑΛΑΙΟ 3: Τεχνολογίες και ανάπτυξη κώδικα

Σε αυτή την παράγραφο θα γίνει περιγραφή των τεχνολογιών που χρησιμοποιήθηκαν και η δομή του κώδικα συνδυαστικά με το οπτικό αποτέλεσμα της εφαρμογής.[11]

3.1 Εργαλεία που χρησιμοποιήθηκαν

Τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη του λογισμικού είναι η γλώσσα προγραμματισμού Java, η χρήση της βιβλιοθήκης JavaFX και η MySQL για την αποθήκευση δεδομένων. Στο κεφάλαιο αυτό θα γίνει εστίαση στην JavaFX η οποία έδωσε τη δυνατότητα για την υλοποίηση αυτής της desktop εφαρμογής.

Η JavaFX είναι μια πλατφόρμα λογισμικού για τη δημιουργία και την παροχή εφαρμογών, καθώς και πλούσιων εφαρμογών web που μπορούν να τρέξουν σε μια μεγάλη ποικιλία συσκευών. Η JavaFX υποστηρίζει desktop υπολογιστές και προγράμματα περιήγησης ιστού σε Microsoft Windows, Linux και macOS, καθώς και φορητές συσκευές με iOS και Android.

Σε desktop υπολογιστές, η JavaFX υποστηρίζει λειτουργικά συστήματα Windows Vista, Windows 7, Windows 8, Windows 10, macOS και Linux. Ξεκινώντας με τη

JavaFX 1.2, η Oracle έχει κυκλοφορήσει εκδόσεις beta για το Open Solaris. Σε κινητά, το JavaFX Mobile 1.x μπορεί να εκτελείται σε πολλαπλά λειτουργικά συστήματα για κινητά, συμπεριλαμβανομένων των Symbian OS, Windows Mobile και ιδιόκτητων λειτουργικών συστημάτων σε πραγματικό χρόνο.

Το JavaFX προοριζόταν να αντικαταστήσει το Swing ως την τυπική βιβλιοθήκη GUI για την Java SE, αλλά έχει αποσυρθεί από τις νέες Standard Editions ενώ το Swing και το AWT εξακολουθούν να περιλαμβάνονται, υποθετικά επειδή το μερίδιο αγοράς της JavaFX έχει «διαβρωθεί από την άνοδο του κινητών εφαρμογών και των διαδικτυακών εφαρμογών. Με την κυκλοφορία του JDK 11 το 2018, η Oracle έκανε την JavaFX μέρος του OpenJDK στο πλαίσιο του έργου OpenJFX, προκειμένου να αυξηθεί ο ρυθμός ανάπτυξής του. Η υποστήριξη της Oracle για την JavaFX είναι επίσης διαθέσιμη για Java JDK 8 έως τον Μάρτιο του 2025.

Επιπλέον για την ανάπτυξη του κώδικα έγινε η χρήση του Intellij IDE. Το IntelliJ IDEA είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης γραμμένο σε Java για την ανάπτυξη λογισμικού υπολογιστών γραμμένου σε Java, Kotlin, Groovy και άλλες γλώσσες που βασίζονται στο JAR.

Επίσης για την δημιουργία της βάσης δεδομένων και κατασκευή των πινάκων έγινε η χρήση του MySQL Workbench. Το MySQL Workbench είναι ένα εργαλείο σχεδίασης βάσεων δεδομένων που ενσωματώνει την ανάπτυξη, διαχείριση, σχεδιασμό βάσεων δεδομένων, δημιουργία και συντήρηση SQL σε ένα ενιαίο περιβάλλον ανάπτυξης για το σύστημα βάσεων δεδομένων MySQL.[12]

3.2 Κώδικας

Σε αυτή την παράγραφο, θα παρουσιαστούν τα κυριότερα σημεία του κώδικα που προσφέρουν τις λειτουργίες που έχουν αναφερθεί πιο πάνω στα διαγράμματα. Για αρχή ας ξεκινήσουμε με την δομή της JavaFX. Η JavaFX συνδυάζει μαζί με τον κώδικα της Java την χρήση της xml. Παρόλαυτα, για λόγους βελτιστοποίησης, έγινε η αποφυγή χρήσης της xml αξιοποιώντας μόνο την γλώσσα Java. Για την παρουσίαση και αναπαράσταση των γραφικών παραθύρων, η συγκεκριμένη βιβλιοθήκη χρησιμοποιεί την λογική των *Stages* (στάδια) και των *Scenes* (σκηνικά).

Ένα JavaFX Stage, javafx.stage.Stage, αντιπροσωπεύει ένα παράθυρο σε μια εφαρμογή υπολογιστή JavaFX. Μέσα σε ένα stage μπορούμε να εισαγάγουμε μια σκηνή JavaFX που αντιπροσωπεύει το περιεχόμενο που εμφανίζεται μέσα σε ένα παράθυρο (μέσα σε ένα *Stage*). Όταν ξεκινά μια εφαρμογή JavaFX, δημιουργεί ένα *root* αντικείμενο *Stage* το οποίο μεταβιβάζεται στη μέθοδο *start(Stage stage)* της *root* κλάσης της εφαρμογής JavaFX. Αυτό το αντικείμενο *Stage* αντιπροσωπεύει το κύριο παράθυρο της εφαρμογής. Μπορούμε να δημιουργήσουμε νέα αντικείμενα *Stage* αργότερα κατά τη διάρκεια της εφαρμογής σε περίπτωση που η εφαρμογή χρειαστεί να ανοίξει περισσότερα παράθυρα.

Παρόλαυτα η εφαρμογή που υλοποιήθηκε για την πτυχιακή εργασία, αξιοποιεί ένα μονάχα *Stage* καθώς μέσα του εναλλάσσει πολλαπλά Scenes. Για

λόγους καλύτερη κατανόησης, η κλάση που δημιουργήθηκε για να κληρονομεί τις ιδιότητες της JavaFX μέσω της κλάσης *Application*, ονομάστηκε *Main.*

```
public class Main extends Application {
    /**
     * Holds the various scenes to switch between
     */
    private static final Map<SceneName, Scene> scenes = new HashMap<>();
    private static final String TITLE = "Έλεγχος Ασθένειας";

    @Override
    public void start(Stage stage) {
        // Create and store all scenes up front
        scenes.put(SceneName.MAIN, new MainSceneImpl(stage).getScene());
        scenes.put(SceneName.USERDATA, new UserDataScene(stage).getScene());
        scenes.put(SceneName.HABITS, new HabitsScene(stage).getScene());
        scenes.put(SceneName.DISEASES, new DiseasesScene(stage).getScene());
        scenes.put(SceneName.SYMPTOMS, new SymptomsScene(stage).getScene());

        // Start with the main scene
        stage.setScene(scenes.get(SceneName.MAIN));
        stage.setTitle(TITLE);
        stage.getIcons().add(new Image("icon.png"));
        stage.show();
    }

    public static Map<SceneName, Scene> getScenes() {
        return scenes;
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

Όπως φαίνεται στο παραπάνω πλαίσιο του κώδικα, χρησιμοποιείται ένα HashMap για την εύκολη εκχώρηση των πολλαπλών σκηνών στο Stage και ως πρωταρχική σκηνή μέσα στο Stage ορίζεται η Main σκήνη όπου ουσιαστικά είναι το πρώτο γραφικό περιβάλλον που αντικρίζει ο χρήστης.

Επιπρόσθετα, η μέθοδος *getScenes()* επιστρέφει τα αποτελέσματα του HashMap που περιέχει τις σκηνές. Η μέθοδος είναι static το οποίο σημαίνει ότι μπορούμε να την καλέσουμε όπου θέλουμε χωρίς να δημιουργείται αντικείμενο της κλάσης Main.

Για την καλύτερη κατανομή του κώδικα και τη σωστή διαχείριση ορισμένων ενεργειών στην εφαρμογή, χρησιμοποιήθηκε το πρότυπο ModelView-Controller (MVC). Το συγκεκριμένο πρότυπο, μας βοηθάει να χωρίσουμε τον κώδικα μας σε ξεχωριστά εξαρτήματα δίνοντας τους συγκεκριμένο σκοπό. Σε γενικά πλαίσια ο **Controller**, το **Model** και το **View** είναι έννοιες που προσδιορίζουν τις κλάσεις που κατασκευάζουμε. Για παράδειγμα, μία κλάση **Model** δίνει ουσία σε έννοιες όπως αυτή του χρήστη. Δηλαδή όλες οι ιδιότητες του χρήστη που τον απαρτίζουν περιέχονται εντός μίας κλάσης **User** η οποία μοντελοποιεί τα χαρακτηριστικά που θέλουμε να έχει ο χρήστης.

```
public class User {
    private String userName;
    private String age;
    private Integer weight;
    private Double height;
    private Double bmi;
    private String gender;
    private String habit;
    private String symptoms;
    private String diseases;

    public User(String userName, String age, Integer weight, Double height, Double
bmi, String gender, String habit, String symptoms, String diseases) {
        this.userName = userName;
        this.age = age;
        this.weight = weight;
        this.height = height;
        this.bmi = bmi;
        this.gender = gender;
        this.habit = habit;
        this.symptoms = symptoms;
        this.diseases = diseases;
    }

    public User(String user) {
        this.userName = user;
    }


    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getAge() {
        return age;
    }

    public void setAge(String age) {
        this.age = age;
    }

    public String getGender() {
        return gender;
    }

    public Integer getWeight() {
        return weight;
    }

    public void setWeight(Integer weight) {
        this.weight = weight;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }

    public Double getHeight() {
        return height;
    }
```

Επιπλέον, μία κλάση **View** δε θα μπορούσε να είναι τίποτα άλλο από μία υλοποίηση των γραφικών στοιχείων που θα παρουσιάζουν τα δεδομένα στην οθόνη του χρήστη. Από την άλλη μία κλάση **Controller** έχει στόχο την ανεξάρτητη ανάπτυξη του κώδικα από τα υπόλοιπα δύο εξαρτήματα που εξηγήθηκαν. Εν ολίγοις ένας **Controller** πρέπει να διαθέτει συγκεκριμένες υλοποιήσεις όσον αφορά το *business logic* της εφαρμογής. Στη συγκεκριμένη εφαρμογή έχουμε δώσει μικρή σημασία στον Controller διότι η μεγαλύτερη υλοποίηση της εφαρμογής στηρίζεται στην οπτικοποίηση των δεδομένων.

```
public class MainController {
    private final Stage stage;
    /**
     * Injecting stage instance from main
     */
    public MainController(Stage stage) {
        this.stage = stage;

    }

    public void handleOnPressMainScene(MouseEvent event) {
        stage.setScene(Main.getScenes().get(SceneName.MAIN));
    }

    public void handleOnPressUserDataButton(MouseEvent event) {
        stage.setScene(Main.getScenes().get(SceneName.USERDATA));
    }

    public void handleOnPressHabitsButton(MouseEvent event) {
        stage.setScene(Main.getScenes().get(SceneName.HABITS));
    }

    public void handleOnPressDiseasesButton(MouseEvent event) {
        stage.setScene(Main.getScenes().get(SceneName.DISEASES));
    }

    public void handleOnPressSymptomsButton(MouseEvent event) {
        stage.setScene(Main.getScenes().get(SceneName.SYMPTOMS));
    }

}
```

Προτού όμως προχωρήσουμε στις υλοποιήσεις των Views ας δούμε την χρήση ενός Controller που αξιοποιεί από την βιβλιοθήκη της JavaFX της διαχείριση των events (τα κλικ στα κουμπιά της οθόνης).

Στον παραπάνω κώδικα βλέπουμε ότι για τα event που πραγματοποιούνται καλείται η *static* μέθοδος που υλοποιήθηκε στην κλάση *Main.* Τώρα ας δούμε πως έχει κατασκευαστεί μία κλάση View η οποία υλοποιεί τα scenes που θέλουμε να απεικονίζονται. Όπως είδαμε, η πρώτη οθόνη που εμφανίζεται στον χρήστη είναι το MainScene. Το MainScene φορτώνεται εντός του Stage. Περιέχει έναν πίνακα που θέλουμε να περιέχει όλα τα στοιχεία του

χρήστη, τις συνήθειες, τις παθήσεις και τα συμπτώματα καθώς και όλα τα κουμπιά για να μετακινούμαστε από τη μία σκηνή στην άλλη.

```
public class MainSceneImpl implements SceneMaker {

@Override
public Scene getScene() {
    // Inject stage from Main into controller
    MainController controller = new MainController(stage);


Button userData = new Button("Προσθήκη Ασθενή");
userData.setTextFill(Color.WHITE);
userData.setOnMousePressed(controller::handleOnPressUserDataButton);

Button habits = new Button("Συνήθειες");
habits.setTextFill(Color.WHITE);
Button diseases = new Button("Παθήσεις");
diseases.setTextFill(Color.WHITE);
Button symptoms = new Button("Συμπτώματα");
symptoms.setTextFill(Color.WHITE);


welcomeMessage = new Label();
welcomeMessage.setFont(new Font(20));


if (transactions.getTempUserFromDb().size() == 0) {
    welcomeMessage.setText("Καλωσόρισες!  Παρακαλώ κάντε εισαγωγή ένα νέο χρήστη ή
επιλέξτε έναν ήδη υπάρχον");
} else {
    welcomeMessage.setText("Επιλεγμένος χρήστης: " +
transactions.getTempUserFromDb().toString().replace("[", "").replace("]", ""));
}


// Build scene
VBox vbox = new VBox();
vbox.setSpacing(50);
vbox.getStyleClass().addAll("pane", "vbox");
vbox.setPadding(new Insets(10));
vbox.getChildren().addAll(selectUserLabel, userSelection(), userData, habits,
diseases, symptoms, deleteAllData, terminateApplication());

// Build window
BorderPane root = new BorderPane();
root.setLeft(vbox);
root.setTop(welcomeMessage);
root.setCenter(instantiateTableView());
Scene scene = new Scene(root, 1360, 750);
scene.getStylesheets().add(Objects.requireNonNull(getClass().getResource("/style.cs
s")).toExternalForm());
return scene;

}
```

Παρατηρούμε ότι στη μέθοδο getScene() κατασκευάζουμε ότι θέλουμε να απεικονίζει η συγκεκριμένη σκηνή με βάση κάποιους ελέγχους εγκυρότητας κατά την εισαγωγή των δεδομένων. Επίσης, είναι σημαντικό που πρέπει να αναφερθεί ότι η κλάση VBox κατασκευάζει ένα Vertical Box (κατακόρυφο κου-

τί) το οποίο μπορεί να έχει ότι διαστάσεις θέλουμε, όποια μορφοποίηση θέλουμε και φυσικά τι ενέργειες θα εκτελούνται. Για παράδειγμα η μέθοδος getChildren() μας επιτρέπει να εισχωρήσουμε παραμέτρους οι οποίες είναι γραφικά στοιχεία που πραγματοποιούν κάποια συμβάντα.

Επιπλέον, είναι εξίσου σημαντικό να εξηγήσουμε και το BorderPane. Στην JavaFX ένα Pane θεωρείται ένα κοντέινερ διάταξης που μπορεί να περιέχει εσωτερικά άλλα στοιχεία JavaFX και να τα τοποθετεί στην οθόνη. Στην πραγματικότητα, η κλάση Pane δεν παρέχει κανέναν αλγόριθμο διάταξης. Η κλάση Pane απλώς εμφανίζει τα στοιχεία που περιέχει στις θέσεις που θέλουν να βρίσκονται τα ίδια τα στοιχεία.

```
private ComboBox<User> userSelection() {
    comboBox = new ComboBox<>(transactions.getUsers());
    //the event which selects the click item from comboBox
    EventHandler<ActionEvent> comboEvent =
            actionEvent -> {
                welcomeMessage.setText("Επιλεγμένος χρήστης: " +
comboBox.getValue().getUserName());
                transactions.cacheUser(comboBox.getValue().getUserName());

sharedTableView.instantiateTableView().setItems(transactions.getTempUserFromDb());
                stage.setScene(getScene());
            };
    comboBox.getSelectionModel().select(new
User(transactions.getTempUserFromDb().toString().replace("[", "").replace("]",
"")));
    comboBox.setOnAction(comboEvent);
    return comboBox;
}
```

Στον παραπάνω κώδικα πραγματοποιείται η υλοποίηση ενός comboBox για την επιλογή του χρήστη από το γραφικό περιβάλλον. Το comboBox στην JavaFX είναι μία αναδυόμενη γραφική λίστα που μας διευκολύνει να επιλέξουμε μία μόνο εγγραφή από τα δεδομένα που υπάρχουν εντός της λίστας.

```
private TableView<User> instantiateTableView() {
    usersTable = new TableView<>();
    usersTable.setPlaceholder(new Label("Παρακαλώ εισάγετε ασθενή"));
    TableColumn<User, String> name = new TableColumn<>("Όνομα");
    name.setCellValueFactory(new PropertyValueFactory<>("userName"));
    TableColumn<User, String> age = new TableColumn<>("Ηλικία");
    age.setCellValueFactory(new PropertyValueFactory<>("age"));
    TableColumn<User, String> gender = new TableColumn<>("Φύλλο");
    gender.setCellValueFactory(new PropertyValueFactory<>("gender"));
    TableColumn<User, Integer> weight = new TableColumn<>("Σωματικό Βάρος (kg)");
    weight.setCellValueFactory(new PropertyValueFactory<>("weight"));
    TableColumn<User, Double> height = new TableColumn<>("Ύψος (μέτρα)");
    height.setCellValueFactory(new PropertyValueFactory<>("height"));
    TableColumn<User, Double> bmi = new TableColumn<>("BMI");
    bmi.setCellValueFactory(new PropertyValueFactory<>("bmi"));
    TableColumn<User, String> habits = new TableColumn<>("Συνήθειες");
    habits.setCellValueFactory(new PropertyValueFactory<>("habit"));
    TableColumn<User, String> diseases = new TableColumn<>("Παθήσεις");
    diseases.setCellValueFactory(new PropertyValueFactory<>("diseases"));
    TableColumn<User, String> symptoms = new TableColumn<>("Συμπτώματα");
    symptoms.setCellValueFactory(new PropertyValueFactory<>("symptoms"));
    usersTable.setItems(transactions.getUsers());
    usersTable.getSelectionModel().setSelectionMode(SelectionMode.SINGLE);
    usersTable.getColumns().addAll(name, age, gender, weight, height, bmi, habits,
diseases, symptoms);
    usersTable.setRowFactory(tableView -> checkUpResults());
    return usersTable;
}
```

Στον παραπάνω κώδικα αρχικοποιείται το γραφικό στοιχείο που λαμβάνει τους χρήστες από την βάση δεδομένων έτσι ώστε όταν ο χρήστης θελήσει να επιλέξει τον χρήστη που θέλει να εξετάσει, να κάνει κλικ πάνω του για να ολοκληρωθεί η διαδικασία της επιλογής του χρήστη.

```
//με αριστερό κλικ προκύπτει το πόρισμα
//με δεξί κλικ πραγματοποιείται η διαγραφή του επιλεγμένου χρήστη
//με το κλικ της ροδέλας γίνεται υπολογισμός bmi
private TableRow<User> checkUpResults() {
    TableRow<User> tableRow = new TableRow<>();
    tableRow.setOnMouseClicked(event -> {
        User rowData = tableRow.getItem();
        if (rowData == null) {
            alert = new Alert(Alert.AlertType.ERROR, "Το πεδίο που επέλεξες είναι
άδειο", ButtonType.FINISH);
            alert.showAndWait();
        } else if (Objects.equals(rowData.getUserName(),
transactions.getTempUserFromDb().toString().replace("[", "").replace("]", ""))) {
            if (event.getButton() == MouseButton.PRIMARY) {

                singleChoices(rowData.getHabit(), rowData.getDiseases(),
rowData.getSymptoms());
                moreThanOneChoices(rowData.getHabit(), rowData.getDiseases(),
rowData.getSymptoms());
                allChoices(rowData.getHabit(), rowData.getDiseases(),
rowData.getSymptoms());

            } else if (event.getButton() == MouseButton.SECONDARY) {
                Alert alert = new Alert(Alert.AlertType.WARNING, "Είσαι σίγουρος
ότι θέλεις να διαγράψεις τον χρήστη " + rowData.getUserName() + " ;",
ButtonType.NO, ButtonType.YES);
                alert.showAndWait()
                        .filter(response -> response == ButtonType.YES)
                        .ifPresent(response -> {
                            try {

transactions.deleteUser(rowData.getUserName().replace("[", "").replace("]", ""));
                                //εδω ανανεώνεται το scene του stage για να φανούν
οι αλλαγές στα γραφικά
                                stage.setScene(getScene());
                            } catch (SQLException e) {
                                throw new RuntimeException(e);
                            }
                        });
            } else if (event.getButton() == MouseButton.MIDDLE) {
                if (rowData.getGender().contains("Άντρας")) {
                    if (rowData.getBmi() < 19.0) {
                        alert = new Alert(Alert.AlertType.WARNING, "Ο χρήστης " +
rowData.getUserName() + " που επέλεξες είναι ελλιποβαρής ", ButtonType.OK);
                        alert.showAndWait();
                    } else if (rowData.getBmi() >= 19.0 && rowData.getBmi() <=
25.99) {
                        alert = new Alert(Alert.AlertType.WARNING, "Ο χρήστης " +
rowData.getUserName() + " που επέλεξες έχει φυσιολογικό βάρος ", ButtonType.OK);
                        alert.showAndWait();
                    } else if (rowData.getBmi() >= 26.0 && rowData.getBmi() <=
29.99) {
                        alert = new Alert(Alert.AlertType.WARNING, "Ο χρήστης " +
rowData.getUserName() + " που επέλεξες είναι υπέρβαρος ", ButtonType.OK);
                        alert.showAndWait();
                    } else if (rowData.getBmi() >= 30.0 && rowData.getBmi() <=
34.99) {
                        alert = new Alert(Alert.AlertType.WARNING, "Ο χρήστης " +
rowData.getUserName() + " που επέλεξες είναι παχύσαρκος ", ButtonType.OK);
                        alert.showAndWait();
                    } else if (rowData.getBmi() >= 35.0 && rowData.getBmi() <=
39.99) {
                        alert = new Alert(Alert.AlertType.WARNING, "Ο χρήστης " +
rowData.getUserName() + " που επέλεξες έχει σοβαρή παχυσαρκία", ButtonType.OK);
                        alert.showAndWait();
                    } else if (rowData.getBmi() >= 40.0 && rowData.getBmi() <=
```

Στον κώδικα παραπάνω, βλέπουμε την υλοποίηση της μεθόδου check-UpResults() η οποία καλείται στην προηγούμενη μέθοδο που γίνεται η αρχικοποίηση του γραφικού πίνακα. Αυτό γίνεται διότι θέλουμε να γίνεται ο έλεγχος του check up πάνω σε κάθε γραμμή του πίνακα. Υποτίθεται ότι κάθε γραμμή είναι ένας χρήστης που θέλουμε να εξετάσουμε. Αν δεν υπάρχουν δεδομένα πάνω στις γραμμές τότε εκτελείται ένας έλεγχος εγκυρότητας και ελέγχει ότι δεν βρέθηκε χρήστης.

Στις υπόλοιπες κλάσεις η κυριότερη λειτουργία είναι ο έλεγχος των επιλογών του χρήστη για τις συνήθειες, παθήσεις ή συμπτώματα που έχει και όταν επιλεγούν γίνονται εισαγωγή αυτών των επιλογών στην βάση δεδομένων και αντίστοιχα ο γραφικός πίνακας αντλεί αυτές τις επιλογές.

```
private void singleChoicesUnchecked(CheckBox alcohol, CheckBox smoking, CheckBox
drug) {
    if (!alcohol.isSelected() && !smoking.isSelected() && !drug.isSelected()) {
        alert = new Alert(Alert.AlertType.WARNING, "Παρακαλώ συμπληρώστε
τουλάχστιον μία ή περισσότερες συνήθειες");
        alert.showAndWait();
    }
}

private void singleChoices(CheckBox alcohol, CheckBox smoking, CheckBox drugs) {
    if (alcohol.isSelected()) {
        transactions.insertHabits(alcohol.getText(),
sharedTableView.instantiateTableView().getItems().toString().replace("[",
"").replace("]", ""));
        alert = new Alert(Alert.AlertType.CONFIRMATION, "Καταχωρήθηκε με
επιτυχία");
        alert.showAndWait();
    } else if (smoking.isSelected()) {
        transactions.insertHabits(smoking.getText(),
sharedTableView.instantiateTableView().getItems().toString().replace("[",
"").replace("]", ""));
        alert = new Alert(Alert.AlertType.CONFIRMATION, "Καταχωρήθηκε με
επιτυχία");
        alert.showAndWait();
    } else if (drugs.isSelected()) {
        transactions.insertHabits(drugs.getText(),
sharedTableView.instantiateTableView().getItems().toString().replace("[",
"").replace("]", ""));
        alert = new Alert(Alert.AlertType.CONFIRMATION, "Καταχωρήθηκε με
επιτυχία");
        alert.showAndWait();
    }
}
```

```
private void moreThanTwoChoices(CheckBox alcohol, CheckBox smoking, CheckBox drugs)
{
    if (alcohol.isSelected() && smoking.isSelected()) {
        transactions.insertHabits(alcohol.getText() + ", " + smoking.getText(),
sharedTableView.instantiateTableView().getItems().toString().replace("[",
"").replace("]", ""));
    } else if (alcohol.isSelected() && drugs.isSelected()) {
        transactions.insertHabits(alcohol.getText() + ", " + drugs.getText(),
sharedTableView.instantiateTableView().getItems().toString().replace("[",
"").replace("]", ""));
    } else if (drugs.isSelected() && alcohol.isSelected()) {
        transactions.insertHabits(drugs.getText() + ", " + alcohol.getText(),
sharedTableView.instantiateTableView().getItems().toString().replace("[",
"").replace("]", ""));
    } else if (smoking.isSelected() && alcohol.isSelected()) {
        transactions.insertHabits(smoking.getText() + ", " + alcohol.getText(),
sharedTableView.instantiateTableView().getItems().toString().replace("[",
"").replace("]", ""));
    } else if (smoking.isSelected() && drugs.isSelected()) {
        transactions.insertHabits(smoking.getText() + ", " + drugs.getText(),
sharedTableView.instantiateTableView().getItems().toString().replace("[",
"").replace("]", ""));
    } else if (drugs.isSelected() && smoking.isSelected()) {
        transactions.insertHabits(drugs.getText() + ", " + smoking.getText(),
sharedTableView.instantiateTableView().getItems().toString().replace("[",
"").replace("]", ""));
    }
}
private void allChoices(CheckBox alcohol, CheckBox smoking, CheckBox drugs) {
    if (alcohol.isSelected() && smoking.isSelected() && drugs.isSelected()) {
        transactions.insertHabits(alcohol.getText() + ", " + smoking.getText() + ",
" + drugs.getText(),
sharedTableView.instantiateTableView().getItems().toString().replace("[",
"").replace("]", ""));
    } else if (alcohol.isSelected() && drugs.isSelected() && smoking.isSelected())
{
        transactions.insertHabits(alcohol.getText() + ", " + drugs.getText() + ", "
+ smoking.getText(),
sharedTableView.instantiateTableView().getItems().toString().replace("[",
"").replace("]", ""));
    } else if (smoking.isSelected() && alcohol.isSelected() && drugs.isSelected())
{
        transactions.insertHabits(smoking.getText() + ", " + alcohol.getText() + ",
" + drugs.getText(),
sharedTableView.instantiateTableView().getItems().toString().replace("[",
"").replace("]", ""));
    } else if (smoking.isSelected() && drugs.isSelected() && alcohol.isSelected())
{
        transactions.insertHabits(smoking.getText() + ", " + drugs.getText() + ", "
+ alcohol.getText(),
sharedTableView.instantiateTableView().getItems().toString().replace("[",
"").replace("]", ""));
    } else if (drugs.isSelected() && alcohol.isSelected() && smoking.isSelected())
{
        transactions.insertHabits(drugs.getText() + ", " + alcohol.getText() + ", "
+ smoking.getText(),
sharedTableView.instantiateTableView().getItems().toString().replace("[",
"").replace("]", ""));
    } else if (drugs.isSelected() && smoking.isSelected() && alcohol.isSelected())
{
        transactions.insertHabits(drugs.getText() + "," + smoking.getText() + ", "
+ alcohol.getText(),
sharedTableView.instantiateTableView().getItems().toString().replace("[",
"").replace("]", ""));
    }
}
```

Τώρα είναι σημαντικό να αναφερθεί η υλοποίηση διασύνδεση της βάσης δεδομένων με την Java. Για την συγκεκριμένη λειτουργία έχουν υλοποιηθεί τα κατάλληλα εξαρτήματα κλάσεων που στοχεύουν στην βέλτιστη λύση για την αρχικοποίηση των αντικειμένων τους στις υπόλοιπες κλάσεις. Σε προηγούμενη παράγραφο έγινε η αναφορά του Singleton Pattern. Ένα από τα σημαντικότερα παραδείγματα που συναντάμε στις εφαρμογές είναι η κλάση που πραγματοποιεί τη σύνδεση με την βάση. Ας δούμε πως γίνεται αυτό:

```
public class DBConnection {
    private static Connection connection;

    private DBConnection() {

    }
    public static Connection getConnection() {
        try {
            if (connection == null) {
                connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/diseases", "root", "1234");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return connection;
    }
}
```

Παρατηρούμε ότι οι ιδιότητες της κλάσεις είναι private οπότε δεν είναι ορατές μέσω των υπόλοιπων κλάσεων, ωστόσο είναι σημαντικό να προσέξουμε ότι η μέθοδος getConnection() είναι public static. Αυτό σημαίνει ότι όταν ένα αντικείμενο μίας κλάσης, όταν θέλει να αποδώσει δεδομένα στην Β.Δ,
τότε απλά αυτή η μέθοδος καλείται κατευθείαν με το όνομα της κλάσης DBConnection χωρίς να γίνει η αρχικοποίηση της.

Ουσιαστικά η αρχικοποίηση γίνεται μονάχα εμφωλευμένα σε αυτή την κλάση και μέσω της static μεθόδου απλά έχουμε κατευθείαν πρόσβαση στην λειτουργίας της. Αυτό γίνεται διότι θέλουμε οπωσδήποτε να αποφύγουμε τα πολλαπλά Connections στην Β.Δ (ουσιαστικά θέλουμε να πραγματοποιείται μόνο ένα Connection) διότι μπορεί να προκληθούν προβλήματα ως προς την ταχύτητα της εφαρμογής και όχι μόνο.

Παρακάτω θα δούμε την υλοποίηση των SQL ερωτημάτων για την εισαγωγή, την λήψη δεδομένων αλλά και πως γίνεται ο έλεγχος του επιλεγμένου χρήστη!

```
public interface DbTransactions {
    ObservableList<User> getUsers();

    ObservableList<TempUser> getTempUserFromDb();

    void createUser(final String name, final String age, final int weight, final
double height, final double bmi, final String gender);

    void deleteUser(final String name) throws SQLException;

    void insertHabits(final String habit, final String userName);

    void cacheUser(final String name);

    void insertDiseases(final String diseases, final String userName);

    void insertSymptoms(final String symptoms, final String userName);

    boolean existsByUserName(String result);

    void deleteEverything();

}
```

Στον παραπάνω κώδικα βλέπουμε την δομή μίας διεπαφής (interface). Αυτό το κάναμε για να μπορέσουμε να διατηρήσουμε τα εξαρτήματα του κώδικα ανεξάρτητα μεταξύ τους αποδίδοντας σε κάθε εξάρτημα που δημιουργούμε το δικό του ρόλο χωρίς να «σπάσει» στο μέλλον ο κώδικας από τυχόν προσθήκες οι αλλαγές που ενδεχομένως να προκύψουν.

Ας υποθέσουμε ότι μία από αυτές τις μεθόδους (που θα δούμε παρακάτω να υλοποιούνται), χρειαστεί αλλαγή του ονόματος της μεθόδου ή προσθήκη μίας παραμέτρου. Φανταστείτε να ήταν διάσπαρτες αυτές οι μέθοδοι μέσα στον κώδικα και να ήταν γραμμένες πολλαπλές φορές. Θα ήταν πολύ δύσκολο να θυμόμαστε που βρίσκονται για να κάνουμε τις αλλαγές. Παρόλαυτα, το σημαντικότερο είναι ότι αν βρίσκονταν παντού διάσπαρτες, θα είχαμε διπλότυπο κώδικα (πολλαπλά αντίγραφα).

```
public class DbTransactionsImpl implements DbTransactions {
    private final Connection connection = DBConnection.getConnection();
    @Override
    public ObservableList<User> getUsers() {
        ObservableList<User> userObservableList = FXCollections.observableArrayList();
        try {
            String query = "select * from User";
            PreparedStatement getUsers = connection.prepareStatement(query);
            ResultSet resultSet = getUsers.executeQuery();
            while (resultSet.next()) {
                String id = resultSet.getString(1);
                String name = resultSet.getString(2);
                String age = resultSet.getString(3);
                Integer weight = resultSet.getInt(4);
                Double height = resultSet.getDouble(5);
                Double bmi = resultSet.getDouble(6);
                String gender = resultSet.getString(7);
                String habit = resultSet.getString(8);
                String symptoms = resultSet.getString(9);
                String diseases = resultSet.getString(10);
                User user = new User(name, age, weight, height, bmi, gender, habit, symptoms,
diseases);
                System.out.println(id + " " + name);
                userObservableList.add(user);
            }
            getUsers.close();
            resultSet.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return userObservableList;
    }
    @Override
    public ObservableList<TempUser> getTempUserFromDb() {
        ObservableList<TempUser> tempUserObservableList = FXCollections.observableArrayList();
        try {
            String query = "select * from TempUser";
            PreparedStatement getTempUser = connection.prepareStatement(query);
            ResultSet resultSet = getTempUser.executeQuery();
            while (resultSet.next()) {
                String id = resultSet.getString(1);
                String name = resultSet.getString(2);
                TempUser user = new TempUser(name);
                tempUserObservableList.add(user);
            }
            getTempUser.close();
            resultSet.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return tempUserObservableList;
    }
```

```
    @Override
    public void createUser(final String name, final String age, final int weight, final
double height, final double bmi, final String gender) {
        try {
            PreparedStatement createUser = connection.prepareStatement("INSERT INTO User
(username,age,weight,height,bmi,gender) VALUES (?,?,?,?,?,?)");
            createUser.setString(1, name);
            createUser.setString(2, age);
            createUser.setInt(3, weight);
            createUser.setDouble(4, height);
            createUser.setDouble(5, bmi);
            createUser.setString(6, gender);
            createUser.executeUpdate();
            createUser.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

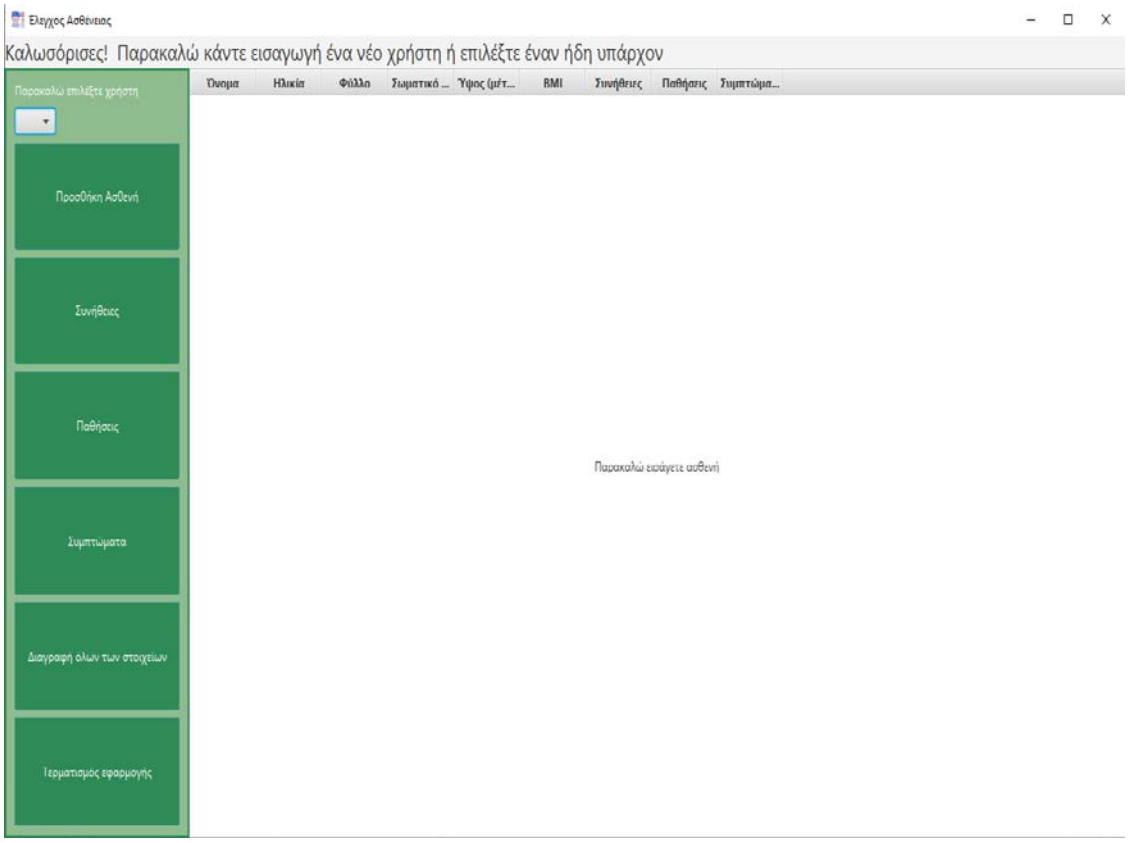
Όπως φαίνεται, ο παραπάνω κώδικας είναι μία κλάση (DbTransactionsImpl) που υλοποιεί τις μεθόδους που περιέχει η διεπαφή (interface). Ουσιαστικά τις «κληρονομεί» από την διεπαφή και απλά προστίθεται ο κορμός της υλοποίησης για την κάθε μέθοδο. Οπότε, τώρα αν ανατρέξουμε λίγο στην κλάση που δημιουργείται ο χρήστης, θα αρχικοποιούσαμε απλά ένα αντικείμενο της κλάσης DbTransactionsImpl και θα καλούσαμε επιτόπου την μέθοδο createUser() και τίποτα παραπάνω! Αντίστοιχα η ίδια λογική πραγματοποιείται και στις υπόλοιπες κλάσεις.

```
@Override
    public void deleteUser(String name) throws SQLException {
        try {
            PreparedStatement deleteUserFromUserTable = connection.prepareStatement("DELETE
FROM User WHERE username=?");
            deleteUserFromUserTable.setString(1, name);

            PreparedStatement deleteUserFromTempUserTable =
connection.prepareStatement("DELETE FROM TempUser WHERE username=?");
            deleteUserFromTempUserTable.setString(1, name);

            deleteUserFromUserTable.executeUpdate();
            deleteUserFromTempUserTable.executeUpdate();

            deleteUserFromUserTable.close();
            deleteUserFromTempUserTable.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @Override
    public void upsertHabits(final String habit, final String userName) {
        try {
            PreparedStatement upserteHabits = connection.prepareStatement("UPDATE User SET
habit =? where username=?");
            updateHabits.setString(1, habit);
            updateHabits.setString(2, userName);
            updateHabits.executeUpdate();
            updateHabits.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @Override
    public void upsertDiseases(final String diseases, final String userName) {
        try {
            PreparedStatement upsertDiseases = connection.prepareStatement("UPDATE User SET
diseases =? where username=?");
            insertDiseases.setString(1, diseases);
            insertDiseases.setString(2, userName);
            insertDiseases.executeUpdate();
            insertDiseases.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @Override
    public void upsertSymptoms(String symptoms, String userName) {
        try {
            PreparedStatement upsertSymptoms = connection.prepareStatement("UPDATE User SET
symptoms =? where username=?");
            insertSymptoms.setString(1, symptoms);
            insertSymptoms.setString(2, userName);
            insertSymptoms.executeUpdate();
            insertSymptoms.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @Override
    public void cacheUser(final String name) {
        try {
            int id = 1;
            PreparedStatement cacheUser = connection.prepareStatement("INSERT INTO TempUser
(userId, username) VALUES (?,?) ON DUPLICATE KEY UPDATE username = VALUES(username) ");
            cacheUser.setInt(1, id);
            cacheUser.setString(2, name);
            cacheUser.executeUpdate();
            cacheUser.close();
        } catch (Exception e) {
```

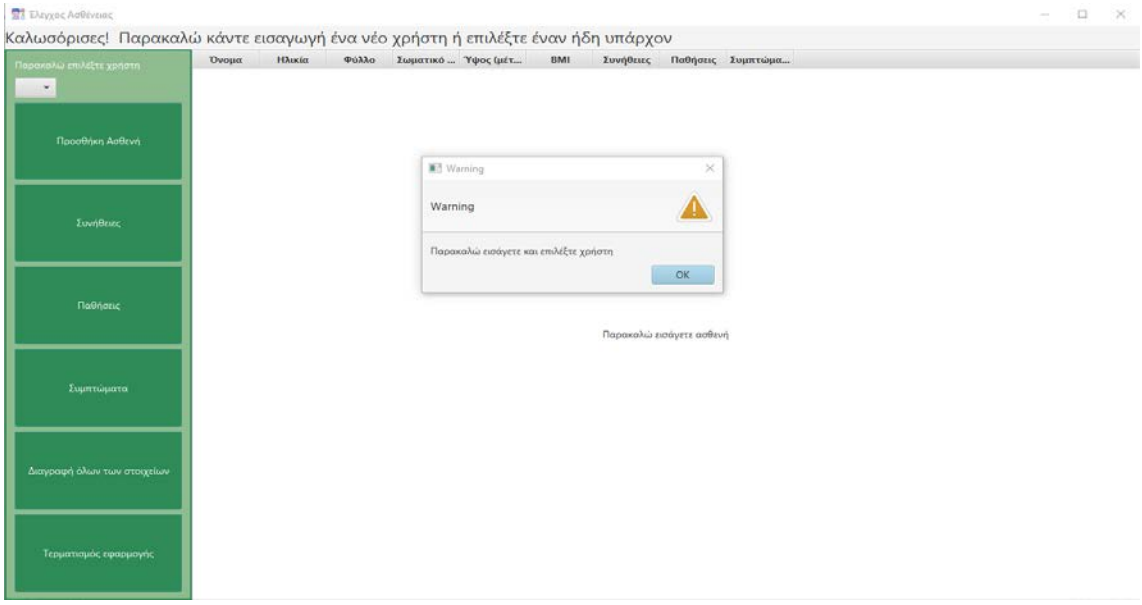
Εξίσου σημαντικό είναι να αναφερθεί και η λογική της μεθόδου cacheUser(). Στην αρχική οθόνη της εφαρμογής παρατηρούμε μία κενή λίστα για την επιλογή του χρήστη μέσω του γραφικού περιβάλλοντος. Μόλις ο χρήστης επιλέξει αυτή την λίστα, θα του εμφανίσει όλους τους εγγεγραμμένους χρήστες που υπάρχουν στη Β.Δ. Από εκεί και έπειτα, εφόσον ο χρήστης κάνει κλικ σε έναν ασθενή, αυτό που θέλουμε να πραγματοποιείται από πίσω είναι, να κρατήσουμε σε δύο μεταβλητές το id και το username του χρήστη και να τα αποθηκεύσουμε εκ νέου στον πίνακα TempUser.

Αφού εκχωρηθούν στον πίνακα αυτά τα δεδομένα, θέλουμε κάθε φορά που επιλέγουμε από την λίστα έναν νέο χρήστη να αντικαθίσταται η προηγούμενη εγγραφή με την επόμενη. Αυτό επιτυγχάνεται καθώς ελέγχουμε εάν το primary key της προηγούμενης εγγραφής είναι ίδιο με της επόμενης. Ουσιαστικά, κάθε εγγραφή στον TempUser έχει πάντα για id τον αριθμό 1. Επομένως, αν ο χρήστης της εφαρμογής κάνει κλικ πάνω στον πίνακα των αποτελεσμάτων για να δει το πόρισμα του ασθενή που επιθυμεί, εάν ο επιλεγμένος χρήστης δεν είναι ίδιος με αυτόν που κάνει κλικ, τότε θα εμφανιστεί μήνυμα σφάλματος ειδάλλως θα γίνει απεικόνιση του πορίσματος.[6]

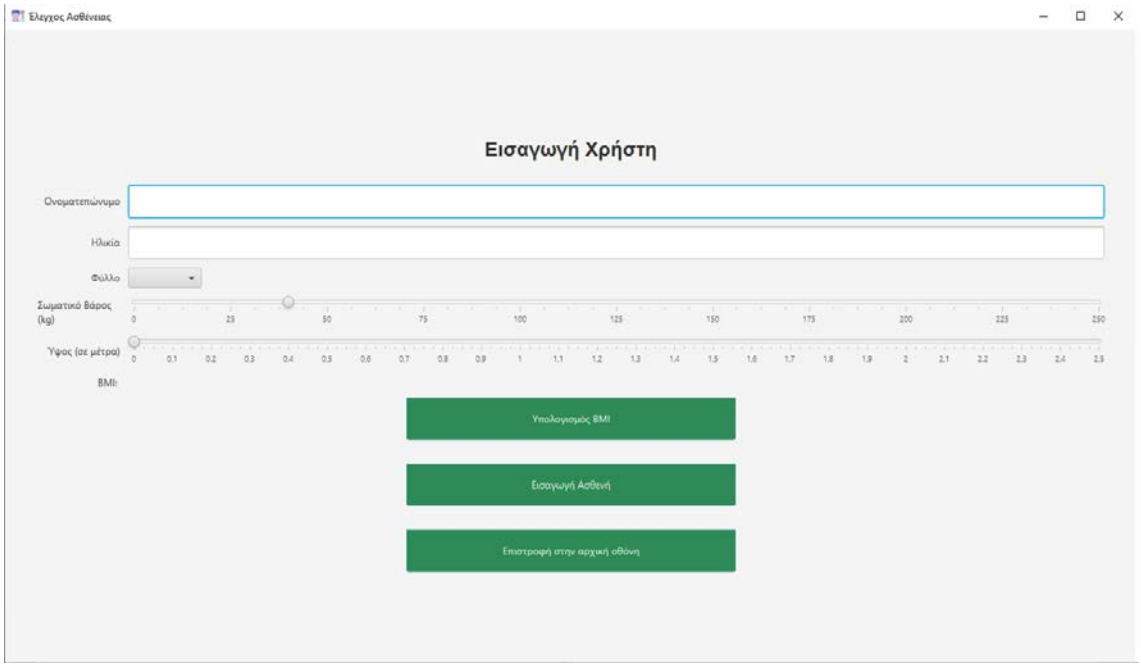
3.3 Παρουσίαση Εφαρμογής



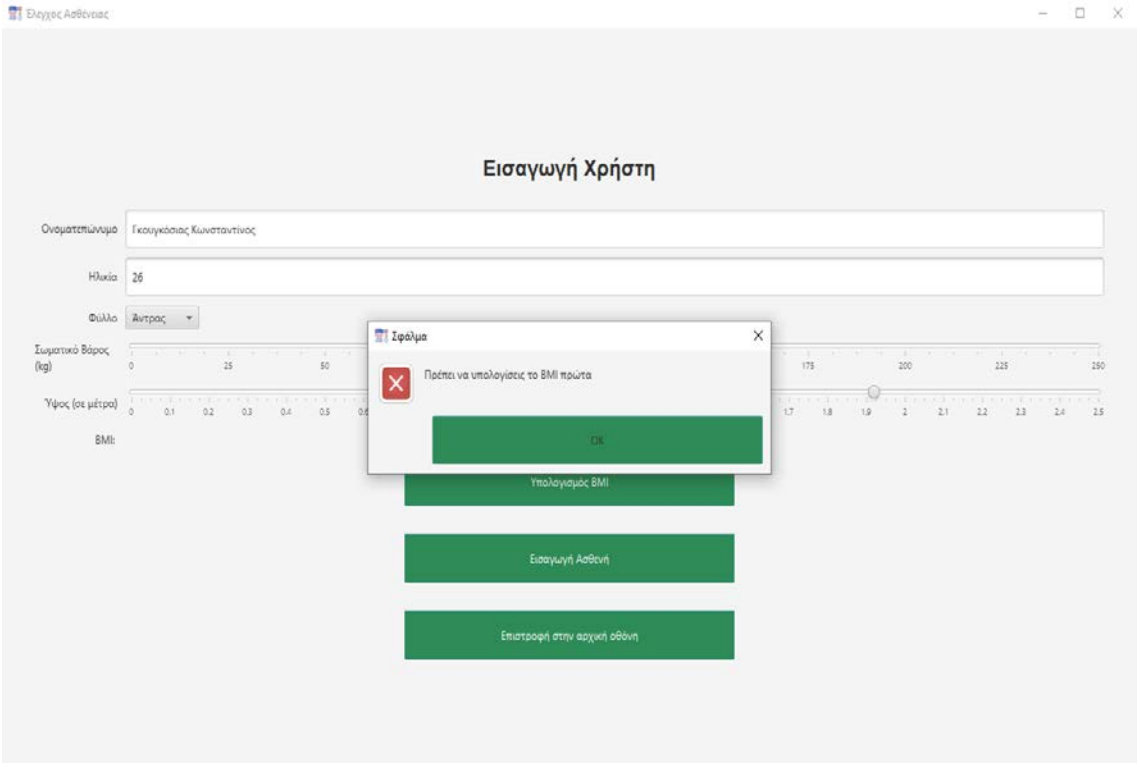
Πρώτη επαφή που θα έχει ο χρήστης με την εφαρμογή θα ειναι το παραπάνω γραφικό περιβάλον της εφαρμογης.



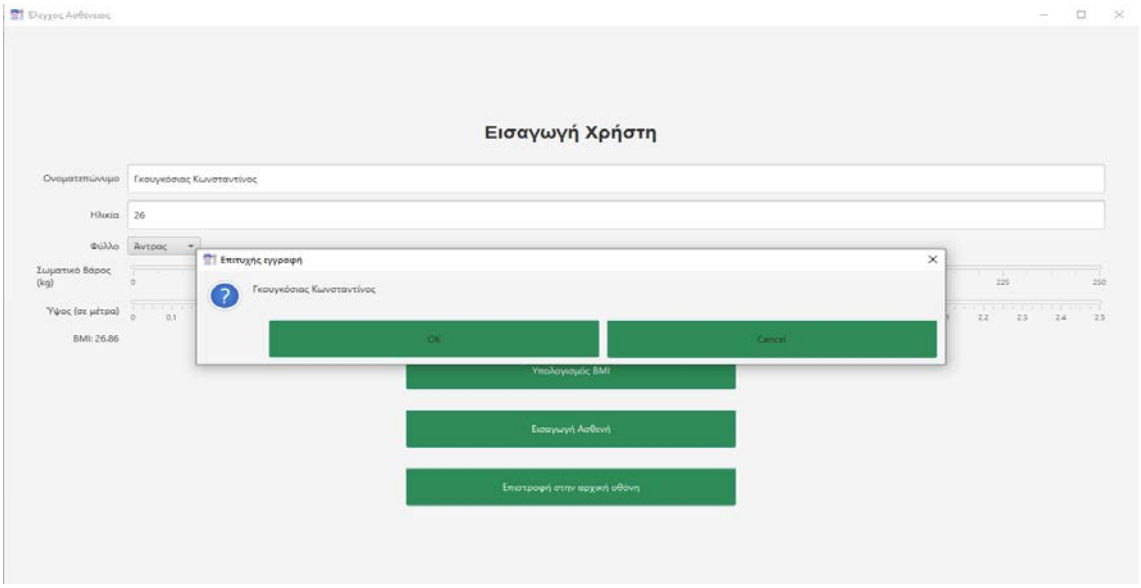
Στην προσπάθεια εισαγωγη συνηθειών,παθήσεων ή συμπτωμάτων προτού την προσθήκη ασθενή θα εμφανίζεται το μήνυμα warning οπως φαινεται στην παραπάνω εικόνα.



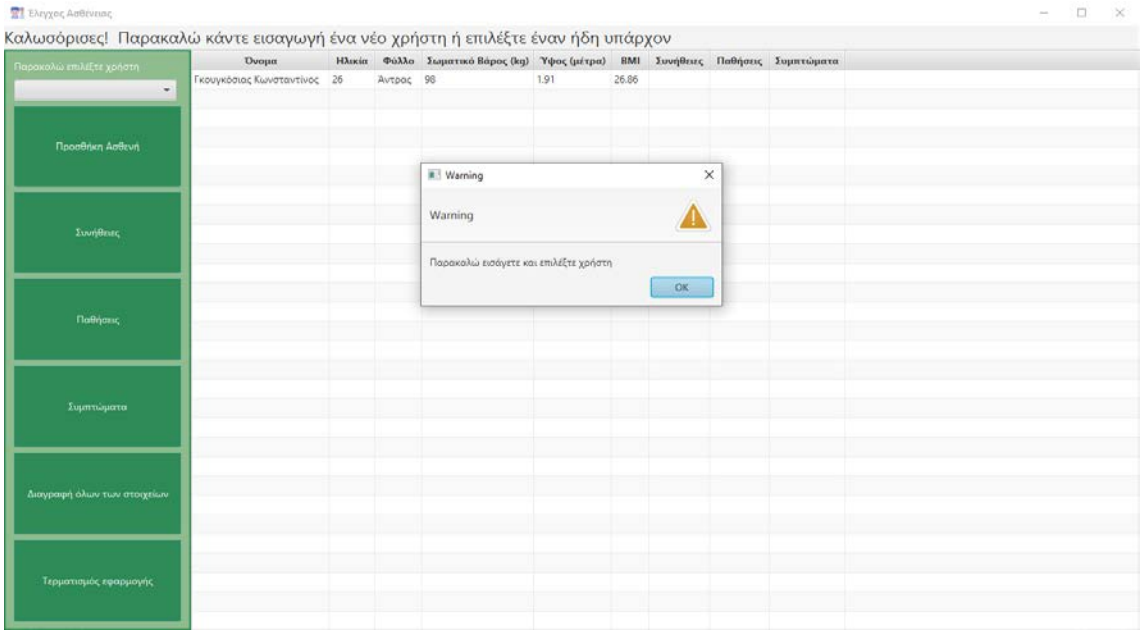
Εδώ ο χρήστης θα χρειαστεί να συμπληρώσει ολα τα στοιχεία του ασθενή.

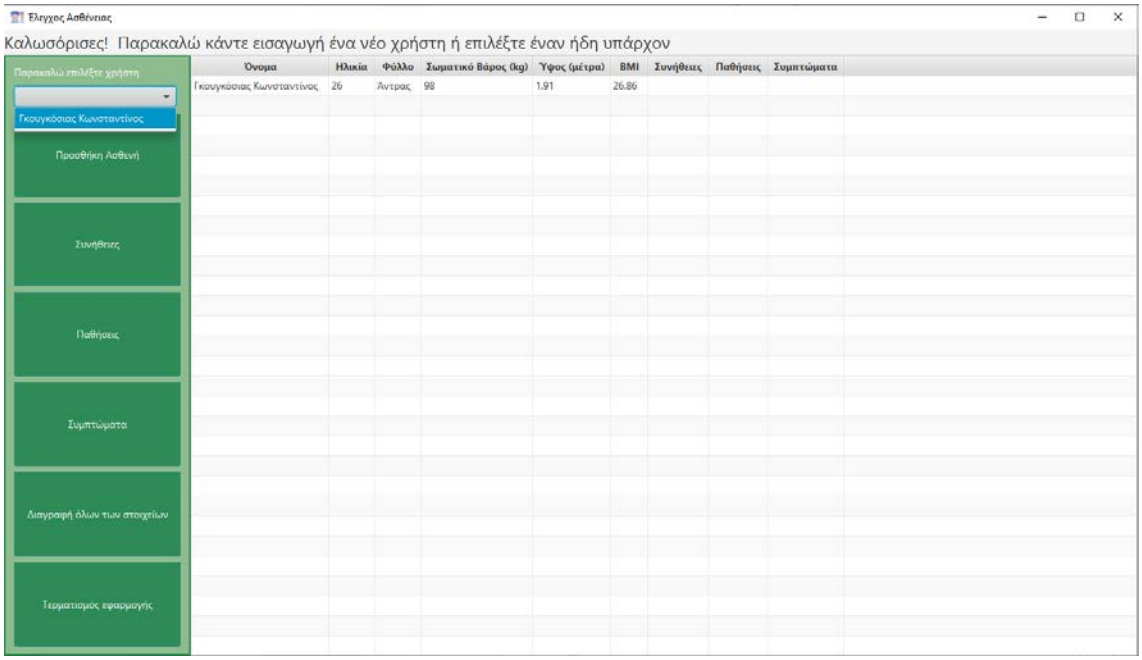


Απαραίτητη προϋπόθεση για να γινει εισαγωγή του ασθενή είναι ο υπολογισμός του BMI. Διαφορετικά θα εμφανίζεται μήνυμα warning οπού θα ενημερώνει τον χρηστή πως χρειάζεται να υπολογιστεί το BMI του ασθενή πρώτα.



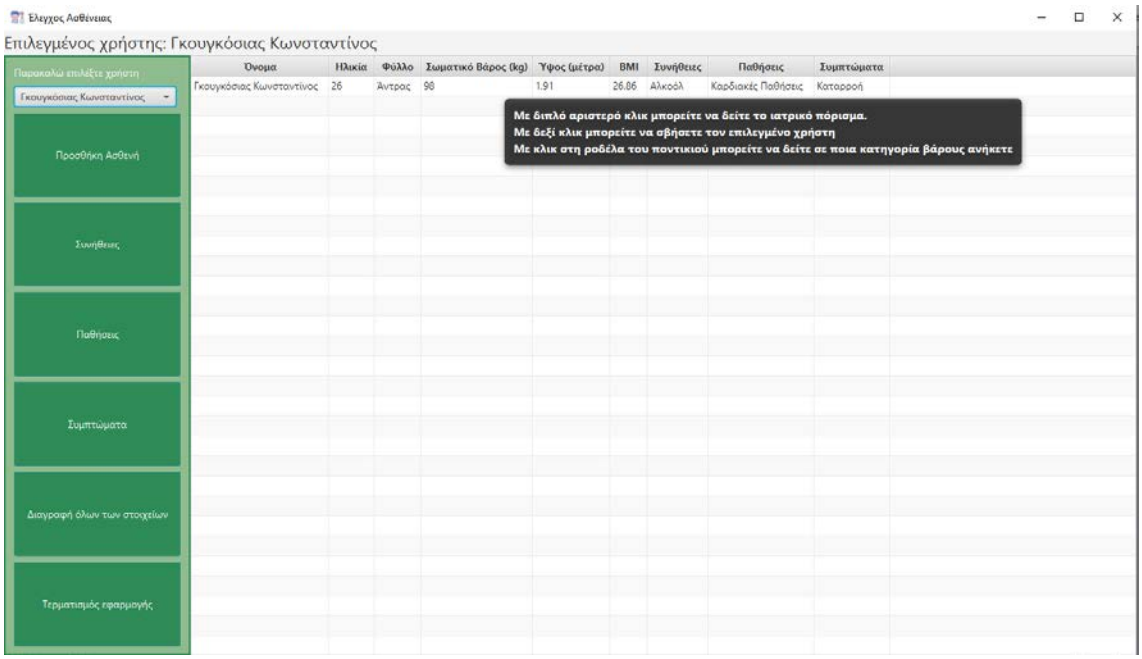
Εφόσον έχουν συμπληρωθεί ολα τα στοιχεία και έχει γίνει υπολογισμός του BMI, πατώντας “προσθήκη ασθενη” θα εμφανίζεται μήνυμα πως η εισαγωγή του νέου ασθενή εχει γίνει με επιτυχεία.





Αφού έχουμε πλεον εισάγη τον νέο ασθενή, επόμενο βήμα ειναι να εισάγουμε τα συμπτώματα, τις παθήσεις και τις συνήθειες του.Για να γίνει όμως αυτο θα πρέπει ο χρήστης να επιλέξει έναν ασθενή απο το combo box διαφορετικά θα εμφανίζεται μήνυμα warning όπου θα ενημερώνει τον χρήστη πως πρέπει να επιλεξεί ασθενή.

Στις παραπάνω εικόνες βλέπουμε ποιές συνήθειες, συμπτώματα και παθήσεις μπορεί να επιλέξει ο χρήστης για τον ασθενή.Επίσης στο κάθε ενα παράθυρο θα εμφανίζοντε μικρά ενημερωτικά μηνύματα όπου θα αλλάζουν ανα λίγα δευτερόλεπτα.



Τώρα πλέον ο χρήστης πατώντας αριστερό κλικ να δεί το ιατρικό πόρισμα του ασθενή που έχει επιλέξει.Επίσης με κλικ της ροδέλας μπορεί να δεί την κατηγορία βάρους στην οποία ανήκει.Τέλος με δεξί κλικ μπορεί να διαγράψει τον ασθενή.

Συμπεράσματα

Ο τελικός στόχος της πτυχιακής εργασίας είχε την ανάπτυξη ενός βασικού ιατροδιαγνωστικού ελέγχου ασθενών με πιθανά συμπτώματα, παθήσεις του οργανισμού και συνήθειες που καταβάλουν την υγεία μας σε κίνδυνο. Το αποτέλεσμα της διάγνωσης παράγεται κατά προσέγγιση και λειτουργεί ως προσομοίωση. Σε καμία περίπτωση δε μπορούν να εξεταστούν με αυτό τον τρόπο όλα τα πιθανά σενάρια ασθένειας.

Αν θέλουμε ρεαλιστικά αποτελέσματα θα χρειαστούμε μία αποθήκη δεδομένων, δηλαδή εκατομμύρια εγγραφές στις οποίες θα έπρεπε να γίνεται στατιστική ανάλυση και εξόρυξη πληροφορίας με ειδικούς αλγορίθμους για να μπορούμε να αποσπάσουμε πραγματικές διαγνώσεις με βάση το input του χρήστη. Αν κάποιος μελλοντικά αποφασίσει να πάει ένα βήμα παραπέρα τη συγκεκριμένη εφαρμογή, θα μπορούσε να γίνει μετατροπή σε web και mobile εφαρμογές κυρίως για την ύπαρξη μίας ενιαίας βάσης δεδομένων στο cloud που θα συγκεντρώνεται όλα τα στοιχεία και τα δεδομένα τον χρηστών σε ένα κεντρικό σημείο και όχι όπως τώρα τοπικά στον υπολογιστή του κάθε χρήστη.

Επίσης, η ανάπτυξη αλγορίθμων εξόρυξης δεδομένων θα μπορούσε να υλοποιηθεί είτε σε cloud level υποδομή είτε σε μορφή backend, έτσι ώστε κατά την εισαγωγή των δεδομένων στην Β.Δ, το cloud ή ο server να αναλάβουν την βαριά εκτέλεση των αλγορίθμων παράγοντας το αποτέλεσμα ταχύτατα σερβίροντας τα στον τελικό χρήστη μέσω του διαδικτύου.

Βιβλιογραφία

[1] http://www.project2061.org/publications/, “Science For All Americans Online.”

[2] https://www.labsexplorer.com/, “Labs Explorer is the place to search for R&D partners, increase your visibility and gain financing.”

[3] https://www.healthdatagateway.org/, Gateway to health data and tools for research.

[4] Braison B. (2019). Το σώμα:Ένας ταξιδιωτικός οδηγός. ΜΕΤΑΙΧΜΙΟ.

[5] Davies T. , Craig T. (2014) , ABC στην ψυχική υγεία , Παρισιάνου Α.Ε.

[6] Deitel P. , (2015) , Java Προγραμματισμός, Γκιούρδας Μ.

[7] Fowler M. (2006) , Εισαγωγή στη UML , Κλειδάριθμος.

[8] Lervik E.,Vergard B. , (2004) , Java με UML Αντικειμενοστραφής σχεδίαση και προγραμματισμός, Κλειδάριθμος.

[9] Parker S. (2008). Το ανρθώπινο σώμα. Ιατρικές εκδόσεις Π. Χ. Πασχαλίδης.

[10] Savitch W. ,(2016) , Java Μια εισαγωγη στην επίλυση προβλημάτων και στον προγραμματισμό, Τζιόλα.

[11] Γκλαβά Μ. (2019) ,Συστήματα βάσεων δεδομένων, Δίσιγμα.

[12] Σταυρακούδης Α. (2015) , Βάσεις δεδομένων και SQL, Κλειδάριθμος.