Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Συστηματων Μεταδοσης Πληροφοριας και Τεχνολογιας Υλικων

Σχεδίαση και Ανάπτυξη εφαρμογής μηχανικής μάθησης για την ανάκτηση πληροφορίας διατροφικών χαρακτηριστικών γευμάτων για κινητές συσκευές Android

Διπλωματικη Εργασια

του

Αλέξης Ορφέας Κουρκάκης

Επιβλέπων: Ιάκωβος Βενιέρης
Καθηγητής

Αθήνα, Νοέμβριος 2021

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Συστημάτων Μετάδοσης Πληροφορίας και Τεχνολογίας Υλικών

Σχεδίαση και Ανάπτυξη εφαρμογής μηχανικής μάθησης για την ανάκτηση πληροφορίας διατροφικών χαρακτηριστικών γευμάτων για κινητές συσκευές Android

Διπλωματικη Εργασια

του

Αλέξης Ορφέας Κουρκάκης

Επιβλέπων: Ιάκωβος Βενιέρης
Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 30η Νοεμβρίου 2020.

(Υπογραφή) *(Υπογραφή)* *(Υπογραφή)*

.....	.....	.....
Ιάκωβος Βενιέρης	Δ. Κακλαμάνη	Γ. Ματσόπουλος
Καθηγητής	Καθηγήτρια	Καθηγητής

Αθήνα, Νοέμβριος 2021

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Συστημάτων Μετάδοσης Πληροφορίας και Τεχνολογίας Υλικών

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάσει επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.
(Υπογραφή)

.....
Αλέξης Ορφέας Κουρκάκης
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών
ΗΜΕΡΟΜΗΝΙΑ ΔΗΛΩΣΗΣ ΠΕΡΙ ΜΗ ΛΟΓΟΚΛΟΠΗΣ 30 Νοεμβρίου 2021

Περίληψη

Η άμεση και αξιόπιστη πρόσβαση σε πληροφορίες σχετικές με τα διατροφικά στοιχεία των τροφών που καταναλώνουμε μπορεί να οδηγήσει στη βελτίωση της ποιότητας ζωής ενός ατόμου. Αυτό φαίνεται καλύτερα στη περίπτωση ατόμων με προβλήματα υγείας, όπως ο διαβήτης, η παχυσαρκία κ.α., αλλά και αυτών που η επαγγελματική τους δραστηριότητα επιβάλλει ειδική διατροφή όπως π.χ. οι αθλητές.

Σκοπός αυτής της διπλωματικής εργασίας είναι η σχεδίαση και ανάπτυξη εφαρμογής μηχανικής μάθησης για κινητές συσκευές Android που θα παρέχει στον χρήστη πληροφορίες για τα διατροφικά χαρακτηριστικά γευμάτων (ενέργεια, υδατάνθρακες, πρωτεΐνες, λίπη, κλπ). Η εφαρμογή θα λαμβάνει σε μορφή απλού κειμένου την περιγραφή του γεύματος. Με τη χρήση αλγορίθμου μηχανικής μάθησης, θα εξάγει τις πληροφορίες από το κείμενο που αφορούν τις διάφορες τροφές και τις ποσότητες τους. Έπειτα, μετά από αίτημα στην βάση δεδομένων FoodData Central του Υπουργείου Γεωργίας των ΗΠΑ, που περιλαμβάνει τα ζητούμενα δεδομένα, θα εμφανίζει στον χρήστη τα διατροφικά στοιχεία για τις επιμέρους ποσότητες των τροφών του συγκεκριμένου γεύματος, αλλά και για το σύνολο του γεύματος.

Επιπρόσθετα, για να δίνεται η δυνατότητα να χρησιμοποιείται η εφαρμογή και offline, τα διατροφικά στοιχεία για τις τροφές που ο χρήστης έχει ήδη αναζητήσει, θα αποθηκεύονται τοπικά σε βάση δεδομένων στη συσκευή του.

Στην πρώτη φάση της διπλωματικής εργασίας, θα παρουσιαστούν οι διαθέσιμες τεχνολογίες. Έπειτα, θα γίνει η ανάλυση των απαιτήσεων της εφαρμογής και ο σχεδιασμός της. Τέλος, θα παρουσιαστεί η ανάπτυξη της εφαρμογής για κινητές συσκευές Android.

Λέξεις Κλειδιά

Εφαρμογή Android, μηχανική μάθηση, ανάκτηση διατροφικών στοιχείων, γεύματα, βάση δεδομένων, διατροφικά στοιχεία

Abstract

Direct and reliable access to nutrition information of meals can improve the quality of one's life. This becomes more evident in the case of people with health problems, such as diabetes, obesity, etc., but also in the case of those whose professional activity requires special diet such that of the athletes.

The purpose of this thesis is to design and develop an application for Android mobile devices that will provide the user with nutrition information of meals (energy, carbohydrates, proteins, fats, etc.). The application will receive a simple text description of the meal. With the use of a machine learning algorithm, it will extract the information related to the foods and the quantities of the meal. Afterwards, upon request in the FoodData Central database by US Department of Agriculture, which contains the requested data, it will display to the user the nutrition information for the given quantities of each food of the specific meal, but also for the whole meal.

In addition, in order to allow offline usage of the application, the nutritional information for the foods that the user has already searched for, will be stored in a local database.

Initially in this thesis, the available technologies for the implementation will be presented. Then, the requirements of the application will be analyzed and it will be designed. Finally, the application for Android devices will be developed and presented.

Keywords

Android application, machine learning, nutrition facts retrieval, meals, database

Ευχαριστίες

Θα ήθελα καταρχήν να ευχαριστήσω τον καθηγητή κ. Βενιέρη για την επίβλεψη αυτής της διπλωματικής εργασίας υπό το Εργαστηρίου Ευφυών Επικοινωνιών και Δικτύων Ευρείας Ζώνης, καθώς και την κα Κακλαμάνη και τον κο Ματσόπουλο που αποτέλεσαν την τριμελή εξεταστική επιτροπή της παρούσας διπλωματικής. Επίσης ευχαριστώ ιδιαίτερα τη Χριστίνα Δαμιανού για την καθοδήγησή του και την εξαιρετική συνεργασία που είχαμε.

Αθήνα, Νοέμβριος 2021

Αλέξης Ορφέας Κουρκάκης

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

1.1 Διατροφή

1.1.1 Η σημασία της διατροφής

Η διατροφή αποτελεί κρίσιμο μέρος της υγείας και της ανάπτυξης ενός ατόμου. Η καλή διατροφή σχετίζεται άμεσα με τη βελτιωμένη υγεία βρεφών, παιδιών και μητέρων, με ισχυρότερο ανοσοποιητικό σύστημα, ασφαλέστερη εγκυμοσύνη και τοκετό. Επίσης, μια υγιεινή διατροφή μπορεί να προστατεύσει το ανθρώπινο σώμα από ορισμένους τύπους ασθενειών, ιδίως χρόνιων μη μεταδοτικών ασθενειών όπως η παχυσαρκία, ο διαβήτης, οι καρδιαγγειακές παθήσεις, ορισμένοι τύποι καρκίνου και σκελετικές παθήσεις. [2].

Μια πλήρης διατροφή απαιτεί τη λήψη και απορρόφηση βιταμινών, μετάλλων, βασικών αμινοξέων από πρωτεΐνες και βασικών λιπαρών οξέων από τρόφιμα που περιέχουν λίπος, καθώς και ενέργεια τροφής με τη μορφή υδατανθράκων, πρωτεϊνών και λίπους. Οι διατροφικές συνήθειες και επιλογές διαδραματίζουν σημαντικό ρόλο στην ποιότητα ζωής, την υγεία και τη μακροζωία. [3].

Οι υγιεινές δίαιτες μπορούν επίσης να συμβάλουν στη σταθεροποίηση του σωματικού βάρους ενώ αποτελούν μια καλή ευκαιρία για δοκιμή και πειραματισμό με ποικιλία τροφίμων από διαφορετικές κουλτούρες, προελεύσεις και με διαφορετικούς τρόπους παρασκευής τροφής. Τα θρεπτικά συστατικά των τροφών ενδυναμώνουν τις σωματικές ικανότητες του ατόμου και ενισχύουν τη δύναμη, ταχύτητα, συγκέντρωση, ευκινησία και συντονισμό. Είναι αυτά που βοηθούν το σώμα να αναπλάθει και να αναγεννά κύτταρα. Ρυθμίζουν επίσης την έκκριση και πετυχαίνουν ισορροπημένη παραγωγή ενέργειας. [4] Επιπλέον, η διατροφή είναι βασικός παράγοντας στη ρύθμιση της διάθεσης καθώς οι υδατάνθρακες συγκεκριμένα είναι αυτοί που επιτρέπουν στην τριπτοφάνη να περνά στον εγκέφαλο, ένα δομικό συστατικό της σεροτονίνης κι απαραίτητο για την παραγωγή της από τον εγκέφαλο. Πρόκειται για ένα χημικό νευροδιαβιβαστή που επηρεάζει την ψυχολογική διάθεση, την ποιότητα του ύπνου και την συναισθηματική ισορροπία, μεταφέροντας μηνύματα και ρυθμίζοντας την επικοινωνία μεταξύ των νευρικών κυττάρων ή των νευρώνων.[5]

1.1.2 Πως αξιολογείται μια διατροφή;

Γνωρίζουμε λοιπόν ότι μια καλή διατροφή βελτιώνει την συνολική ποιότητα ζωής ενός ατόμου. Σε πολλά θέματα της διατροφολογίας οι απόψεις διίστανται. Υπάρχουν όμως συ-

γκεκριμένες κατευθυντήριες γραμμές οι οποίες χρησιμοποιούνται για την αξιολόγηση μιας διατροφής.[6].

- **Βάρος:** Μια καλή διατροφή καταρχάς είναι μια διατροφή η οποία βοηθάει το άτομο να συντηρεί σταθερό και υγιές σωματικό βάρος καθώς και το BMI (Body Mass Index) του σε φυσιολογικά όρια. Παρόλα αυτά το βάρος αποτελεί μόνο ένα παράγοντα από τους πολλούς και δεν είναι ικανός να καθορίσει εξ΄ ολοκλήρου το κατά πόσο μια διατροφή είναι υγιεινή ή όχι.

- **Λίπη:** Όσον αφορά τα λίπη μιας διατροφής, θα πρέπει να κρίνουμε με βάση το ποσοστό της ενέργειας που προέρχεται από αυτά καθώς και τα είδη τους. Τα τελευταία χρόνια, συνιστάται τα λίπη να αποτελούν λιγότερο από 30% της συνολικής ενέργειας ή να παραμένει στα όρια 20-35%. Το είδος των λιπών ωστόσο παίζει πρωταρχική σημασία. Είναι γνωστή η σχέση αίτιου κι αποτελέσματος ανάμεσα στα κορεσμένα λιπαρά και σε αυξημένη χοληστερόλη και καρδιακές παθήσεις. Πρόσφατες έρευνες προτείνουν ότι τα κορεσμένα λιπαρά παίζουν ρόλο και σε παθήσεις όπως ο διαβήτης. Εδώ πρέπει να αναφέρουμε και τα trans λιπαρά τα οποία δε διαφέρουν πολύ και συνδέονται επίσης άμεσα με τον κίνδυνο καρδιακής πάθησης.

- **Υδατάνθρακες:** Οι υδατάνθρακες αποτελούν τη βασική πηγή ενέργειας. Και εδώ έχει μεγάλη σημασία η ποσότητα αλλά και είδος τους. Υγιής τρόπος κατανάλωσης υδατανθράκων είναι οι ολικής άλεσης τροφές οι οποίες σε έρευνες φαίνονται να έχουν αποτρεπτική δράση ενάντια σε καρδιακές παθήσεις, διαβήτη τύπου 2 και καρκίνο. Ωστόσο, σε χαμηλό ποσοστό της τάξεως έως 10% της συνολικής ενεργείας θα πρέπει να παραμένουν τα κατεργασμένα σάκχαρα, των οποίων η υψηλή κατανάλωση συνδέεται με τη παχυσαρκία.

- **Φρούτα και Λαχανικά:** Σε κάθε διατροφή συνιστάται ανεπιφύλακτα μια πλούσια πρόσληψη φρούτων και λαχανικών. Περιλαμβάνουν διαιτητικές (ή φυτικές) ίνες, διάφορα θρεπτικά συστατικά (κάλιο, φολικό οξύ, βιταμίνη C κ.α.) καθώς και φυτοχημικά τα οποία παρέχουν σημαντική προστασία από τον καρκίνο και πολλές άλλες ασθένειες. (Σημείωση: Τα συμπληρώματα διατροφής δεν υποκαθιστούν την κατανάλωση φρούτων και λαχανικών καθώς ένας κύριος λόγος είναι ότι τα συμπληρώματα δεν διαθέτουν φυτοχημικά.)

- **Αλάτι:** Η ποσότητα πρόσληψης αλατιού είναι, επίσης, ένας σημαντικός παράγοντας στην αξιολόγηση μιας διατροφής. Οι τρέχουσες συστάσεις προτείνουν να μειωθεί σε λιγότερο από 6 g (ή 2300 mg νατρίου) ενώ η πρόσληψη του μισού αυτού του επιπέδου είναι ιδιαίτερα επιθυμητή. Ενδεικτικά να αναφέρουμε ότι ο μέσος Αμερικανός καταναλώνει περίπου 9 γραμμάρια αλάτι την ημέρα ενώ έρευνα στην ευρύτερη περιοχή της Θεσσαλονίκης διαπίστωσε ότι μόνο 5.6% του δείγματος κατανάλωνε κάτω από 5 g αλάτι. [7]

1.2 Συνήθεις Παθήσεις και περιπτώσεις όπου απαιτείται παρακολούθηση και χειρισμός διατροφικών συνηθειών

Η διατροφή τις μέρες μας αποτελεί πηγή πόλωσης θεωριών σχετικά με το τι πρέπει να τρώει κάποιος ενώ ταυτόχρονα περιβάλλεται από πολλά προβλήματα. Ορισμένα από αυτά συνδέονται με ανθυγιεινές διατροφικές συνήθειες και γενικότερα τρόπο ζωής του ατόμου, κάποια μπορεί να είναι κληρονομικά όπου ο οργανισμός δε λειτουργεί όπως φυσιολογικά προβλέπεται ενώ κάποια προκύπτουν από ψυχολογικούς παράγοντες. Είτε πρόκειται για ιατρικές παθήσεις είτε για περιπτώσεις όπως αυτές του αθλητισμού, διαφαίνεται η ανάγκη για πρόσβαση σε πληροφορίες διατροφικού περιεχόμενου σε ατομικό επίπεδο και καθημερινή βάση. Παρακάτω βλέπουμε κάποια από τα πιο συχνά φαινόμενα που το απαιτούν.

Διαβήτης

Ο σακχαρώδης διαβήτης αποτελεί συχνή μεταβολική νόσο η οποία χαρακτηρίζεται από υψηλά επίπεδα γλυκόζης (σακχάρου) στο αίμα. Η αυξημένη συγκέντρωση του σακχάρου στο αίμα (υπεργλυκαιμία) και διαταραχή του μεταβολισμού της γλυκόζης εμφανίζεται είτε ως αποτέλεσμα της ελαττωμένης έκκρισης ινσουλίνης είτε λόγω της ελάττωσης της ευαισθησίας των κυττάρων του σώματος στη δράση της ορμόνης αυτής (ινσουλίνη).

Ο διαβήτης τύπου Ι χαρακτηρίζεται από καταστροφή των β-κυττάρων του παγκρέατος, που είναι υπεύθυνα για την παραγωγή ινσουλίνης, με αποτέλεσμα ολική έλλειψη ή ελάχιστη έκκριση ινσουλίνης. Ο διαβήτης τύπου ΙΙ χαρακτηρίζεται από το συνδυασμό μειωμένης έκκρισης ινσουλίνης και ελαττωμένης ευαισθησίας των κυττάρων στη δράση της (φαινόμενο που ονομάζεται ινσουλινοαντίσταση).

Η καταστροφή των β-κυττάρων του παγκρέατος είναι στην πλειοψηφία των περιπτώσεων αυτοάνοσης αιτιολογίας και οι ασθενείς αυτοί είναι απόλυτα εξαρτημένοι από την εξωγενή χορήγηση ινσουλίνης για τη διατήρηση των επιπέδων σακχάρου του αίματος σε φυσιολογικά επίπεδα.

Ο διαβήτης τύπου 2 είναι η συχνότερη αιτία διαβήτη στους ενηλίκους. Σπουδαίος παράγοντας προδιάθεσης για την ανάπτυξη διαβήτη τύπου 2 είναι η παχυσαρκία ενώ άλλοι είναι η ηλικία και το οικογενειακό ιστορικό. Εδώ τα συμπτώματα είναι πιο ήπια και υπάρχει μικρή πιθανότητα εμφάνισης διαβητικής κετοξέωσης (οξεία και απειλητική για τη ζωή επιπλοκή η οποία οφείλεται στην πλήρη ή μερική έλλειψη της ινσουλίνης). Παρ' όλα αυτά, ο κίνδυνος απώτερων και σοβαρών επιπλοκών παραμένει υψηλός. [8]

Σε κάθε τύπο διαβήτη ωστόσο κρίνεται απαραίτητη η παρακολούθηση της κατανάλωσης τροφής και των διατροφικών στοιχείων που εμπεριέχει αυτή, κυρίως των υδατανθράκων. Ο ασθενής καλείται να μάθει να προσαρμόζει τον αριθμό υδατανθράκων που περιέχουν τα γεύματά του αναλογικά με την ινσουλίνη που λαμβάνει. Επιπλέον, μια κατάλληλη διατροφή, όταν ξεκινήσει εγκαίρως, μπορεί να μειώσει τις αυξημένες συγκεντρώσεις γλυκόζης στο αίμα ώστε να επιβραδύνεται η εξάντληση των β-κυττάρων και να προλαμβάνονται οι βλαβερές συνέπειες της υπεργλυκαιμίας. Τέλος, η καταμέτρηση των υδατανθράκων δεν σημαίνει ότι οι μερίδες κρέατος και λίπους μπορούν να αγνοηθούν. Τα άτομα με διαβήτη πρέπει επίσης να γνωρίζουν

τον κατά προσέγγιση αριθμό των μερίδων κρέατος και λίπους που πρέπει να επιλέγουν για τα γεύματα και τα σνακ. Η ιατρική διατροφική θεραπεία λοιπόν εξακολουθεί να αποτελεί σημαντικό συστατικό της διαχείρισης του διαβήτη, αλλά αλλάζει κατά τη φυσική εξέλιξη της θεραπείας της νόσου.

Παχυσαρκία

Η παχυσαρκία ορίζεται και μετράται είτε από αυξημένο σωματικό βάρος, όπως εκφράζεται με το δείκτη μάζας σώματος (ΔΜΣ) = [Σωματικό Βάρος (kg) / (Ύψος)^2(m^2)] , είτε με την περίμετρο της μέσης. Η παθοφυσιολογία της παχυσαρκίας μπορεί να γίνει κατανοητή ως διεύρυνση των λιποκυττάρων, και σε ορισμένα άτομα την αύξηση τους σε αριθμό. Είναι σαφές ότι υπάρχει μια επιδημία παχυσαρκίας που ξεκίνησε τη δεκαετία του 1980 και συνεχίζεται μέχρι σήμερα. Αφορά τόσο παιδιά όσο και ενήλικες. Συνάμα, παρατηρείται αύξηση των περιστατικών του διαβήτη τύπου 2 στους εφήβους που σχετίζονται άμεσα με την παχυσαρκία.

Στην αιτιολογία της παχυσαρκίας παίζουν ρόλο πολλοί γενετικοί, φυσιολογικοί και συμπεριφορικοί παράγοντες όπως και τροφές, φάρμακα, τοξίνες και έλλειψη άσκησης. Η δίαιτα και η άσκηση είναι γνωστό ότι παίζουν πολύτιμο ρόλο στη θεραπεία και την πρόληψη της παχυσαρκίας και των συνδεδεμένων διαταραχών, όπως υπέρταση, καρδιαγγειακές νόσοι και διαβήτης. Στις δυτικές εύπορες κοινωνίες, τα τρόφιμα, ιδιαίτερα τα τρόφιμα με υψηλή περιεκτικότητα σε λιπαρά, είναι άφθονα. [9]

Η παχυσαρκία σχετίζεται με μειωμένη διάρκεια ζωής και συμβάλλει μεταξύ 100.000 έως 400.000 επιπλέον θανάτους ετησίως. Τόσο τα δεδομένα του NCHS όσο και του Framingham έδειξαν ότι ένας ΔΜΣ 30 ή περισσότερο μειώνει τη διάρκεια ζωής κατά 3-5 χρόνια σε σύγκριση με το φυσιολογικό βάρος. Επιπλέον, η καμπυλόγραμμη σχέση σχήματος "J" του ΔΜΣ με τον κίνδυνο επιπλοκών είναι γνωστή για πάνω από 100 χρόνια. Ο συχνότητα περιστατικών διαβήτη καθώς και πολλών ειδών καρκίνου είναι αυξημένος σε παχύσαρκα άτομα.

Τέλος, η παχυσαρκία είναι δαπανηρή. Σε έναν μεγάλο οργανισμό συντήρησης της υγείας, το μέσο ετήσιο κόστος ήταν κατά 25% υψηλότερο στους συμμετέχοντες με ΔΜΣ μεταξύ 30 και 35, και κατά 44% υψηλότερο σε εκείνους με ΔΜΣ μεγαλύτερο από 35, από ό,τι στα άτομα με ΔΜΣ μεταξύ 20 και 25.

Υπέρταση

Η αρτηριακή υπέρταση είναι ένα παγκόσμιο πρόβλημα δημόσιας υγείας. Περίπου 1 δισεκατομμύριο άνθρωποι παγκοσμίως εκτιμάται ότι έχουν κλινικά σημαντική αύξηση της αρτηριακής πίεσης (ΑΠ). Ως φυσιολογική ΑΠ ορίζεται ένα επίπεδο μικρότερο από 120/80 (συστολική/διαστολική σε mmHg). Αρτηριακή υπέρταση λοιπόν είναι μια διαρκώς αυξημένη ΑΠ πάνω από 140/90 mmHg.

Η αρτηριακή υπέρταση συμβάλλει σημαντικά στην ανάπτυξη των περισσοτέρων χρόνιων νοσημάτων και σχετίζεται με αυξημένο κίνδυνο για στεφανιαία νόσο, εγκεφαλικό επεισόδιο και νεφρική νόσο. Οι παθήσεις της καρδιάς και οι εγκεφαλοαγγειακές παθήσεις είναι η πρώτη και η τρίτη κύρια αιτία θνησιμότητας στις Ηνωμένες Πολιτείες, αντιπροσωπεύοντας περισσότερο από το ένα τρίτο όλων των θανάτων.

Στο 95% των περιπτώσεων η αρτηριακή υπέρταση οφείλεται σε ιδιοπαθή αίτια, δηλαδή σε κληρονομικά αίτια, παχυσαρκία, μακροχρόνια αυξημένη πρόσληψη αλατιού, καθιστική ζωή κ.λπ. Συνήθως εμφανίζεται μετά την ηλικία των 30 ετών, αλλά μπορεί να εμφανιστεί σπάνια και σε παιδιά ενώ πολυάριθμες κλινικές παρεμβάσεις διαπιστώνουν ότι και στα δύο φύλα η υπέρταση είναι περίπου διπλάσια στους παχύσαρκους από ό,τι στους μη παχύσαρκους.

Ωστόσο υπάρχουν πολλές διατροφικές αλλαγές οι οποίες έχουν σημαντικό αντίκτυπο στην ΑΠ. Παράγοντες όπως η μειωμένη κατανάλωση νατρίου, η αυξημένη πρόσληψη καλίου, τα πολυακόρεστα λίπη, χορτοφαγικές διατροφές, διατροφές υψηλές σε περιεκτικότητα φρούτων, λαχανικών και γαλακτοκομικών με χαμηλά λιπαρά συνδέονται με χαμηλότερα επίπεδα ΑΠ. Αντιθέτως, δίαιτες με υψηλή περιεκτικότητα σε χοληστερόλη και κορεσμένα λιπαρά καθώς και η αυξημένη κατανάλωση αλκοόλ φαίνεται να αυξάνουν τη συστολική και διαστολική ΑΠ.

Διαταραχές Πρόσληψης Τροφής

Οι διατροφικές διαταραχές αντιπροσωπεύουν ακραίες καταστάσεις στη διατροφή. Αυτά τα άκρα του υποσιτισμού και της υπερβολικής πρόσληψης τροφής μπορούν να υπάρχουν στο ίδιο άτομο ταυτόχρονα. Υπάρχουν πολλές μορφές διατροφικών διαταραχών όπως νευρική βουλιμία, νευρική ανορεξία, Υπερφαγία, Ψυχαναγκαστική Υπερφαγία (Binge-Eating Disorder), Νυχτερινό Σύνδρομο Υπερφαγίας. Γενικότερα σ΄ όλες τις διατροφικές διαταραχές παρατηρείται μεγάλη δυσαρέσκεια για το σωματικό βάρος και το σχήμα.[10] Στις περισσότερες διαγνώσεις, υπάρχει επίσης μια ανεξέλεγκτη ώθηση για φαγητό. Οι πιο συνήθεις είναι η νευρική ανορεξία και νευρική βουλιμία.

- **Νευρική Ανορεξία:** Η νευρική ανορεξία περιλαμβάνει 3 βασικά χαρακτηριστικά: άρνηση διατήρησης ενός ελάχιστου φυσιολογικού σωματικού βάρους για την ηλικία και το ύψος, έντονο φόβο αύξησης του βάρους παρά το γεγονός ότι ο/η πάσχων/ούσα είναι λιποβαρής και διαστρέβλωση στον τρόπο με τον οποίο αντιλαμβάνονται το σωματικό τους βάρος και το σχήμα ή μια άρνηση της σοβαρότητας της κατάστασης. Οι αντιλήψεις τους για το σωματικό βάρος και το σχήμα τους καθορίζει επίσης σχεδόν εξ΄ ολοκλήρου την αυτοεκτίμησή τους. Τέλος, υπάρχουν δύο υποτύποι της νευρικής ανορεξίας, ο περιοριστικός υποτύπος και αυτός της υπερφαγίας/κάθαρσης.

  Σχεδόν κάθε σωματικό σύστημα επηρεάζεται αρνητικά από τη νευρική ανορεξία. Αυτό οφείλεται στην ασιτία και, όταν υπάρχει, στις επιπτώσεις της κάθαρσης. Οι ανωμαλίες που προκύπτουν περιλαμβάνουν βραδυκαρδία, αρρυθμία, υποθυρεοειδισμό, χαμηλή οστική πυκνότητα, δυσκοιλιότητα, υπογονιμότητα και περιγεννητικές επιπλοκές. Ο όγκος της φαιάς ουσίας στον εγκέφαλο μειώνεται. Ταυτόχρονα, τα ατροφικά νευρωνικά δίκτυα μπορεί να διατηρούν τις ψυχολογικές ψευδαισθήσεις των πασχόντων σχετικά με τους φόβους τους για το λίπος και τις πεποιθήσεις ότι δεν είναι αρκετά αδύνατοι, καθώς και τις εμμονές και τις καταναγκαστικές τελετουργίες με το φαγητό. Η καρδιακή λειτουργία μπορεί επίσης να είναι κακή σε αυτό το σημείο. Για αυτούς τους λόγους, μαζί με τα υψηλά ποσοστά αυτοκτονιών, η νευρική ανορεξία θεωρείται η πιο θανατηφόρα ψυχιατρική διαταραχή.

- **Νευρική Βουλιμία:** Τα βασικά χαρακτηριστικά της νευρικής βουλιμίας είναι η υ-

περφαγία και η επακόλουθη χρήση ακατάλληλων αντισταθμιστικών συμπεριφορών. Οι συμπεριφορές αυτές χρησιμοποιούνται σε μια προσπάθεια να επιτευχθεί χαμηλό σωματικό βάρος ή να αποτραπεί η αύξηση του βάρους. Όπως και στη νευρική ανορεξία, υπάρχει αδικαιολόγητη επιρροή του βάρους και του σχήματος στην αυτοαξιολόγηση και την αυτοεκτίμηση. Κι εδώ υπάρχουν δύο υποτύποι της νευρικής βουλιμίας: ο τύπος της κάθαρσης και αυτός της μη-κάθαρσης. Οι συμπεριφορές κάθαρσης συνίστανται συχνότερα σε εμετό, που χρησιμοποιείται στο 80-90% των περιπτώσεων, ακολουθούμενη από κατάχρηση καθαρτικών.

Τα περισσότερα άτομα με νευρική βουλιμία έχουν ΔΜΣ σε υγιή όρια. Οι περισσότερες ιατρικές επιπλοκές σε αυτή τη διαταραχή προκαλούνται από τις συμπεριφορές κάθαρσης. Ενώ αυτές οι ιατρικές επιπλοκές δεν είναι τόσο σοβαρές όσο αυτές που παρατηρούνται στη νευρική ανορεξία, τα άτομα με νευρική βουλιμία είναι γενικά λιγότερο ανεκτικά στα σωματικά τους συμπτώματα ενώ συχνά αισθάνονται ενοχές και ντροπή που σχετίζονται με τις συμπεριφορές υπερφαγίας και κάθαρσης.

Θεραπείες και τρόποι αντιμετώπισης για τη νευρική ανορεξία που να έχουν και μακροπρόθεσμη αποτελεσματικότητα εξακολουθούν να υστερούν. Για ασθενείς συχνά δικαιολογείται νοσηλεία σε νοσοκομείο για επανασίτιση και θεραπεία σε ιδρύματα. Στις θεραπείες περιλαμβάνονται επίσης θεραπευτικά γεύματα, όπου οι ασθενείς καλούνται να καταναλώνουν διατροφικά ισορροπημένα γεύματα και σνακ σε τακτά χρονικά διαστήματα κάθε μέρα. Για τη θεραπεία της νευρικής βουλιμίας εφαρμόζεται αρκετά αποτελεσματικά ψυχοθεραπευτική προσέγγιση ασθενών γνωσιακής συμπεριφορικής κατεύθυνσης, η οποία βοηθάει την αποχή από τις συμπεριφορές υπερφαγίας και κάθαρσης. Χρήσιμη κρίνεται επίσης και στη θεραπεία της ενεργού νευρικής ανορεξίας καθώς και τη διατήρηση των επιτευγμάτων. Τέλος, δε λείπουν και τα αντικαταθλιπτικά που συνταγογραφούνται ευρέως γιατί είναι αποτελεσματικά στη μείωση των συμπεριφορών υπερφαγίας και κάθαρσης και βελτιώνουν τα συνυπάρχοντα συμπτώματα διάθεσης και άγχους.

Καρδιακές Παθήσεις

Οι καρδιακές παθήσεις μαστίζουν το δυτικό κόσμο με την στεφανιαία νόσος (ΣΝ) να αποτελεί μείζονα αιτία νοσηρότητας και θνησιμότητας. Παράγοντες που συνδέονται στενά με αυξημένο κίνδυνο εμφάνισης ΣΝ είναι η αυξημένη ηλικία, το ανδρικό φύλο, το κάπνισμα, η έλλειψη άσκησης, η παχυσαρκία, η υπέρταση και ο διαβήτης τύπου 2. Επιπλέον, τα επίπεδα των λιπιδίων του αίματος αποτελούν ισχυρούς προγνωστικούς παράγοντες του κινδύνου ΣΝ. Υποστηρίζεται σθεναρά ότι η διατροφή παίζει ένα σημαντικό ρόλο στην πρόληψη και τη θεραπεία της ΣΝ. Οι διατροφικοί παράγοντες που επηρεάζουν τον κίνδυνο εμφάνισης ΣΝ περιλαμβάνουν κορεσμένα λιπαρά οξέα, trans λιπαρά οξέα, πολυακόρεστα λιπαρά οξέα, φυτικές ίνες, βιταμίνες του συμπλέγματος Β και αντιοξειδωτικές βιταμίνες. Είναι πλέον αποδεκτό ότι επηρεάζουν την αιτιολογία της ΣΝ μέσω πολλαπλών μηχανισμών, συμπεριλαμβανομένης της αντίστασης στην ινσουλίνη, της αρτηριακής πίεσης, της ενδοθηλιακής λειτουργίας, της φλεγμονής και της θρόμβωσης.

Με βάση τα ισχυρότερα στοιχεία που υπάρχουν σήμερα, μπορούμε να συνοπτικά να πούμε ότι οι δίαιτες με χαμηλή περιεκτικότητα σε κορεσμένα και trans λιπαρά και με άφθονες

ποσότητες φρούτων, λαχανικών, δημητριακών ολικής αλέσεως και τροφίμων που παρέχουν n-3 πολυακόρεαστα, είναι ιδιαίτερα προστατευτικές έναντι της ΣΝ. Συνάμα, η διατήρηση ενός υγιούς σωματικού βάρους και η τακτική άσκηση μειώνουν επίσης τον κίνδυνο εμφάνισης ΣΝ. Όσον αφορά άλλες καρδιαγγειακές παθήσεις, όπως το εγκεφαλικό επεισόδιο, μπορούμε να συμπεράνουμε ότι, ενώ η σχετική σημασία των διαφόρων παραγόντων κινδύνου ποικίλλει από τη μία μορφή καρδιαγγειακής νόσου στην άλλη, οι παραπάνω συστάσεις θα συμβάλουν σε μεγάλο βαθμό στην επίτευξη της πρόληψης όλων των καρδιαγγειακών παθήσεων.

Λοιπές περιπτώσεις

Βλέπουμε λοιπόν ότι η ανάγκη για ρύθμιση της διατροφής διαφαίνεται σε πολλές ιατρικές περιπτώσεις. Πέραν των προαναφερθέντων, απαραίτητη είναι και σε γαστρεντερικές παθήσεις, παθήσεις του παγκρέατος, στη χρόνια νεφρική ανεπάρκεια καθώς και άλλες μη-ιατρικές όπως στον αθλητισμό και την επιθυμία του ατόμου για υγιεινή διατροφή.

Σχετικά με τον αθλητισμό, η διατροφή μπορεί να συμβάλλει στην καλή ή κακή απόδοση ενός αθλητή και συχνά υποστηρίζεται ότι είναι εξίσου σημαντική με την προπόνηση. Η σημασία της αντανακλάται σε όλα τα επίπεδα του αθλητισμού. Μπορεί να προστατεύσει τον αθλητή από την εξάντληση των αποθεμάτων γλυκογόνου, την υπογλυκαιμία, την ¨κεντρική κόπωση' μέσω νευρικών διαβιβαστών, την υπερθερμία, την αφυδάτωση κ.α. Φροντίζει ώστε το σώμα να μπορεί να ανταποκριθεί στις υψηλές απαιτήσεις μιας προπόνησης ή ενός αγώνα.

1.3 Τεχνολογική εξέλιξη

1.3.1 Smartphones

Τα smartphones έχουν υιοθετηθεί ταχύτερα από οποιαδήποτε άλλη τεχνολογία στην ιστορία της ανθρωπότητας. Έχουν γίνει τόσο αναπόσπαστο κομμάτι της καθημερινής ζωής που οι περισσότεροι χρήστες δηλώνουν ότι δεν μπορούν να φανταστούν τη ζωή χωρίς. Όπως μας πληροφορεί το Pew Research Center, η συντριπτική πλειοψηφία των πολιτών των ΗΠΑ - 85% - κατέχει σήμερα ένα smartphone (2021), από μόλις 35% στην πρώτη έρευνα για την κατοχή smartphone που διεξήχθη το 2011.[11] Σε έρευνα που πραγματοποιήθηκε στην Ελλάδα το 2019 το ποσοστό του πληθυσμού που κατείχε smartphone ήταν 59%, νούμερο που εκτιμάται να έχει αυξηθεί δραματικά τα τελευταία δύο χρόνια ενώ την ίδια χρονιά ο μέσος ενήλικας στις ΗΠΑ αφιέρωσε 3 ώρες και 43 λεπτά την ημέρα σε κινητές συσκευές, χρόνος που για πρώτη χρονιά ξεπέρασε τον χρόνο που αφιερώνεται στην τηλεόραση. [12]

Πρόκειται για φορητούς προσωπικούς υπολογιστές που συνδυάζουν τη λειτουργικότητα ενός συμβατικού κινητού τηλεφώνου με συνδεσιμότητα δικτύου και την εγκατάσταση εφαρμογών λογισμικού. Πλέον, όλοι γνωρίζουμε τις δυνατότητες που μας παρέχουν. Η συνδεσιμότητα τύπου ¨οποτεδήποτε/οπουδήποτε' που παρουσιάζουν τα smartphones σημαίνει ότι μπορούμε να έχουμε πρόσβαση σε ποιοτικά δεδομένα και σε όλων των ειδών τα κανάλια πληροφοριών πρακτικά συνέχεια και εύκολα απ΄ τις φορητές μας συσκευές. Οι άνθρωποι είναι σε θέση να αναζητήσουν άγνωστα μέχρι τώρα εδάφη και να διανείμουν γνώση δίχως περιορισμούς. Ο εκπληκτικός μετασχηματισμός των ανθρώπινων κοινωνιών μπορεί να απεικονιστεί από την πανταχού παρούσα πρόσβαση σε ψηφιακούς κόσμους και τη δυνατότητα να επικοινωνούμε με

εύκολους τρόπους σε μεγάλες αποστάσεις σχεδόν χωρίς κόστος. Αυτή τη στιγμή, η κοινωνία μας εξελίσσεται ραγδαία προς έναν κόσμο πλήρως συνδεδεμένο με το Διαδίκτυο, όπου τα πάντα, από μια κλασική οικιακή συσκευή όπως η ηλεκτρική σκούπα μέχρι τον επιτραπέζιο υπολογιστή, αποτελούν μέρος του Internet of Things - (IoT).[13] Αυτή η συνδεσιμότητα λοιπόν καθιστά δυνατή την δημιουργία και την ενσωμάτωση στην καθημερινή ζωή εύχρηστων και καθόλα χρήσιμων εφαρμογών που βελτιώνουν την ποιότητα ζωής του χρήστη σε ¨πραγματικόχρόνο¨. Τα δεδομένα είναι προσβάσιμα φαινομενικά παντού και αμέσως.

1.3.2 Βάσεις Δεδομένων

Μια βάση δεδομένων είναι μια δομημένη συλλογή συγκεντρωμένων, συγκρίσιμων δεδομένων που μπορούν να ταξινομηθούν και να προσπελαστούν για κάποιο σκοπό παραγωγής γνώσης. Ένα κοινό χαρακτηριστικό όλων των βάσεων δεδομένων - και ίσως το κύριο - είναι ότι επιτρέπουν την σύνδεση, κάτω από νέα πλαίσια, μεταξύ πληροφοριών που προέρχονται από πολλαπλές, συχνά ετερογενείς πηγές. Τις τελευταίες δεκαετίες, τα μεγέθη, οι δυνατότητες και οι επιδόσεις των βάσεων δεδομένων και των αντίστοιχων συστημάτων διαχείρισής τους έχουν αυξηθεί σε τάξεις μεγέθους.

Ουσιαστικά, οποιαδήποτε πληροφορία μπορεί να τεθεί σε μορφή πίνακα μπορεί να αναπτυχθεί σε μια βάση δεδομένων. [14] Οι δυνατές χρήσεις είναι αμέτρητες, αφού οι βάσεις δεδομένων δεν είναι απλώς εργαλεία για την αποθήκευση και επεξεργασία μεγάλου όγκου δεδομένων: είναι επίσης συστήματα ορθολογικής οργάνωσης που δημιουργούν κατηγορίες στα δεδομένα, ορίζουν σχέσεις και υπαγορεύουν δομή στις αντιλήψεις μας για την πραγματικότητα. Δημιουργούν έτσι νέες οπτικές στη ζωή και μας βοηθάνε να κατανοήσουμε εις βάθος και να βγάλουμε συμπεράσματα. [15] Καθώς η χρήση μεθόδων στατιστικής εδραιώθηκε κατά το δεύτερο μισό του δέκατου ένατου αιώνα, εμφανίστηκαν νέα είδη ανάλυσης και πρόβλεψης για μεγάλα σύνολα δεδομένων ενώ οι βάσεις δεδομένων έγιναν κεντρικό εργαλείο για τις κυβερνήσεις και τη ρύθμιση της ζωής των πολιτών καθώς και για ιδιωτικά συμφέροντα (ασφαλιστικές εταιρείες, τράπεζες κ.α.). Έτσι, οδηγηθήκαμε στη δημιουργία τεράστιων ηλεκτρονικών βάσεων δεδομένων που αφορούν κάθε πτυχή της ζωής μας στις οποίες δύναται να έχουμε πρόσβαση ακόμα και μέσω των smartphones.

Με αυτά κατά νου, δηλαδή τον εκτενή όγκο δεδομένων τα οποία μπορούμε να αποθηκεύσουμε και να οργανώσουμε, καθώς και την αμεσότητα με την οποία μπορούμε να τα προσπελάσουμε, μια διαδικασία σαν την αναζήτηση διατροφικών στοιχείων τροφών σε ατομικό επίπεδο καθίσταται αποδοτική και χρήσιμη. Συγχρόνως, όπως είδαμε στο 1.1, η πρόσβαση σε πληροφορίες για τις τροφές που καταναλώνουμε αποτελεί αναγκαιότητα και είναι σε ορισμένες περίπτωσης καίριας σημασίας. Αν αναλογιστούμε και τα σχετικά μέσα που διαθέτουμε, θα λέγαμε ότι επιβάλλεται η υλοποίηση ενός κατάλληλου εργαλείου.

1.4 Αντικείμενο της Διπλωματικής

Προκύπτει λοιπόν η ανάγκη να ¨ξέρουμε τι τρώμε¨, να γνωρίζουμε αναλυτικά τα διατροφικά στοιχεία των τροφών που καταναλώνουμε ώστε να μπορούμε να ανταπεξερχόμαστε σε διάφορες ιατρικές καταστάσεις. Ακόμα, και για άτομα που δε βρίσκονται σε μια απ' αυτές τις

καταστάσεις αλλά επιθυμούν να παρακολουθήσουν τα γεύματα τους ώστε να διαμορφώσουν μια συγκεκριμένη διατροφή, η εύκολη κι άμεση πρόσβαση είναι ίσως το πιο μεγάλο βοήθημα για το στόχο τους. Τα δεδομένα που αποζητούν είναι ήδη προσβάσιμα αλλά όχι άμεσα, εύκολα και αξιόπιστα στο μέσο χρήστη μέσα από το περιβάλλον μιας εφαρμογής στο κινητό του. Στην ¨εποχή των apps' για κάθε ανάγκη ή στόχο υπάρχει και η αρμόδια εφαρμογή για την κάλυψη της.

Αντικείμενο αυτής της διπλωματικής εργασίας είναι η δημιουργία μιας τέτοιας εύχρηστης εφαρμογής μηχανικής μάθησης για περιβάλλον Android με σκοπό την εξυπηρέτηση αυτής της ανάγκης. Πρόκειται για μια εφαρμογή που θα δίνει τη δυνατότητα στο χρήστη να πληκτρολογεί το σύνολο των τροφών που κατανάλωσε σε μορφή πρότασης. Στη συνέχεια, η εφαρμογή, με τη χρήση μηχανικής μάθησης, θα αναγνωρίζει τις τροφές και τις αντίστοιχες ποσότητες και θα εξάγει αυτή την πληροφορία. Έπειτα για κάθε μια από αυτές ο χρήστης θα μπορεί να πραγματοποιήσει αίτημα και να δει τις σχετικές διατροφικές πληροφορίες που υπάρχουν στη βάση δεδομένων FoodData Central του U.S. Department Of Agriculture (USDA).

Η παρούσα διπλωματική εργασία έχει ως σκοπό τη διευκόλυνση της καθημερινότητας του μέσου χρήστη Android που χρειάζεται να γνωρίζει τα διατροφικά στοιχεία που καταναλώνει. Μέσω της εφαρμογής αυτής, επιδιώκεται η βελτίωση της ποιότητας ζωής του χρήστη καθώς μπορεί να διαχειριστεί καλύτερα δύσκολες καταστάσεις και να ελέγχει αναλυτικά τη διατροφή που ακολουθεί. Σκοπός είναι τα δεδομένα να εμφανίζονται αναλυτικά και δυναμικά κάθε φορά με βάση επακριβώς το κάθε γεύμα, καλύπτοντας τις βασικές καθημερινές ανάγκες για πληροφορία.

1.4.1 Δομή και Οργάνωση

Η εργασία αυτή χωρίζεται σε έξι κεφάλαια: Στο Κεφάλαιο 2 παρουσιάζονται οι βασικές τεχνολογίες και εργαλεία που χρησιμοποιήθηκαν στην διπλωματική αυτή. Στο Κεφάλαιο 3, γίνεται η βασική ανάλυση των απαιτήσεων της εφαρμογής και των σεναρίων χρήσης. Στο Κεφάλαιο 4, παρουσιάζεται η σχεδίαση της εφαρμογής από την αρχιτεκτονική της μέχρι τη δομή των οθονών και το συνδυασμό των διαφόρων τεχνολογιών. Στο Κεφάλαιο 5, παρουσιάζεται η ανάπτυξη της εφαρμογής και βασικά μέρη του κώδικα. Στο Κεφάλαιο 6, γίνεται η αποτίμηση κι ο έλεγχος της εφαρμογής μέσω των δύο βασικών παραδειγμάτων χρήσης. Τέλος, στο Κεφάλαιο 7 παρουσιάζονται συμπεράσματα που βγήκαν από την εκπόνηση αυτής της διπλωματικής εργασίας και προτείνονται κάποιες πιθανές μελλοντικές επεκτάσεις της εφαρμογής.

Κεφάλαιο 2

Τεχνολογίες

Στο κεφάλαιο αυτό παρουσιάζονται αναλυτικά οι βασικές τεχνολογίες καθώς και τα περιβάλλοντα που έχουν σχέση με την εργασία αυτή.

2.1 Android OS

Το Android είναι ένα λειτουργικό σύστημα open-source βασισμένο στο Linux και προορίζεται κυρίως για κινητές συσκευές. Η ιστορία του Android ξεκινά τον Οκτώβριο του 2003, αλλά η απόφαση-κλειδί στην ιστορία του ήταν η δέσμευση της Google να το καταστήσει ανοικτού κώδικα, γεγονός που του επέτρεψε να γίνει δημοφιλές σε τρίτους κατασκευαστές τηλεφώνων.

Η δημόσια δοκιμαστική έκδοση (beta) του Android 1.0 είδε το φως για τους προγραμματιστές στις 5 Νοεμβρίου 2007 μαζί με το σχηματισμό της λεγόμενης Open Handset Alliance, στην οποία συμμετείχαν κατασκευαστές τηλεφώνων και υλικού καθώς και εταιρείες λογισμικού ενώ τον Σεπτέμβριο του 2008 ανακοινώθηκε το πρώτο smartphone Android. Έκτοτε, σχεδόν κάθε χρόνο το λειτουργικό σύστημα Android ανανεώνεται και ενημερώνεται με νέες εκδόσεις με τη Google να ονοματίζει κάθε εκδόση με τον αριθμό της και δίπλα ένα γλυκό ή επιδόρπιο (Oreo, Lollipop, Cupcake κ.λπ.). Η παράδοση αυτή σταμάτησε το 2019 με την έκδοση του Android 10.

Εξακολουθώντας να είναι αφοσιωμένη στην περαιτέρω εξέλιξη του, η Google κατάφερε να είναι το Android σήμερα το κορυφαίο λειτουργικό σύστημα κινητών τηλεφώνων παγκοσμίως, με μερίδιο αγοράς περίπου 75%. Το λειτουργικό σύστημα χρησιμοποιείται από τηλέφωνα που πωλούνται λιγότερο από 100$ μέχρι και ακριβές συσκευές ναυαρχίδες που κοστίζουν πολύ περισσότερο από 1.000$. Αυτή η ευελιξία, σε συνδυασμό με τις ετήσιες ενημερώσεις, μας υπόσχεται ότι το Android θα παραμείνει ο ηγέτης στον κλάδο για τα επόμενα χρόνια. [16]

2.2 Android Studio

Το Android Studio είναι το επίσημο ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) για την ανάπτυξη εφαρμογών Android. Βασίζεται στο IntelliJ IDEA και ενσωματώνει εργαλεία επεξεργασίας κώδικα και ανάπτυξης. Ανακοινώθηκε για πρώτη φορά στο Google I/O τον Μάιο του 2013 και το πρώτο σταθερό build κυκλοφόρησε τον Δεκέμβριο του 2014. Το Android Studio είναι διαθέσιμο για τις πλατφόρμες Mac, Windows και Linux. [17]

Για την υποστήριξη της ανάπτυξης εφαρμογών στο πλαίσιο του λειτουργικού συστήματος Android, το Android Studio χρησιμοποιεί ένα "build" σύστημα βασισμένο στο Gradle, emulator, πρότυπα κώδικα και ενσωμάτωση του Github. Κάποια βασικά χαρακτηριστικά είναι τα εξής:

- Ο Android Emulator εγκαθιστά και εκκινεί εφαρμογές ταχύτερα από μια πραγματική συσκευή και επιτρέπει την πρωτοτυποποίηση και τον έλεγχο της εφαρμογής σε διάφορα είδη συσκευών Android: τηλέφωνα, tablet, Android Wear και συσκευές Android TV. Μπορεί επίσης να προσομοιώσει μια ποικιλία χαρακτηριστικών υλικού, όπως τοποθεσία GPS, latency δικτύου, αισθητήρες κίνησης και πολλαπλή αφή.
- Πρότυπα ολόκληρων project και κώδικα που διευκολύνουν την προσθήκη ήδη καθιερωμένων προτύπων, όπως ένα navigation drawer και ένα view pager.
- Εκτενή εργαλεία που βοηθούν στη δοκιμή εφαρμογών Android με το JUnit 4 και με λειτουργικά πλαίσια ελέγχου UI. Με το Espresso Test Recorder, μπορεί κανείς να δημιουργήσει δοκιμαστικό κώδικα UI καταγράφοντας τις αλληλεπιδράσεις με την εφαρμογή σε μια συσκευή ή έναν εξομοιωτή.
- Ένα drag-and-drop visual editor που κάνει πιο εύκολη από ποτέ τη δημιουργία ενός νέου layout όταν εργαζόμαστε με αρχεία XML.[18]

2.3 Kotlin

Η Kotlin είναι μια ελεύθερη, ανοικτού κώδικα γλώσσα προγραμματισμού γενικού σκοπού με στατικό σύστημα τύπων. Είναι αντικειμενοστραφής και υποστηρίζει λειτουργίες συναρτησιακού προγραμματισμού ενώ φτιάχτηκε για να λειτουργεί ταυτόχρονα, πλήρως και ανεμπόδιστα με τη Java, με την έκδοση Java Virtual Machine (JVM) της τυπικής βιβλιοθήκης της Kotlin να εξαρτάται από τη βιβλιοθήκη κλάσης της Java. Η εξαγωγή συμπερασμάτων τύπου επιτρέπει στη σύνταξή της να είναι πιο συνοπτική. Αρχικά σχεδιάστηκε για το JVM, πράγμα που συνεπάγεται τη συγγραφή εφαρμογών με μεταγλώττιση σε Java bytecode (π.χ. για την ανάπτυξη εφαρμογών Android) αλλά μεταγλωττίζει και σε JavaScript (π.χ. για frontend web εφαρμογή με χρήση React) ή σε εγγενή (native) κώδικα (π.χ. για εγγενείς εφαρμογές iOS που έχουν κοινή επιχειρησιακή λογική με εφαρμογές Android). [19]

Η Kotlin ξεκίνησε από την JetBrains, την εταιρεία πίσω από το IntelliJ IDEA, το 2010. Ο επικεφαλής της JetBrains, Dmitry Jemerov, δήλωσε ότι οι περισσότερες γλώσσες δεν είχαν τα χαρακτηριστικά που αναζητούσαν, με εξαίρεση τη Scala, της οποίας όμως ο αργός χρόνος μεταγλώττισης αποτελούσε σημαντική έλλειψη. Ένας από τους αναφερόμενους στόχους της Kotlin είναι να μεταγλωττίζει το ίδιο γρήγορα με τη Java. Τον Φεβρουάριο του 2012, η JetBrains προχώρησε στην ανοιχτή διάθεση του έργου με την άδεια ανοικτού κώδικα Apache 2.

Μέχρι τον Μάιο του 2017, οι μόνες επίσημα υποστηριζόμενες γλώσσες προγραμματισμού για το Android ήταν η Java και η C++, όταν η Google ανακοίνωσε την επίσημη υποστήριξη της Kotlin στο Android στο Google I/O 2017. Από το Android Studio 3.0 η Kotlin ενσωματώνεται στο σύνολο εργαλείων ανάπτυξης Android ενώ στις 7 Μαΐου 2019, η Google

ανακοίνωσε ότι η γλώσσα προγραμματισμού Kotlin είναι πλέον η προτιμώμενη γλώσσα για τους προγραμματιστές εφαρμογών Android. Η τρέχουσα έκδοση είναι η Kotlin v1.5, η οποία κυκλοφόρησε τον Μάιο του 2021.[20]

2.4 IntelliJ Idea

Το IntelliJ IDEA είναι ένα έξυπνο, ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) γραμμένο σε Java για την ανάπτυξη λογισμικού, που προσαρμόζεται στο συγκείμενο. Αναπτύσσεται από την JetBrains (παλαιότερα γνωστή ως IntelliJ) και είναι κατάλληλο για πλήρεις διαδικτυακές εφαρμογές, χάρη στα ισχυρά ενσωματωμένα εργαλεία του, την υποστήριξη της JavaScript και συναφών τεχνολογιών και την προηγμένη υποστήριξη δημοφιλών πλαισίων(frameworks).

Η πρώτη έκδοση του IntelliJ IDEA κυκλοφόρησε τον Ιανουάριο του 2001, και ήταν ένα από τα πρώτα διαθέσιμα Java IDEs με ενσωματωμένες προηγμένες δυνατότητες πλοήγησης στον κώδικα και αναδιαμόρφωσης κώδικα. Σε μια έκθεση του InfoWorld το 2010, το IntelliJ έλαβε την υψηλότερη βαθμολογία του κέντρου δοκιμών ανάμεσα στα τέσσερα κορυφαία εργαλεία προγραμματισμού για Java: Eclipse, IntelliJ IDEA, NetBeans και JDeveloper. Τον Δεκέμβριο του 2014, η Google ανακοίνωσε την έκδοση 1.0 του Android Studio, το οποίο βασίζεται στην ανοικτού κώδικα έκδοση Community του IntelliJ IDEA.

Μερικά από τα βασικά χαρακτηριστικά είναι η πρόσθετη βοήθεια συγγραφής κώδικα με ορισμένα χαρακτηριστικά όπως η συμπλήρωση κώδικα με ανάλυση του συγκείμενου, η πλοήγηση κώδικα που επιτρέπει την άμεση μετάβαση σε μια κλάση ή δήλωση στον κώδικα, η αναδιαμόρφωση κώδικα, η αποσφαλμάτωση κώδικα και οι επιλογές διόρθωσης ασυνεπειών μέσω προτάσεων. Επιπλέον, το IDE παρέχει πολλά ενσωματωμένα εργαλεία και εργαλεία κατασκευής/συσκευασίας όπως τα grunt, bower, gradle και SBT. ενώ υποστηρίζει συστήματα ελέγχου εκδόσεων όπως Git, Mercurial, Perforce και SVN και βάσεις δεδομένων όπως Microsoft SQL Server, Oracle, PostgreSQL, SQLite και MySQL. Σημαντικό ρόλο παίζει και το μεγάλο οικοσύστημα από plugins που προσθέτουν λειτουργικότητα στο IDE. Κάθε έκδοση έχει ξεχωριστά αποθετήρια από plugins, με τις εκδόσεις Community και Ultimate να έχουν συνολικά πάνω από 3000 πρόσθετα η καθεμία από το 2019, ενώ υποστηρίζονται περισσότερες από 30 γλώσσες.[21]

2.5 Python

Η Python είναι μια υψηλού επιπέδου, γενικού σκοπού γλώσσα προγραμματισμού που μπορεί να εφαρμοστεί σε διάφορες κατηγορίες προβλημάτων. Είναι διερμηνευόμενη (interpreted), διαδραστική και αντικειμενοστραφής(object-oriented). Ο κύριος στόχος της είναι η αναγνωσιμότητα του κώδικά της και η ευκολία χρήσης της. Το συντακτικό της επιτρέπει στους προγραμματιστές να εκφράσουν έννοιες σε λιγότερες γραμμές κώδικα από ότι θα ήταν δυνατόν σε γλώσσες όπως η C++ ή η Java.

Η Python δημιουργήθηκε στις αρχές της δεκαετίας του 1990 από τον Guido van Rossum στο Stichting Mathematisch Centrum στην Ολλανδία ως διάδοχος μιας γλώσσας που ονομαζόταν ABC, αλλά η ανάπτυξη της γλώσσας έλαβε χώρα κάτω από διαφορετικές εταιρείες και ιδρύματα ανά τα χρόνια. Αυτό συνέβαινε μέχρι το 2001, όταν δημιουργήθηκε το Python

Software Foundation (PSF), ένας μη κερδοσκοπικός οργανισμός ειδικά για να κατέχει την πνευματική ιδιοκτησία και δικαιώματα σχετικά με την Python. Η Zope Corporation αποτελεί μέλος-χορηγός του PSF.

Ενσωματώνει ενότητες (modules) εξαιρέσεις, δυναμική γραφή, δυναμικούς τύπους δεδομένων πολύ υψηλού επιπέδου και κλάσεις. Υποστηρίζει επίσης πολλαπλά υποδείγματα προγραμματισμού πέρα από τον αντικειμενοστραφή προγραμματισμό, όπως ο διαδικαστικός και ο προστακτικός προγραμματισμός. Η Python συνδυάζει μεγάλη ισχύ με πολύ ξεκάθαρο συντακτικό. Διατίθεται με μια μεγάλη βιβλιοθήκη που καλύπτει τομείς όπως string processing, διαδικτυακά πρωτόκολλα, μηχανική λογισμικού και διεπαφές λειτουργικών συστημάτων. Ταυτόχρονα, είναι επεκτάσιμη σε C ή C++ και μπορεί να χρησιμοποιηθεί ως γλώσσα επέκτασης για εφαρμογές που χρειάζονται μια προγραμματιζόμενη διεπαφή. Τέλος, η Python είναι φορητή: τρέχει σε πολλές παραλλαγές του Unix, συμπεριλαμβανομένων των Linux και macOS, και στα Windows, ενώ όλες οι εκδόσεις είναι ανοιχτού κώδικα. [22]

2.6 Μηχανική Μάθηση

Η μηχανική μάθηση (ML) είναι το σύνολο των μεθόδων που μπορούν να ανιχνεύουν αυτόματα μοτίβα στα δεδομένα, και στη συνέχεια χρησιμοποιούν αυτά τα μοτίβα για να προβλέψουν μελλοντικά δεδομένα, ή να εκτελέσουν άλλα είδη αποφάσεων υπό αβεβαιότητα.

Η μηχανική μάθηση αποτελεί μέρος του κλάδου της Τεχνητής Νοημοσύνης. Πολύ συχνά οι δύο όροι λανθασμένα συγχέονται. Η τεχνητή νοημοσύνη είναι η επιστήμη και η τεχνολογία κατασκευής ευφυών μηχανών. Πρόκειται λοιπόν για τον γενικότερο κλάδο της πληροφορικής που ασχολείται με την ανάπτυξη ευφυών μηχανών, μιας προσπάθειας δηλαδή να κάνουμε τους υπολογιστές να μιμηθούν την ευφυή συμπεριφορά του ανθρώπου ώστε να εκτελέσουμε εργασίες οι οποίες απαιτούν ανθρωπινή ευφυΐα όπως αναγνώριση λόγου, οπτική αντίληψη και λήψη αποφάσεων.

Η βασική διαφορά της μηχανικής μάθησης συγκριτικά με άλλα πεδία της τεχνητής νοημοσύνης, είναι η ικανότητα της να τροποποιείται και να προσαρμόζεται στα νέα δεδομένα που δέχεται. Μέσω αυτής της ικανότητας της μηχανικής μάθησης, μπορούμε πλέον τροφοδοτώντας με δεδομένα να οδηγήσουμε τους υπολογιστές σε νέα γνώση την οποία ούτε οι ίδιοι κατέχουμε. Στις μέρες μας, όπου πληθώρα δεδομένων είναι διαθέσιμα και υπάρχει ολοένα και μεγαλύτερη ψηφιοποίηση αυτών των δεδομένων, η μηχανική μάθηση αποτελεί ένα πολύ ισχυρό εργαλείο για την επεξεργασία και αξιοποίηση αυτού του όγκου πληροφορίας, καθώς και για τη δημιουργία νέας γνώσης.

Οι αλγόριθμοι μηχανικής μάθησης δημιουργούν ένα μοντέλο βασισμένο σε δείγματα δεδομένων, γνωστά ως "train data", προκειμένου να μπορούν έπειτα να κάνουν προβλέψεις ή να παίρνουν αποφάσεις χωρίς να έχουν προγραμματιστεί ρητά να το κάνουν.[23]

2.7 Natural Language Processing (NLP)

Natural Language Processing (NLP) σημαίνει Επεξεργασία Φυσικής Γλώσσας και ευρέως ορίζεται ως ο αυτόματος χειρισμός της φυσικής γλώσσας, όπως η ομιλία και το κείμενο, από λογισμικό. Αποτελεί διεπιστημονικό κλάδο της επιστήμης της πληροφορικής, της τεχνητής

νοημοσύνης (και της μηχανικής μάθησης) και της υπολογιστικής γλωσσολογίας και ασχολείται με τις αλληλεπιδράσεις μεταξύ των υπολογιστών και των ανθρώπινων (φυσικών) γλωσσών. Ασχολείται με την ικανότητα ενός υπολογιστή να κατανοεί, να αναλύει, να χειρίζεται και ενδεχομένως να παράγει ανθρώπινη γλώσσα.

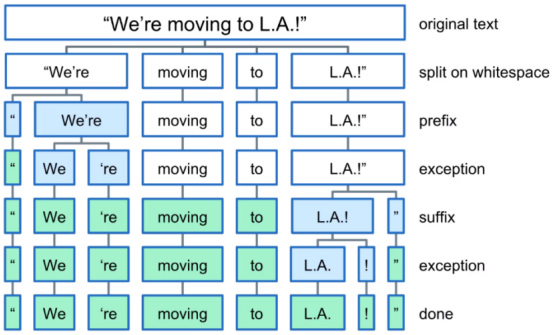
Η μελέτη της NLP υπάρχει εδώ και περισσότερα από 50 χρόνια και ξέφυγε από τον τομέα της γλωσσολογίας με την εδραίωση των υπολογιστών. Από τότε που ο άνθρωπος δημιούργησε τους υπολογιστές, ήθελε πάντα να μπορεί να επικοινωνήσει και να καταφέρει ο υπολογιστής να τον καταλαβαίνει. Πλέον, έχουμε smartphones με λογισμικό "Text-to-speech" για τη μετατροπή της ανθρώπινης γλώσσας σε κείμενο. Είτε πρόκειται για μετατροπή κειμένου σε ομιλία είτε για το αντίστροφο, όλα εντάσσονται στο κλάδο της NLP. Η NLP έχει τεράστιο πεδίο εφαρμογής και παίζει σημαντικό ρόλο στη δημιουργία μιας γέφυρας αλληλεπίδρασης ανθρώπου-υπολογιστή. Θα μπορούσαμε να θεωρήσουμε την NLP ως οποιοδήποτε είδος επεξεργασίας της φυσικής γλώσσας από τον υπολογιστή. Στο ένα άκρο, θα μπορούσε να είναι τόσο απλό όσο η καταμέτρηση συχνότητας των λέξεων για τη σύγκριση διαφορετικών στυλ γραφής ενώ στο άλλο άκρο, η NLP περιλαμβάνει την «κατανόηση» ολοκληρωμένων ανθρώπινων εκφράσεων, τουλάχιστον σε βαθμό που να είναι σε θέση να δίνει χρήσιμες απαντήσεις σε αυτές.[24]

Οι εφαρμογές της NLP περιλαμβάνουν ταξινόμηση (classification) και κατηγοριοποίηση (categorization) κειμένου, αναγνώριση ονομαστικών οντοτήτων (Named Entity Recognition), επισήμανση μερών του λόγου (Parts of Speech Tagging), σημασιολογική ανάλυση (Semantic Parsing), μηχανική μετάφραση (Machine Translation), ανάλυση συναισθήματος (Sentiment Analysis), έλεγχο ορθογραφίας, φίλτρο spam, αναγνώριση χαρακτήρων, αναγνώριση ομιλίας και άλλα. Μερικά από τα πιο δημοφιλή εργαλεία στον κλάδο είναι το Standford CoreNLP, το NLTK της Python, το SpaCy και το TextBlog, με πολλές από τις λειτουργίες που παρέχουν να αποτελούν εκπαιδευμένα μοντέλα μηχανικής μάθησης.

2.7.1 Natural Language Toolkit (NLTK)

Το Natural Language Toolkit (NLTK) είναι μια πολύ σημαντική πλατφόρμα ανοικτού κώδικα για την κατασκευή προγραμμάτων Python, μηχανικής μάθησης και μη, που πραγματεύονται δεδομένα ανθρώπινης γλώσσας. Παρέχει εύχρηστες διεπαφές σε πάνω από 50 οντότητες κειμένων και πόρους λεξικών όπως το WordNet. Περιέχει βιβλιοθήκες, αλγορίθμους και προγράμματα για συμβολική NLP, στατιστική NLP, καθώς και με τη χρήση μηχανικής μάθησης. Οι βιβλιοθήκες αυτές είναι πολύ μεγάλης ισχύος και περιέχουν αλγορίθμους και ήδη εκπαιδευμένα μοντέλα με τα οποία επιτυγχάνονται λειτουργίες επεξεργασίας κειμένου όπως classification, tokenization, stemming, POS tagging, parsing, sentiment analysis, topic segmentation και named entity recognition.[25] Ορισμένες είναι:

- Το Tokenization είναι το πρώτο βήμα στην ανάλυση κειμένου. Η διαδικασία ανάλυσης και διάσπασης μιας παραγράφου κειμένου σε επιμέρους κομμάτια, όπως παραγράφους, προτάσεις ή tokens, ονομάζεται Tokenization. Το "Token" είναι μια ενιαία οντότητα που αποτελεί δομικό στοιχείο για την πρόταση ή την παράγραφο.[26] Η μέθοδος tokenize της NLTK υλοποιεί ένα συνδυασμό του αλγόριθμου κανονικών εκφράσεων TreebankWordTokenizer μαζί με το ήδη εκπαιδευμένο μοντέλο του PunktSentenceTokenizer.[27]

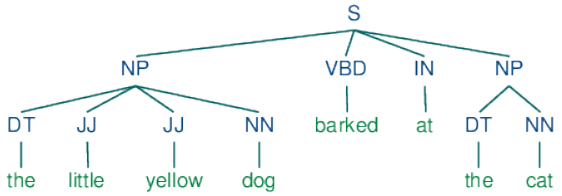


Εικόνα 2.1: *Παράδειγμα Tokenization πρότασης*
[28]

- Το Lemmatization μειώνει τις λέξεις στη βασική τους λέξη, η οποία είναι το γλωσσικά γνωστό λήμμα. Τη μετατρέπει στη λέξη-ρίζα με τη χρήση λεξιλογικής και μορφολογικής ανάλυσης για παράδειγμα οι λέξεις rocks -> rock, cries -> cry και better -> good. Γίνεται με τη χρήση της ενσωματωμένης μεθόδου "morphy", η οποία υλοποιεί κανόνες γλωσσολογίας και πιο συγκεκριμένα έναν συνδυασμό κανόνων κλιτικών καταλήξεων μαζί με λίστες εξαιρέσεων.

- Το Part-of-Speech (POS) Tagging έχει ως στόχο τον προσδιορισμό της γραμματικής ομάδας μιας δεδομένης λέξης δηλαδή αν πρόκειται για ουσιαστικό, αντωνυμία, επίθετο, ρήμα, επίρρημα κ.λπ. με βάση το συγκείμενο. Το POS Tagging αναζητά σχέσεις μέσα στην πρόταση και αποδίδει μια αντίστοιχη ετικέτα στη λέξη. Ο αλγόριθμος του POS tagger που μας παρέχει η NLKT αποτελεί ένα ήδη εκπαιδευμένο μοντέλο μηχανικής μάθησης το οποίο έχει εκπαιδευτεί και δοκιμαστεί με δεδομένα της εφημερίδας The Wall Street Journal.[29] Ταυτόχρονα, χρησιμοποεί το Penn Treebank tagset, το οποίο βλέπουμε αναλυτικά στον πίνακα 2.1. [30]

  Παρακάτω, βλέπουμε ένα πιο συγκεκριμένο παράδειγμα POS tagging με τη χρήση NP Chunker στην πρόταση "The little yellow dog barked at the cat".



Εικόνα 2.2: *Παράδειγμα POS-Tagging με NP Chunker*

Table 2.1: *Penn Treebank Part-of-Speech Tags*

Tag	**Meaning**
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun
PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	*to*
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner
WP	Wh-pronoun
WP$	Possessive wh-pronoun
WRB	Wh-adverb

- Το Named Entity Recognition (NER) είναι ουσιαστικά ο εντοπισμός της φύσης μιας οντότητας σε σχέση με τον πραγματικό κόσμο, όπως αν πρόκειται για ένα πρόσωπο ή έναν οργανισμό, μέσα σε ένα δεδομένο κείμενο. Χρησιμοποιεί τα part-of-speech annotations για να προσθέσει κατάλληλες ετικέτες σε κάθε λέξη καθώς και ενσωματωμένα

μοντέλα ή εκπαιδευμένα δεδομένα για την επιτυχής ανάθεση των ετικετών. Με αυτόν τον τρόπο, το NLTK μπορεί να αναγνωρίσει και να ταυτοποιήσει κάθε οντότητα και έτσι να κατανοήσει το νόημα μιας πρότασης.

2.7.2 WordNet

Η WordNet είναι μια μεγάλη λεξιλογική βάση δεδομένων της αγγλικής γλώσσας, ελεύθερα και δημόσια διαθέσιμη για λήψη. Ουσιαστικά, ρήματα, επίθετα και επιρρήματα ομαδοποιούνται σε σύνολα γνωστικών συνωνύμων (synsets), καθένα από τα οποία εκφράζει μια ξεχωριστή έννοια. Τα σύνολα συνδέονται μεταξύ τους μέσω εννοιολογικών-σημασιολογικών και λεξιλογικών σχέσεων και δημιουργούν ένα δίκτυο από λέξεις και έννοιες. Η δομή της WordNet την καθιστά χρήσιμο εργαλείο για το τομέα της υπολογιστικής γλωσσολογίας και της επεξεργασίας φυσικής γλώσσας(NLP).

Η WordNet μοιάζει επιφανειακά με ένα θησαυρό (thesaurus - ένα συγκεκριμένο είδος λεξικού που παραθέτει λέξεις σε ομάδες συνωνύμων και συναφών εννοιών), καθώς ομαδοποιεί τις λέξεις με βάση τις έννοιές τους. Ωστόσο, υπάρχουν ορισμένες σημαντικές διαφορές. Πρώτον, η WordNet διασυνδέει μεταξύ τους όχι μόνο μορφές λέξεων -σειρές γραμμάτων- αλλά και συγκεκριμένες έννοιες των λέξεων. Ως αποτέλεσμα, οι λέξεις που βρίσκονται σε κοντινή απόσταση η μία από την άλλη στο δίκτυο είναι σημασιολογικά αποσαφηνισμένες και συσχετισμένες. Δεύτερον, η WordNet ονομάζει τις σημασιολογικές σχέσεις μεταξύ των λέξεων, ενώ οι ομαδοποιήσεις των λέξεων σε ένα «θησαυρό» δεν ακολουθούν κανένα ρητό πρότυπο εκτός από την ομοιότητα των σημασιών.

Η κύρια σχέση μεταξύ λέξεων στη WordNet είναι η συνωνυμία, όπως μεταξύ των λέξεων αυτοκίνητο και αμάξι. Τα συνώνυμα -λέξεις που δηλώνουν την ίδια έννοια και είναι ανταλλάξιμες σε πολλά συμφραζόμενα- ομαδοποιούνται σε μη ταξινομημένα σύνολα (synsets). Κάθε ένα από τα 117.000 σύνολα του WordNet συνδέεται με άλλα σύνολα μέσω ενός μικρού αριθμού «εννοιολογικών σχέσεων».

Η πιο συχνά κωδικοποιημένη σχέση μεταξύ συνόλων τώρα, είναι η σχέση υπερ-υπόταξης (super-subordinate) -που ονομάζεται επίσης υπερωνυμία (hyperonymy). Συνδέει πιο γενικά σύνολα όπως {έπιπλα, μέρος επίπλου} με όλο και πιο συγκεκριμένα όπως {κρεβάτι} και {κουκέτα}. Έτσι, η WordNet δηλώνει ότι η κατηγορία έπιπλα περιλαμβάνει το κρεβάτι, το οποίο με τη σειρά του περιλαμβάνει την κουκέτα ενώ αντίστροφα, έννοιες όπως το κρεβάτι και η κουκέτα συνθέτουν την κατηγορία έπιπλα. Όλες οι ιεραρχίες ουσιαστικών καταλήγουν τελικά στον κόμβο ρίζας {οντότητα (entity)}. Άλλες σχέσεις που κωδικοποιούνται στη WordNet είναι η μερωνυμία (meronymy), μια σχέση επιμέρους-όλου που ισχύει μεταξύ συνόλων όπως {καρέκλα} και {πλάτη}, {κάθισμα} και {πόδι}, η ιεραρχία στην οργάνωση των συνόλων ρημάτων σα δέντρο, όπου τα ρήματα προς τη βάση των δέντρων (troponyms) εκφράζουν όλο και πιο συγκεκριμένους τρόπους να χαρακτηρίσουμε ένα γεγονός, όπως στο «δέντρο» {επικοινωνώ}-{μιλάω}-{ψιθυρίζω} και η αντωνυμία (με την έννοια της αντιθετικότητας) στην οργάνωση των επιθέτων.[31]

2.8 Flask Web Framework

Το Flask είναι ένα ευρέως χρησιμοποιούμενο πλαίσιο μικροεφαρμογών ιστού (web application framework) για την ανάπτυξη RESTful APIs σε Python. Είναι ένα απλό αλλά ισχυρό web framework και χρησιμοποιείται συχνά για τη δημιουργία web applications, τη διαχείριση αιτημάτων HTTP και το template rendering. Το «μίκρο» δεν σημαίνει ότι το Flask στερείται λειτουργικότητας. Ο όρος μίκρο αναφέρεται στο ότι ο πυρήνας είναι απλός αλλά επεκτάσιμος. Δεδομένου ότι το Flask είναι ελαφρύ, είναι εύκολο να ενσωματωθεί σε οποιοδήποτε έργο.

Ένα framework είναι μια συλλογή βιβλιοθηκών και ενοτήτων (modules) που βοηθούν στη δημιουργία σύνθετων, κλιμακούμενων, συντηρήσιμων και αξιόπιστων εφαρμογών ενώ παρέχουν επαναχρησιμοποιήσιμο κώδικα και επεκτάσεις. Το Flask ξεκίνησε ως πρωταπριλιάτικο αστείο το 2010 από τον Armin Ronacher, ο οποίος ηγείται μιας ομάδας Python με το όνομα Pocco. Βασίζεται στο Werkzeug, το WSGI toolkit και το Jinja2 engine, τα οποία είναι όλα projects της Pocco. Έκτοτε, αποτελεί ένα από τα πιο γρήγορα αναπτυσσόμενα frameworks για Python και δημοφιλείς ιστότοποι όπως το Netflix, το Pinterest και το LinkedIn, έχουν ενσωματώσει το Flask στον κώδικα ανάπτυξής τους.

Ένα από τα μεγάλα πλεονεκτήματά του είναι ότι το Flask είναι σαν ένας άδειος καμβάς για τη δημιουργία εφαρμογών βασισμένων στην Python - δεν έχει κάποιο απαιτούμενο project layout και έχει λίγες εξαρτήσεις. Διαθέτει επίσης τον δικό του διακομιστή ανάπτυξης για την εκτέλεση οποιασδήποτε εφαρμογής (ο οποίος δεν προορίζεται και για διανομή) και debugger που ανανεώνει τον διακομιστή όταν γίνονται αλλαγές στον κώδικα. Με το Jinja2, επιτρέπει την ασφαλή εκτέλεση σε sandboxed περιβάλλον, ενώ το Templating και το Jinja επιτρέπουν την εύκολη ανταλλαγή δεδομένων μεταξύ frontend και backend. Ταυτόχρονα, βασιζόμενο στο εργαλείο Werkzeug WSGI, παρέχει μια διεπαφή πύλης διακομιστή ιστού (web server gateway interface) για την επικοινωνία μεταξύ του web server και της web application. Τέλος, έχει δυνατότητες όπως cookies, sessions, βιβλιοθήκη βάσεων δεδομένων, χειρισμό μεταφόρτωσης, επικύρωση φόρμας και τεχνολογίες ανοικτού ελέγχου ταυτότητας, καθιστώντας το ελκυστικό για χρήση σε εφαρμογές ιστού. [32]

2.9 Docker

Η Docker είναι μια πλατφόρμα λογισμικού ανοιχτού κώδικα που υλοποιεί εικονικοποίηση (Virtualization) σε επίπεδο λειτουργικού συστήματος. Ουσιαστικά το Docker προσφέρει αυτοματοποιημένες διαδικασίες για την ανάπτυξη εφαρμογών σε απομονωμένες περιοχές χρήστη (User Spaces) που ονομάζονται Software Containers. Το λογισμικό χρησιμοποιεί τεχνολογίες του πυρήνα (kernel) του Linux για να επιτρέπει σε ανεξάρτητα software containers να εκτελούνται στο ίδιο λειτουργικό σύστημα. Έτσι, αποφεύγεται η χρήση επιπλέον υπολογιστικών πόρων που θα απαιτούσε μια εικονική μηχανή (virtual machine).

Η Docker Inc. ιδρύθηκε από τους Kamel Founadi, Solomon Hykes και Sebastien Pahl κατά τη διάρκεια του Y Combinator Summer 2010 και ξεκίνησε το 2011. Ο Hykes ξεκίνησε το έργο Docker στη Γαλλία ως εσωτερικό έργο στο πλαίσιο της dotCloud, μιας εταιρείας τύπου ¨πλατφόρμας ως υπηρεσία' (platform-as-a-service)]. Το Docker έκανε το ντεμπούτο του στο κοινό στη Santa Clara, στο PyCon το 2013. Δημοσιεύτηκε ως λογισμικό ανοικτού

κώδικα τον Μάρτιο του 2013. Την εποχή εκείνη, χρησιμοποίησε το LXC ως το προεπιλεγμένο περιβάλλον εκτέλεσης. Ένα χρόνο αργότερα, με την κυκλοφορία της έκδοσης 0.9, το Docker αντικατέστησε το LXC με το δικό του component, το οποίο γράφτηκε στη γλώσσα προγραμματισμού Go. Το 2017, η Docker δημιούργησε το έργο Moby για ανοιχτή έρευνα και ανάπτυξη.

Η Docker μπορεί ουσιαστικά να συμπεριλάβει μια εφαρμογή και τις εξαρτήσεις της σε ένα virtual container, που μπορεί να τρέξει σε οποιοδήποτε σύστημα Linux. Αυτό βοηθά στην παροχή ευελιξίας και φορητότητας, επιτρέποντας την εκτέλεση της εφαρμογής σε διάφορες τοποθεσίες, είτε on-premises, είτε σε public cloud ή σε private cloud. Η Docker χρησιμοποιεί τις λειτουργίες απομόνωσης πόρων του πυρήνα του Linux όπως τα cgroups και τους χώρους ονομάτων πυρήνα (kernel namespaces) και ένα union-capable σύστημα αρχείων (όπως το OverlayFS), για να επιτρέπει στα containers να εκτελούνται μέσα σε ένα ενιαίο στιγμιότυπο Linux, αποφεύγοντας το επιπλέον υπολογιστικό κόστος εκκίνησης και διατήρησης εικονικών μηχανών. Επειδή, τα πακέτα Docker (containers) είναι ελαφριά από υπολογιστικής απόψεως, ένας και μόνο διακομιστής ή εικονική μηχανή μπορεί να τρέξει ταυτόχρονα πολλά containers. Μια ανάλυση το 2018 διαπίστωσε ότι μια τυπική περίπτωση χρήσης Docker περιλαμβάνει την εκτέλεση οκτώ containers ανά host, και ότι το ένα τέταρτο των οργανισμών που αναλύθηκαν τρέχουν 18 ή περισσότερους ανά host. Η υποστήριξη του πυρήνα του Linux για χώρους ονομάτων, ως επί το πλείστον, απομονώνει την πρόσβαση και αλληλεπίδραση μιας εφαρμογής με το λειτουργικό σύστημα, συμπεριλαμβανομένων των δέντρων διεργασιών, του δικτύου, των αναγνωριστικών χρήστη, και των συστημάτων αρχείων, ενώ τα cgroups του πυρήνα παρέχουν περιορισμό στη χρήση πόρων για τη μνήμη και τη CPU. Επιπρόσθετα, εκτός από τη χρήση abstracted διεπαφών εικονικοποίησης μέσω libvirt, LXC και systemd-nspawn, από την έκδοση 0.9, το Docker περιλαμβάνει το δικό του component (που ονομάζεται "libcontainer") για να χρησιμοποιεί απευθείας τις δυνατότητες εικονικοποίησης, που παρέχει ο πυρήνας του Linux. Τέλος, το Docker υλοποιεί ένα API υψηλού επιπέδου για την παροχή containers που εκτελούν διαδικασίες μεμονωμένα.

2.9.1 Docker Desktop

Το Docker Desktop είναι μια εφαρμογή για περιβάλλον Mac ή Windows που επιτρέπει την κατασκευή και το διαμοιρασμό εφαρμογών και μικρουπηρεσιών με container. Το Docker Desktop αποτελείται από τα Docker Engine, Docker CLI client, Docker Compose, Docker Content Trust, Kubernetes και Credential Helper. Επιλέχθηκε γιατί παρέχει δυνατότητα container και κοινής χρήσης της εφαρμογής σε οποιαδήποτε πλατφόρμα cloud, σε πολλές γλώσσες και frameworks. Ταυτόχρονα, επιτρέπει την εύκολη εγκατάσταση και ρύθμιση ενός πλήρους περιβάλλοντος ανάπτυξης Docker καθώς και πρόσβαση σε εκτελούμενα container στο δίκτυο localhost. Τέλος, είναι δυνατή η ενσωμάτωση για in-container ανάπτυξη και αποσφαλμάτωση με δημοφιλή Ολοκληρωμένα Περιβάλλοντα Ανάπτυξης (IDEs) όπως το IntelliJ IDEA που χρησιμοποιήσαμε για την εφαρμογή Python.

2.10 Google Cloud Platform

Το Google Cloud Platform (GCP) είναι μια σουίτα υπηρεσιών υπολογιστικού νέφους (cloud computing) που εκτελείται με την ίδια υποδομή που χρησιμοποιεί η Google εσωτερικά για τα προϊόντα της για τελικούς χρήστες, όπως το Google Search, το Gmail, η αποθήκευση αρχείων και το YouTube. Παράλληλα με ένα σύνολο εργαλείων διαχείρισης, παρέχει μια σειρά από υπηρεσίες νέφους που περιλαμβάνουν υπολογιστική ισχύς, αποθήκευση δεδομένων, ανάλυση δεδομένων και μηχανική μάθηση. Το GCP παρέχει υποδομή ως υπηρεσία (infrastructure as a service), πλατφόρμα ως υπηρεσία (platform as a service) και περιβάλλοντα υπολογισμού χωρίς διακομιστή (serverless computing environments).

Τον Απρίλιο του 2008, η Google ανακοίνωσε το App Engine, μια πλατφόρμα για την ανάπτυξη και το deployment εφαρμογών ιστού σε κέντρα δεδομένων που διαχειρίζεται η Google, η οποία ήταν η πρώτη υπηρεσία υπολογιστικού νέφους από την εταιρεία. Η υπηρεσία έγινε διαθέσιμη για το κοινό τον Νοέμβριο του 2011. Ύστερα από την ανακοίνωση του App Engine, η Google έχει προβεί στην προσθήκη πολλαπλών υπηρεσιών νέφους στην πλατφόρμα.

Μία από αυτές είναι και το Cloud Run. Το Cloud Run προσφέρει βελτιωμένη εμπειρία στους προγραμματιστές, καθιστώντας την ανάπτυξη και το deployment εφαρμογών απλούστερη και ταχύτερη. Τα βασικά χαρακτηριστικά του περιλαμβάνουν τη φιλοξενία οποιουδήποτε κώδικα ή container που ακούει για αιτήματα ή συμβάντα σε ένα περιβάλλον χωρίς διακομιστή. Είναι εντελώς ανεξάρτητο από τη γλώσσα προγραμματισμού ή τις βιβλιοθήκες του λειτουργικού συστήματος. Επιπρόσθετα, είναι «πλήρως διαχειρίσιμο» από μόνο του, πράγμα που σημαίνει ότι το Cloud Run αφαιρεί από το προγραμματιστή όλη τη διαχείριση της υποδομής με αυτόματη κλιμάκωση προς τα πάνω και προς τα κάτω από το μηδέν σχεδόν ακαριαία και ανάλογα με την κίνηση. Τέλος, το Cloud Run συνδυάζεται εξαιρετικά με το οικοσύστημα των containers: Cloud Build, Cloud Code, Artifact Registry και Docker. [33]

2.11 FoodData Central

Το FoodData Central είναι ένα ολοκληρωμένο, εστιασμένο στην έρευνα σύστημα δεδομένων που παρέχει διευρυμένα δεδομένα για τα θρεπτικά συστατικά και άλλα στοιχεία των τροφίμων. Δημιουργήθηκε από το Υπουργείο Γεωργίας των ΗΠΑ (USDA) προκειμένου να ενοποιηθούν όλες οι βασικές πηγές δεδομένων του USDA για τα τρόφιμα και τα θρεπτικά συστατικά σε ένα μέρος, καθιστώντας την πρόσβαση και τη χρήση πιο απλοποιημένη. Ενστερνίζεται μια νέα προσέγγιση στην ανάλυση, τη σύνθεση και την παρουσίαση πληροφοριών για το προφίλ των τροφίμων με επιστημονικά αυστηρό τρόπο. Οι τιμές στο FoodData Central προκύπτουν μέσω των πλέον σύγχρονων χημικών αναλύσεων, υπολογισμών και άλλων μεθόδων. Το εγχείρημα αυτό αποσκοπεί στο να καταστήσει καθημερινά δυνατή τη διεξαγωγή βελτιωμένων αναλύσεων των διατροφικών προσλήψεων από ερευνητές και ειδικούς.

Τα δεδομένα είναι προσβάσιμα μέσω αναζήτησης, χρησιμοποιώντας τη λειτουργία αναζήτησης σε αυτόν τον σύνδεσμο fdc.nal.usda.gov/index.html (Πρόσβαση στις 01-09-2021). Το API του FoodData Central παρέχει στους προγραμματιστές έναν μηχανισμό για την ενσωμάτωση δεδομένων θρεπτικών συστατικών στις εφαρμογές ή τους ιστότοπούς τους. Δημιουργήθηκε για να καλύψει την επείγουσα ανάγκη για διαφανείς και εύκολα προσβάσιμες

πληροφορίες σχετικά με τα θρεπτικά συστατικά και άλλα στοιχεία των τροφίμων και των προϊόντων τροφίμων. Η διαθεσιμότητα των τροφίμων αλλάζει και εξελίσσεται συνεχώς, με νέα προϊόντα να εισάγονται συνεχώς, ενώ ακόμη και τα συστατικά και τα ακατέργαστα γεωργικά προϊόντα αλλάζουν επίσης συνεχώς.[34]

2.12 Room library

Η Androidx.room, ή απλά Room, είναι μια βιβλιοθήκη χαρτογράφησης αντικειμένων βάσης δεδομένων (Database Object Mapping library) που διευκολύνει την πρόσβαση σε βάσεις δεδομένων σε εφαρμογές Android. Παρέχει ένα στρώμα αφαίρεσης πάνω από την SQLite για να επιτρέπει πιο σταθερή πρόσβαση σε βάσεις δεδομένων, αξιοποιώντας παράλληλα την πλήρη ισχύ της. Αντί να αποκρύπτει τις λεπτομέρειες της SQLite, η Room προσπαθεί να τις «αγκαλιάσει» παρέχοντας βολικά APIs για την υποβολή ερωτημάτων (queries) στη βάση δεδομένων, αλλά και την επαλήθευση τέτοιων ερωτημάτων κατά τη μεταγλώττιση. Έτσι, επιτρέπει την πρόσβαση στην πλήρη ισχύ της SQLite, ενώ παράλληλα δημιουργεί την ασφάλεια τύπων που παρέχουν οι Java SQL query builders.

Οι εφαρμογές που χειρίζονται μη τετριμμένες ποσότητες δομημένων δεδομένων μπορούν να επωφεληθούν σημαντικά. Το πιο συνηθισμένο σενάριο χρήσης του είναι η προσωρινή αποθήκευση σχετικών κομματιών πληροφορίας, ώστε όταν η συσκευή δεν έχει πρόσβαση στο δίκτυο, ο χρήστης να έχει τη δυνατότητα να περιηγείται σε αυτά ακόμα κι όντας offline. Υπάρχουν 3 σημαντικά στοιχεία στη Room:

- Η κλάση της βάσης δεδομένων που κρατάει τη βάση δεδομένων και τους διακομιστές σαν το κύριο σημείο πρόσβασης για την υφιστάμενη σύνδεση στα μόνιμα δεδομένα της εφαρμογής.
- Οντότητες (entities) δεδομένων που αναπαριστούν πίνακες στη βάση δεδομένων της εφαρμογής και συγκρατούν τα δεδομένα της.
- Τα αντικείμενα πρόσβασης σε δεδομένα (Data access objects - DAOs) είναι τα κύρια στοιχεία της Room που είναι υπεύθυνα για τον ορισμό των μεθόδων για την πρόσβαση στη βάση δεδομένων. Παρέχουν μεθόδους προκειμένου η εφαρμογή να πραγματοποιεί λειτουργίες όπως query, update, insert και delete στη βάση δεδομένων.

Η κλάση της βάσης δεδομένων παρέχει στην εφαρμογή στιγμιότυπα των DAOs που σχετίζονται με τη συγκεκριμένη βάση δεδομένων. Με τη σειρά της, η εφαρμογή χρησιμοποιεί τα DAOs για να ανακτήσει δεδομένα από τη βάση δεδομένων σε μορφή στιγμιοτύπων των σχετικών entity objects. Η χρήση της κλάσης DAO για την πρόσβαση στη βάση δεδομένων αντί των query builders ή των άμεσων queries επιτρέπει το διαχωρισμού μεταξύ των διαφόρων τμημάτων και την εύκολη εξομοίωση της πρόσβασης στη βάση δεδομένων κατά τη δοκιμή μιας εφαρμογής.[35]

Κεφάλαιο 3

Ανάλυση

Στο κεφάλαιο αυτό παρουσιάζεται η μελέτη και η ανάλυση που έγινε για την υλοποίηση και σχεδίαση της εφαρμογής. Αρχικά περιγράφεται η ίδια η εφαρμογή καθώς και οι πιο σημαντικές απαιτήσεις, ενώ στη συνέχεια αναλύονται οι λειτουργίες και τα σενάρια χρήσης της.

3.1 Απαιτήσεις Εφαρμογής για Ανάκτηση Διατροφικών Στοιχείων Γευμάτων

Είδαμε ότι ο καθένας μας μπορεί να επωφεληθεί άμεσα από την εύκολη πρόσβαση σε πληροφορίες διατροφικού χαρακτήρα. Αυτό πολλές φορές αποτελεί και σημαντική ανάγκη. Η σύγχρονη τεχνολογία με τις διευκολύνσεις που παρέχει μπορεί να απολέσει σύμμαχο του στην προσπάθεια αυτή. Ταυτόχρονα, είναι γεγονός ότι οι έξυπνες κινητές συσκευές (smartphones) έχουν γίνει αναπόσπαστο κομμάτι της ζωή του, καθώς παρέχουν χιλιάδες εφαρμογές για μετακινήσεις, ψυχαγωγία, επικοινωνία, αθλητισμό αλλά και μόρφωση. Συνεπώς, ιδιαίτερα χρήσιμη θα φαινόταν μια εφαρμογή για κινητές συσκευές η οποία θα ικανοποιούσε την ανάγκη αυτή εύκολα, άμεσα, απλά και κατανοητά για το μέσο χρήστη. Δεδομένου ότι οι πληροφορίες αυτές είναι ήδη διαθέσιμες online, η χρηστικότητα αποτελεί σημαντική απαίτηση στην κατεύθυνση της σχεδίασης της εφαρμογής. Για αυτό το λόγο, πάρθηκαν και αποφάσεις όπως το να γίνεται η είσοδος δεδομένων σε μορφή πρότασης φυσικής γλώσσας καθώς και η δυνατότητα επέκτασης με Speech Recognition.

3.2 Περιγραφή Εφαρμογής

Η εφαρμογή που δημιουργήθηκε στα πλαίσια αυτής της διπλωματικής είναι μια εφαρμογή με σκοπό να βοηθήσει οποιονδήποτε να αποκτήσει πρόσβαση στις διατροφικές πληροφορίες των τροφών που καταναλώνει. Ο χρήστης θα έχει τη δυνατότητα να πληκτρολογεί σε μορφή πρότασης στην Αγγλική γλώσσα το σύνολο των τροφών που κατανάλωσε μαζί με τις αντίστοιχες ποσότητες τους ώστε να αποκτήσει τα αποτελέσματα για την καθεμία από διαδεδομένη βάση δεδομένων τροφών. Επίσης, θα έχει τη δυνατότητα πρόσβασης εκτός δικτύου σε τροφές που έχει αναζητήσει πρόσφατα.

3.2.1 Λειτουργίες εφαρμογής

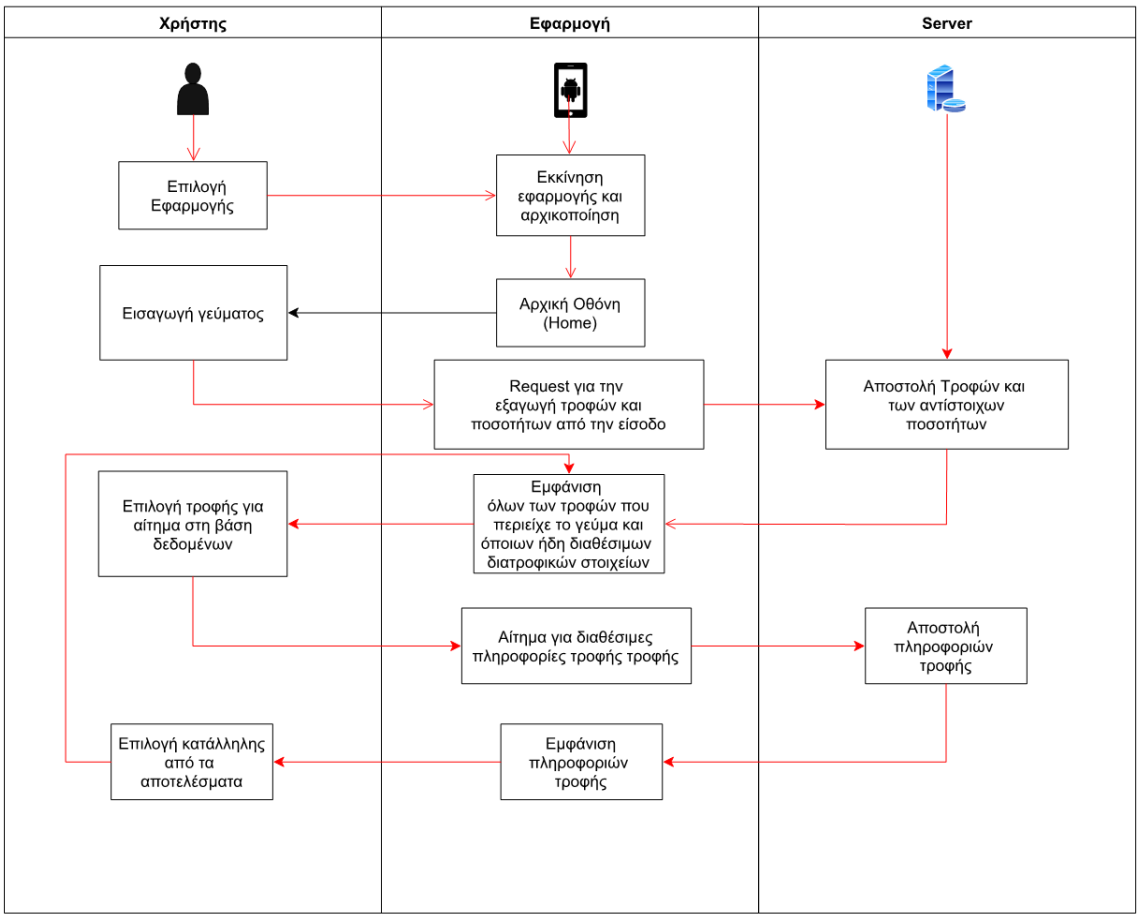
Η εφαρμογή μας προσφέρει ουσιαστικά δύο εκτενείς λειτουργίες στο χρήστη:

- Αναζήτηση και προβολή διατροφικών στοιχείων γεύματος
- Προβολή πρόσφατων αναζητήσεων και των εκάστοτε διατροφικών στοιχείων τους

3.3 Σενάρια Χρήσης

3.3.1 Αναζήτηση και προβολή διατροφικών στοιχείων

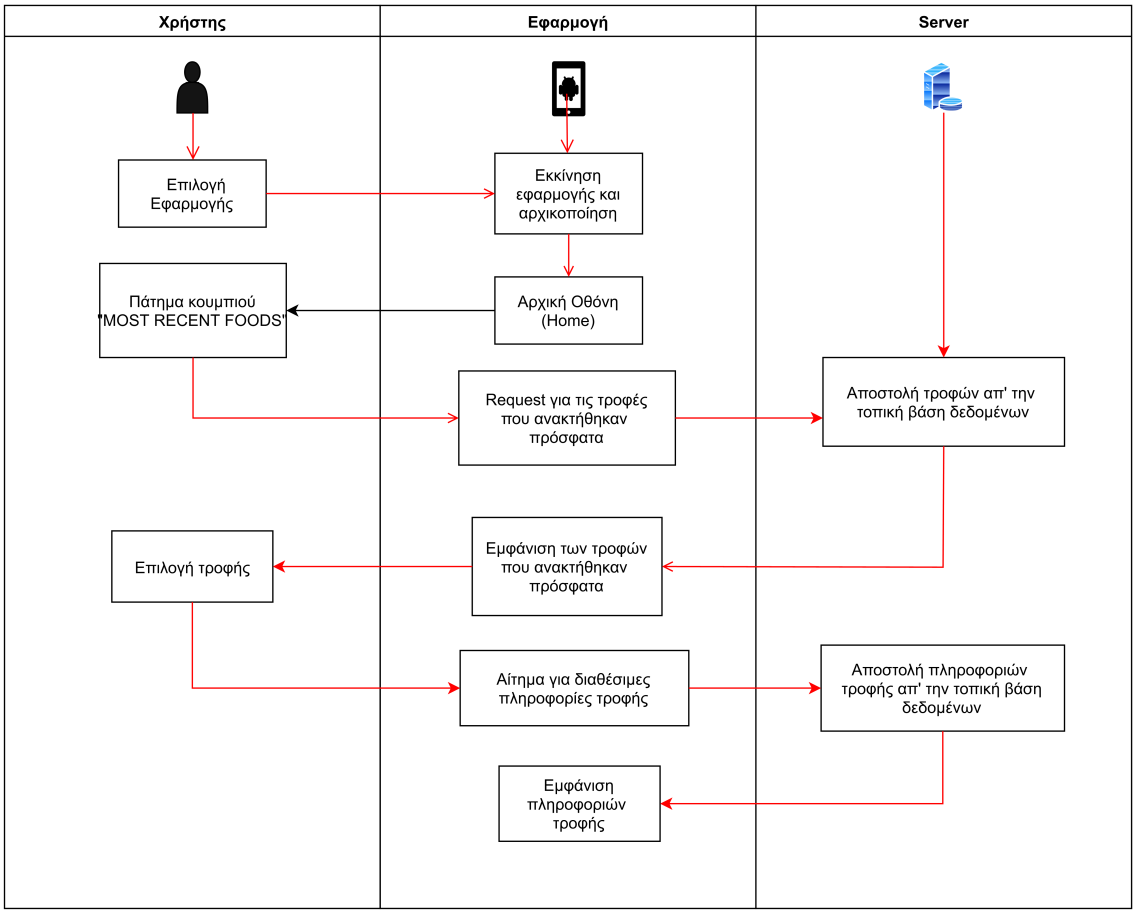
Το σενάριο αυτό αναπαριστά τη διαδικασία που ακολουθείται για την αναζήτηση από το χρήστη και την προβολή των διατροφικών στοιχείων του γεύματος



Εικόνα 3.1: *Σενάριο αναζήτησης διατροφικών στοιχείων*

3.3.2 Προβολή διατροφικών στοιχείων πρόσφατων αναζητήσεων

Το σενάριο αυτό αναπαριστά τη διαδικασία που ακολουθείται για την προβολή των διατροφικών στοιχείων των πιο πρόσφατων αναζητήσεων που έκανε ο χρήστης.



Εικόνα 3.2: *Σενάριο προβολής τροφών πρόσφατων αναζητήσεων*

Κεφάλαιο 4

Σχεδίαση

4.1 Αρχιτεκτονική Εφαρμογής

4.1.1 Γενικές αρχές

Όπως συμβαίνει με κάθε σύστημα, για τη σχεδίαση της εφαρμογής λήφθηκαν υπόψιν κάποιες γενικές αρχές.

- Απλότητα: Για να αποφευχθούν μελλοντικά προβλήματα σχεδιασμού και debugging, έγινε προσπάθεια η εφαρμογή να κατασκευαστεί με τις πιο δημοφιλείς, άμεσες και απλές ενότητες και διαθέσιμους πόρους. Ο στόχος ήταν να παραμείνει εύκολο για οποιονδήποτε να μπορεί να εμπλακεί με το project γρήγορα καθώς και να συντηρηθεί.

- Συντηρησιμότητα (Maintainability): Η βασική ιδέα είναι να χτιστεί η εφαρμογή με όσο το δυνατόν λιγότερες εξαρτήσεις από τον ¨έξω κόσμο¨. Για να επιτευχθεί αυτό, αποφεύγουμε βιβλιοθήκες οι οποίες μπορεί να καταργηθούν κάποια στιγμή, καθιστώντας κατά κάποιο τρόπο και τις λειτουργίες τους καταργημένες. Σημαντικοί παράγοντες για την επιλογή βιβλιοθηκών είναι π.χ. ο αριθμός των χρηστών τους ή το κατά πόσο υπάρχει ισχυρή κοινότητα πίσω από αυτές.

- Ευελιξία: Αυτή η αρχή αφορά κυρίως την ευκολία αλλαγής κι αντικατάστασης μερών της. Η διάσπαση των πραγμάτων σε στρώματα αποβαίνει σχεδόν πάντα επωφελής σε αυτή την κατεύθυνση, και ιδίως με τον καιρό όταν προστίθενται νέα χαρακτηριστικά και λειτουργίες. Ένας τέτοιος τρόπος είναι ο διαχωρισμός των λογικών λειτουργιών από το layer της Διεπαφής του χρήστη (UI) και η μετακίνησή τους σε κάτι σαν Presenter ή View. Όταν η λογική διαχωρίζεται κατάλληλα, η αλλαγή ενός μέρους μιας εφαρμογής χωρίς να επηρεάζει τα υπόλοιπα γίνεται εύκολη.

4.1.2 Model - View - View Model

Η αρχιτεκτονική εφαρμογών διαδραματίζει σημαντικό ρόλο στη δόμηση μιας χαλαρά συνδεδεμένης βάσης κώδικα. Βοηθά στη συγγραφή εύκολα ελέγξιμου, επεκτάσιμου και αποσυνδεδεμένου κώδικα. Με απλά λόγια, η αρχιτεκτονική αναφέρεται στη συνολική οργάνωση του κώδικα σε πράγματα όπως οι αρμοδιότητες της κάθε κλάσης, η οργάνωση φακέλων και η δομή του κώδικα, συμπεριλαμβανομένων των κλήσεων δικτύου, των απαντήσεων, των σφαλμάτων.

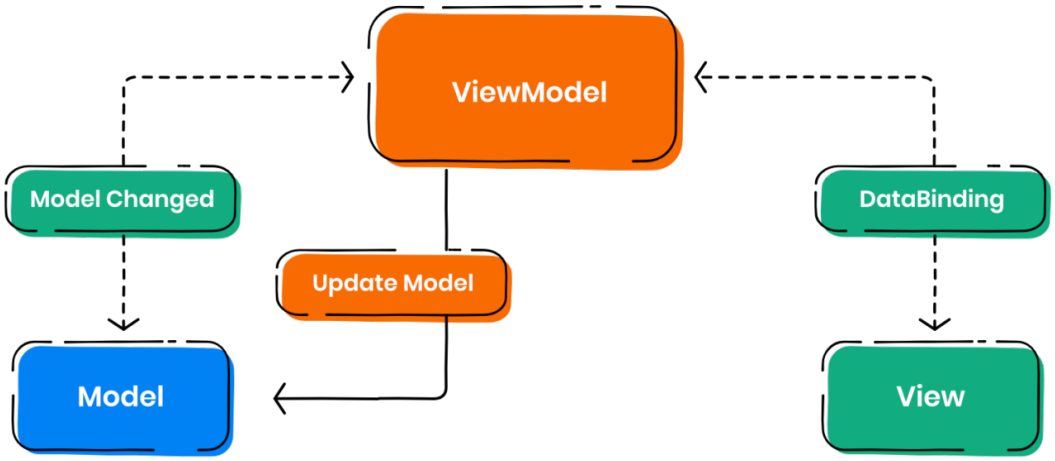
Η αρχιτεκτονική της εφαρμογής βασίστηκε στο πρότυπο Model - View - View Model (MVVM), ένα απ΄ τα συνιστώμενα από την ίδια τη Google και πιο δημοφιλή για εφαρμογές Android. Το πρότυπο εστιάζει τον διαχωρισμό της ανάπτυξης της γραφικής διεπαφής χρήστη από την ανάπτυξη της επιχειρηματικής λογικής ή της λογικής του back-end, έτσι ώστε η προβολή να μην εξαρτάται από κάποια συγκεκριμένο μοντέλο. Οι βασικές δομές του προτύπου αυτού είναι:

- Model: Κλάσεις δεδομένων για την μοντελοποίηση του πραγματικού κόσμου. Τα models είναι στοιχεία που είναι υπεύθυνα για το χειρισμό των δεδομένων μιας εφαρμογής. Είναι ανεξάρτητα από τα αντικείμενα View και τα υπόλοιπα στοιχεία μιας εφαρμογής, οπότε δεν επηρεάζονται από τον κύκλο ζωής της εφαρμογής και τις σχετικές ανησυχίες, όπως π.χ. η «καταστροφή» και η δημιουργία τους κάθε φορά που ανοίγει εκ νέου η εφαρμογή ή περιστρέφεται η οθόνη. [36]

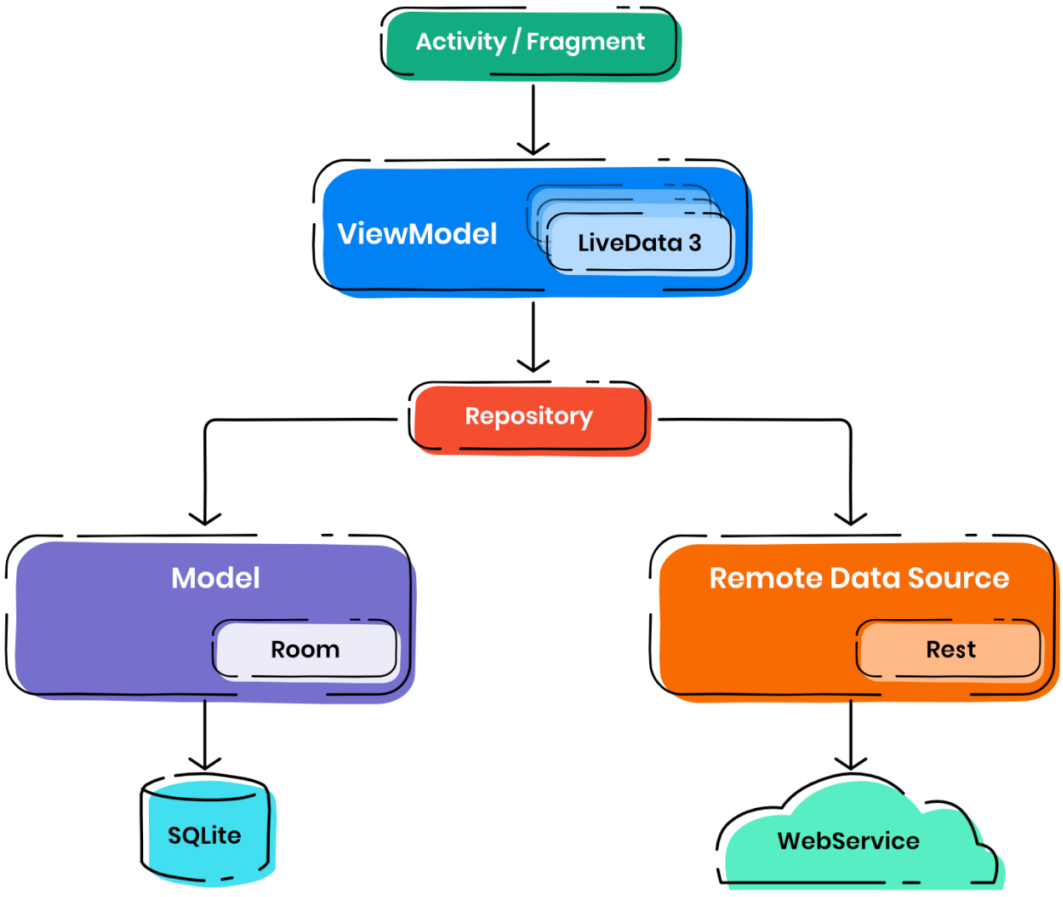
- View: Κλάσεις που δημιουργούν το οπτικό μέρος που σημαίνει ότι όλα όσα βλέπει ο χρήστης εμπίπτουν σε αυτή την κατηγορία. Εμφανίζει μια αναπαράσταση του model, δέχεται την αλληλεπίδραση του χρήστη με την αυτήν και προωθεί τον χειρισμό αυτών στο ViewModel.

- ViewModel: Κλάση που λειτουργεί ως ¨κόλλα' μεταξύ των επιπέδων model και view αλλά λειτουργεί διαφορετικά από το στοιχείο Controller (της αρχιτεκτονικής Model - View - Controller). Εκθέτει εντολές για τη View και συνδέει τη View με το model. Όταν ένα model ενημερώνεται, οι αντίστοιχες Views ενημερώνονται μέσω της σύνδεσης δεδομένων. Το πρότυπο MVVM διαθέτει έναν binder (συνδέτη), ο οποίος αυτοματοποιεί την επικοινωνία μεταξύ μιας View και των δεσμευμένων ιδιοτήτων της στο ViewModel. Ταυτόχρονα, οι συνδέσεις (bindings) λειτουργούν και προς την άλλη κατεύθυνση, όταν ο χρήστης επιδρά με τη View, και ενημερώνουν το model. Στην ουσία, ένα ViewModel οργανώνει την πρόσβαση στη λογική back-end σύμφωνα με τις περιπτώσεις χρήσης από μια View.

Μέρος αυτής της αρχιτεκτονικής αποτελεί και η οργάνωση στην πρόσβαση των δεδομένων. Τα ViewModels μπορούν να κρατήσουν δεδομένα σε περιπτώσεις που χρειάζεται να δημιουργηθεί εκ νέου μια View, π.χ. όταν περιστρέφεται η οθόνη και αλλάζει προσανατολισμό. Ταυτόχρονα, χρησιμοποιούμε αποθετήρια (Repositories) για την παροχή δεδομένων στα ViewModels. Τα αποθετήρια μπορεί να είναι απλά, μπορεί απλώς να αντλούν και να μεταβιβάζουν δεδομένα από ένα δίκτυο, ή μπορεί να κρατούν δεδομένα προσωρινά και να τα αποθηκεύουν τοπικά. Ένα συνηθισμένο μοτίβο είναι η άντληση δεδομένων από το δίκτυο και η τοπική προσωρινή αποθήκευση τους για τη μείωση του αριθμού των κλήσεων δικτύου και την παροχή μιας offline εμπειρίας στους χρήστες.



Εικόνα 4.1: *Η επικοινωνία μες το πρότυπο MVVM*



Εικόνα 4.2: *Αποθετήρια και πρόσβαση στα δεδομένα με MVVM αρχιτεκτονική [1]*

4.2 Οθόνες

Σημαντικό μέρος της σχεδίασης του συστήματος ήταν και ο αρχικός σχεδιασμός των βασικών διεπαφών χρήστη, καθώς βοήθησε στην κατηγοριοποίηση και στην οργάνωση των κύριων λειτουργιών της εφαρμογής αποκτώντας μια πιο ολιστική ματιά. Στη συνέχεια γίνεται μια σύντομη περιγραφή των λειτουργιών, της κάθε οθόνης,

Οθόνη με logo εφαρμογής

Αποτελεί την πρώτη οθόνη με την οποία έρχεται σε επαφή ο χρήστης της εφαρμογής η οποία λειτουργεί σαν εισαγωγή στην εφαρμογή παρουσιάζοντας το logo της εφαρμογής και μετά από λίγο χρόνο φορτώνει την αρχική οθόνη.

	Όνομα	***Τύπος***	***Περιγραφή***
Κλάση	MainActivity	UI View	
Μεθόδοι	openHomeActivity	() → Void	Δημιουργεί καθυστέρηση και καλεί την κλάση της αρχικής οθόνης.

Αρχική οθόνη

Η οθόνη αυτή εμφανίζεται αμέσως μετά, αφού δηλαδή φορτωθεί η οθόνη εισαγωγής. Στο σημείο αυτό, ο χρήστης έχει τη δυνατότητα να πληκτρολογήσει την είσοδο με τις τροφές και τις αντίστοιχες ποσότητες που κατανάλωσε και να πραγματοποιήσει αναζήτηση σε απομακρυσμένη βάση δεδομένων ή να μεταβεί στην οθόνη με τις αποθηκευμένες τροφές στην τοπική βάση δεδομένων.

	Όνομα	***Τύπος***	***Περιγραφή***
Κλάση	HomeActivity	UI View	
Μεθόδοι	onClick	(View) → Void	Μετάβαση στην οθόνη αναγνώρισης τροφών εισόδου ή προσφάτων αναζητήσεων με τα κουμπιά “Submit” και “Most Recent Foods” αντίστοιχα.

Οθόνη Εμφάνισης λίστας με τις τροφές εισόδου

Η οθόνη αυτή εμφανίζεται μετά το πάτημα του κουμπιού "Submit". Εδώ παρουσιάζεται το αποτέλεσμα της «εξαγωγής» των τροφών και των αντίστοιχων ποσοτήτων από την πρόταση

που πληκτρολόγησε σαν είσοδο ο χρήστης, με χρήση της εφαρμογής Python. Ταυτόχρονα εμφανίζονται τα συνολικά διατροφικά στοιχεία του γεύματος, καθώς και για κάθε τροφή ξεχωριστά, εφόσον ο χρήστης έχει επιλέξει για την καθεμία το κατάλληλο αποτέλεσμα.

	Όνομα	***Τύπος***	***Περιγραφή***
Κλάση	AllFoodsActivity	UI View	
Μεθόδοι	getParseFoodsRequestParams	() → Void	Μετατρέπει το JSONObject σε κατάλληλη μορφή text.
	updateTotalFood	(Int, ParseFood) → Void	Εμφανίζει σαν κείμενο τις ποσότητες των διατροφικών στοιχείων
	parseFoods	() → Void	Με τη χρήση των δύο πάνω, εμφανίζει τα αποτελέσματα που επέστρεψε το Web Service.

Οθόνη εμφάνισης των αποτελεσμάτων από την online βάση δεδομένων

Η οθόνη αυτή εμφανίζεται αφότου ο χρήστης επιλέξει να πραγματοποιήσει αίτημα στην online βάση δεδομένων για κάποια από τις τροφές της λίστας. Εδώ παρουσιάζονται τα διάφορα αποτελέσματα που υπάρχουν στη βάση (ανάλογα με την ποικιλία, τον τρόπο μαγειρέματος και άλλους παράγοντες) από τα οποία καλείται ο χρήστης να διαλέξει ποιο του ταιριάζει. Αυτό το βήμα είναι απαραίτητο διότι όλες οι βάσεις δεδομένων διαθέτουν για κάθε τροφή παραπάνω από μια καταχώρηση οι οποίες μπορεί να προκύπτουν από διαφορές στην ποικιλία της τροφής, στην επεξεργασία της, στον τρόπο μαγειρέματος ή μη, στη μορφή που βρίσκεται όταν καταναλώνεται κ.α. Αυτές οι διαφορές αποτελούν παράγοντες που δε μπορούν ούτε να προβλεφθούν καθώς και ούτε να παραβλεφθούν. Επίσης, είναι πιθανό για τον ίδιο χρήστη να μην είναι ίδιοι κάθε φορά.

	Όνομα	***Τύπος***	***Περιγραφή***
Κλάση	NutritionResultActivity	UI View	
Μεθόδοι	searchFoodNutrition	() → Void	Εμφανίζει τα αποτελέσματα από την online βάση δεδομένων για την τροφή που επιλέχθη
	saveParseFood	(FoodNutrition) → Void	Αποθηκεύει τα διατροφικά στοιχεία της καταχώρησης που επέλεξε ο χρήστης και τον επιστρέφει στην οθόνη AllFoodsActivity

Οθόνη εμφάνισης προσφάτως αναζητημένων τροφών από την τοπική βάση δεδομένων

Η οθόνη αυτή εμφανίζεται με το πάτημα κατάλληλου κουμπιού από την αρχική οθόνη. Εδώ παρουσιάζεται μια λίστα με τις τροφές που αναζήτησε πρόσφατα ο χρήστης, η οποία είναι αποθηκευμένη σε τοπική βάση δεδομένων. Ο χρήστης μπορεί να επιλέξει για ποια από τις τροφές επιθυμεί να δει τα διατροφικά στοιχεία στα οποία πρόσφατα απέκτησε πρόσβαση. Θεωρήθηκε θεμιτή η ύπαρξη αυτής της λειτουργίας για την επίτευξη της εκτός δικτύου κι εύκολης πρόσβασης σε τροφές που καταναλώνονται καθημερινά από το χρήστη.

	Όνομα	***Τύπος***	***Περιγραφή***
Κλάση	RecentAllFoodsActivity	UI View	
Μεθόδοι	getRecentParseFoods	() → Void	Εμφανίζει τις πρόσφατες αναζητήσεις από την τοπική βάση δεδομένων

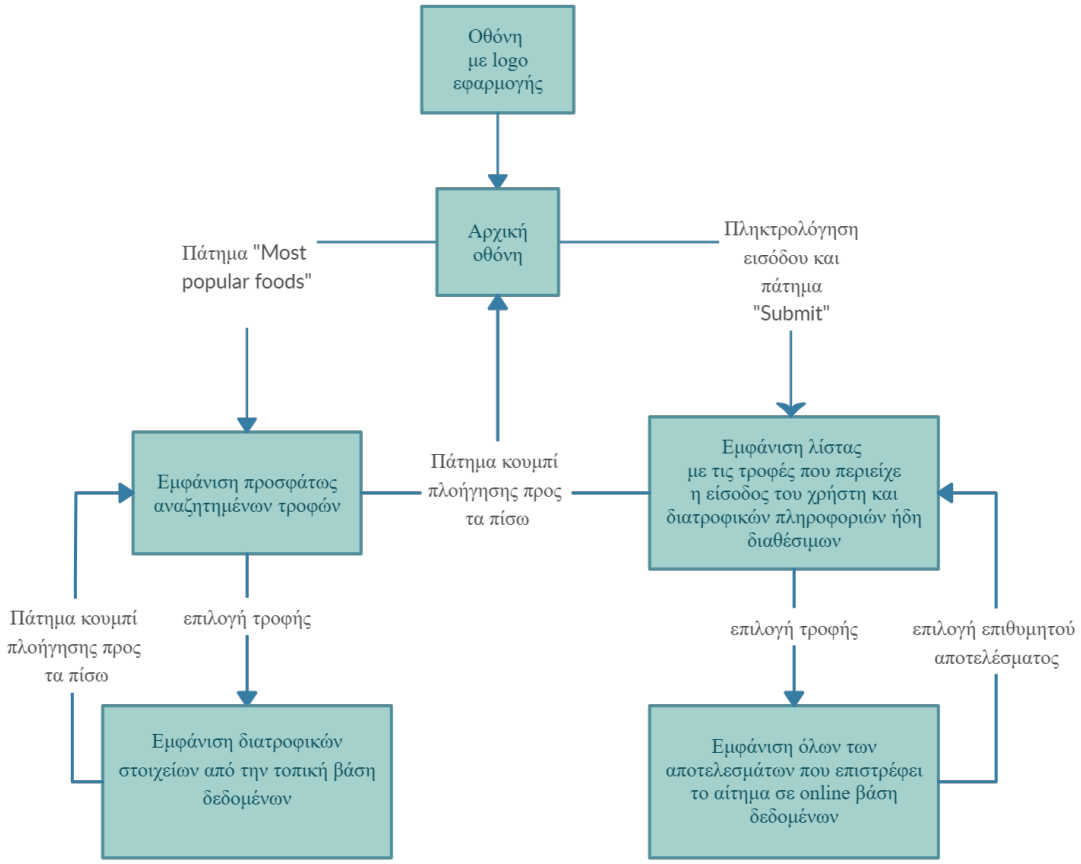
Οθόνη εμφάνισης διατροφικών στοιχείων από την τοπική βάση δεδομένων

Η οθόνη αυτή εμφανίζεται αφότου ο χρήστης επιλέξει τροφή από την οθόνη προσφάτως αναζητημένων τροφών. Σκοπός είναι να εμφανίζονται τα διατροφικά στοιχεία για τη συγκεκριμένη τροφή που επέλεξε ο χρήστης.

	Όνομα	***Τύπος***	***Περιγραφή***
Κλάση	RecentParseFoodsActivity	UI View	
Μεθόδοι	getRecentParseFoods	() → Void	Εμφανίζει τα διατροφικά στοιχεία της τροφής που επιλέχθη από τη τοπική βάση δεδομένων

4.2.1 Δομή οθονών

Η δομή των οθονών της εφαρμογής είναι απλή και μπορεί να γίνει εύκολα κατανοητή με το παρακάτω διάγραμμα.



Εικόνα 4.3: *Διάγραμμα Ροής των οθονών της Εφαρμογής*

4.3 Πρόσβαση και Ανάκτηση από την FoodData Central

Η επικοινωνία μεταξύ της εφαρμογής και της βάσης δεδομένων γίνεται μέσω του RESTful API που η ίδια παρέχει. Η εφαρμογή στέλνει αιτήματα και λαμβάνει απαντήσεις με τη χρήση της βιβλιοθήκης OkHttpClient, φτιαγμένη για τη διαχείριση HTTP κλήσεων. Έπειτα, τις απαντήσεις που λαμβάνει τις αποθηκεύει σε JSONObject τύπο. Η JSON (JavaScript Object Notation) είναι μια ελαφριά μορφή ανταλλαγής δεδομένων και τη χρησιμοποιούμε συνήθως για την επικοινωνία client-server. Είναι εύκολη στην ανάγνωση κι εγγραφή και ανεξάρτητη από γλώσσες. Μια τιμή JSON μπορεί να είναι JSONobject, πίνακας, αριθμός, συμβολοσειρά, boolean (true/false) ή απλά null ενώ, ένα JSONObject είναι ουσιαστικά μια μη ταξινομημένη συλλογή από ζεύγη κλειδιών και τιμών. Στη Java, η JSON είναι διαθέσιμη μέσω της βιβλιοθήκης org.json η οποία μας παρέχει κλάσεις για την ανάλυση και τον χειρισμό της. [37]

Η παραπάνω επικοινωνία γίνεται μέσω API endpoints. Πιο συγκεκριμένα, χρησιμοποιούμε το "Food Search" endpoint το οποίο επιστρέφει λίστα με τα φαγητά που ταιριάζουν

στις λέξεις-κλειδιά (αίτημα). Οι κλήσεις λοιπόν γίνονται στο σύνδεσμο `https://api.nal.usda.gov/fdc/v1/foods/search` με τη χρήση ενός API key που λαμβάνουμε έπειτα από δωρεάν εγγραφή. Για παράδειγμα, με το συγκεκριμένο API key που λάβαμε και για την τροφή κινόα (quinoa), το αίτημα θα ήταν έτσι: `https://api.nal.usda.gov/fdc/v1/foods/search?api_key=gPbs9TDWvay8M1YhRxbPzRh3apizCFNZm563jWXd&query=quinoa`. Από τα διαθέσιμα στοιχεία της βάσης δεδομένων, επιλέγουμε για λόγους ευκολίας και ευχρηστίας να παρουσιάζουμε τα εξής καθώς και τις μονάδες μέτρησης του καθενός (g,ml,kcal κ.λπ.): Energy, Water, Protein, Total lipid (fat), Carbohydrates, Sugars, Fatty acids saturated, Fatty acids monounsaturated, Fatty acids polyunsaturated, Cholesterol. Επιπλέον, η FoodData Central δεν μας παρέχει τρόπο αυτόματης τροποποίησης των μεγεθών των μερίδων και συνεπώς, αυτό θα πρέπει να γίνεται τοπικά πριν την παρουσίαση στο χρήστη.

4.4 Τοπική Βάση δεδομένων

Για την υλοποίηση της τοπικής βάσης δεδομένων, χρησιμοποιήθηκε η βιβλιοθήκη Room, που προσφέρει ένα στρώμα αφαίρεσης πάνω από την SQLite, και συγκεκριμένα η έκδοση 1.0.0.

Η βάση δεδομένων περιέχει ένα πίνακα με τις πρόσφατες αναζητήσεις σε μορφή κειμένου και τα αντίστοιχα διατροφικά στοιχεία. Οι οντότητες (πεδία) είναι τα στοιχεία που επιλέγουμε να εμφανίζουμε από την FoodData Central.

4.5 Εφαρμογή Python και ενσωμάτωση Μηχανικής Μάθησης

Για την οθόνη εμφάνισης λίστας με τις τροφές που περιείχε η είσοδος του χρήστη, γράφτηκε μια εφαρμογή-υπηρεσία σε γλώσσα Python με σκοπό να εξάγει από το κείμενο τα στοιχεία τα οποία είναι απαραίτητα για την ανάκτηση των διατροφικών στοιχείων του γεύματος. Δεν ζητείται από το χρήστη να καταχωρήσει την είσοδο με κάποιο πολύ συγκεκριμένο τρόπο. Οι οδηγίες προτείνουν στο χρήστη να πληκτρολογήσει την είσοδο σε μορφή πρότασης σε αγγλική γλώσσα. Για την υλοποίηση της, χρησιμοποιήθηκε μέρος των αλγορίθμων και των διαφόρων επιμέρους βιβλιοθηκών της NLTK. Έτσι, η ταξινόμηση της εισόδου επιτεύχθηκε μέσω μηχανικής μάθησης, εφαρμογής γλωσσολογικών κανόνων και υπολογιστικής επεξεργασίας φυσικής γλώσσας.

Η πρώτη φάση της επεξεργασίας ξεκινάει με το Tokenization της εισόδου στις επιμέρους λέξεις και την τοποθέτησή τους σε μια λίστα. Αυτό επιτυγχάνεται με την συνάρτηση word_tokenize() του πακέτου nltk.tokenize. Όπως εξηγήθηκε στο *2.7.1*, η υλοποίηση γίνεται με τον συνδυασμό δύο tokenizer αλγορίθμων. Ο πρώτος είναι μια βελτιωμένη έκδοση του αλγόριθμου TreebankWordTokenizer ο οποίος χρησιμοποιεί κανονικές εκφράσεις. Συνοπτικά, τα βήματα που ακολουθεί είναι ότι χωρίζει τα σημεία στίξης σαν ξεχωριστά tokens κι έπειτα χωρίζει σε επιμέρους tokens ανάμεσα σε κόμματα και κενά.[38] Ο δεύτερος είναι ο PunktSentenceTokenizer, ο οποίος παρέχει ένα ήδη εκπαιδευμένο μοντέλο που ενσωματώνει τον αλγόριθμο Kiss and Strunk (2005) για Unsupervised Multilingual Sentence Boundary Detection.[39]. Η λογική αυτού είναι ότι χωρίζει το κείμενο σε λίστα από προτάσεις με τη χρήση

"Unsupervised" εκμάθησης, για τη δημιουργία ενός μοντέλου για συντομογραφίες, φράσεις και λέξεις που ξεκινούν προτάσεις.

Ύστερα, απαραίτητο είναι και το POS Tagging της κάθε λέξης με σκοπό την αναγνώριση του ρόλου της μες την πρόταση. Η συνάρτηση pos_tag() της NLTK υλοποιεί έναν Greedy Averaged Perceptron tagger-αλγόριθμο. Είναι εκπαιδευμένος και δοκιμασμένος στα μέρη 00-18 από τα Wall Street Journal μέρη του project-πακέτου OntoNotes 5.0 προσβάσιμο στον σύνδεσμο `https://catalog.ldc.upenn.edu/LDC2013T19`. Πρόκειται για supervised εκμάθηση δυαδικών ταξινομητών, όπου από τα δεδομένα μας λείπει η πληροφορία POS Tag. Ο Greedy Averaged αλγόριθμος έχει ένα λεξικό με βάρη συνδεδεμένα με τα διάφορα χαρακτηριστικά, τα οποία χρησιμοποιεί για να προβλέψει το σωστό tag. Κατά την εκπαίδευση του, ανάλογα με τις προβλέψεις και αν είναι σωστές ή λάθος, ο αλγόριθμος προσαρμόζει κατάλληλα αυτά τα βάρη προσθέτοντας ή αφαιρώντας +-1 αντίστοιχα, για την κατάλληλη κατηγορία. Στο τέλος, τα βάρη αυτά γίνονται "averaged" ανάλογα με τον αριθμό των επαναλήψεων και τον αριθμό των δεδομένων. [40]

Στη συνέχεια, πραγματοποιείται Lemmatization για την ¨κανονικοποίηση' των λέξεων, όπως περιγράφηκε στο *2.7.1* δηλαδή με κανόνες γλωσσολογίας και πιο συγκεκριμένα με έναν συνδυασμό κανόνων κλιτικών καταλήξεων μαζί με λίστες εξαιρέσεων.

Σε δεύτερη φάση και συνδυαστικά με τη χρήση των POS Tags που δημιουργήθηκαν νωρίτερα, αρχίζει η κατηγοριοποίηση και η αποθήκευση των απαραίτητων λέξεων στις κατηγορίες ποσότητα, λέξεις μέτρησης και τροφή. Η ποσότητα αναγνωρίζεται μέσω του POS Tagging που έχει ήδη γίνει. Οι λέξεις μέτρησης ξεχωρίζονται συγκρίνοντας τις λέξεις με μεγάλες λίστες που παρέχουν οι βιβλιοθήκες με λέξεις που αφορούν τη μέτρηση τροφής. Τέλος, για να καθορίσουμε αν είναι τροφή μια λέξη, με βάση το POS Tag της, χρησιμοποιούμε τα synsets της βιβλιοθήκης WordNet ελέγχοντας τα synsets που επιστρέφει η λέξη και διάφορα synsets που σχετίζονται με τροφές ή και μη για την απόρριψη μιας λέξης.

4.6 Docker

Για το deployment του Python Web Service στο Google Cloud Run (GCR) μέσω του Flask Framework, αλλά και για το localstack προσομοίωσης αυτού με σκοπό την ανάπτυξη και τον έλεγχο της, χρησιμοποιήθηκε η τεχνολογία του docker. Πιο συγκεκριμένα, δημιουργείται ένα docker container image που να περιλαμβάνει την εφαρμογή Flask, σύμφωνα με τις οδηγίες που δίνουμε στο dockerfile. Για το GCR, δημιουργείται με το Cloud build ενώ για το χειρισμό του localstack αξιοποιήθηκε το εργαλείο Docker Desktop το οποίο επιτρέπει τον ορισμό και εκτέλεση του containter σε local server.

4.7 Google Cloud Run

Για το deployment του docker container με την εφαρμογή Python, δημιουργήθηκε ένα project στο GCP και μέσα σε αυτό το Cloud Run service με το container. Παίρνει τη μορφή του Web App ώστε να είναι προσβάσιμη από οποιοδήποτε κινητό και μέρος. Αυτό επιτυγχάνεται μέσω του συνδέσμου `https://food-model-service-wwtxo274zq-lm.a.run.app`

ενώ τα POST requests γίνονται στο σύνδεσμο `https://food-model-service-wwtxo274zq-lm.a.run.app/api/parse_food`.

4.8 Flask - RESTful API

Η επικοινωνία μεταξύ της εφαρμογής Android και της Python Web εφαρμογής, που απαιτείται για την εξαγωγή των τροφών απ΄ την είσοδο του χρήστη, πραγματοποιείται με HTTP requests/responses μέσω ενός restful API. Το API αυτό δημιουργείται από το framework Flask το οποίο κάνει δυνατό και το deployment σε server. Στη συνέχεια, παρουσιάζεται ένα χαρακτηριστικό στιγμιότυπο της λεπτομερής σχεδίασης του endpoint. Το ακόλουθο documentation δημιουργήθηκε με τη χρήση της πλατφόρμας Postman και προβάλλεται μέσω ενός browser.

Python Flask API

POST **Parse Sentence with Meal Details**

https://food-model-service-wwtxo274zq-lm.a.run.app/api/parse_food

Foods breakdown

HEADERS

Content-Type	application/json

BODY raw

```
{
    "text": "sentence with meal details"
}
```

Example Request https://food-model-service-wwtxo274zq-lm.a.run.app/api/parse_food

```
curl --location --request POST 'https://food-model-service-wwtxo274zq-lm.a.run.app/api/parse_food' \
--header 'Content-Type: application/json' \
--data-raw '{
    "text": "I ate 100 g of minced meat and had 100 ml of orange juice"
}'
```

Example Response 200 OK

Body Header (6)


```
{
  "parsed": [
    [
      [
        "100"
      ],
      [
        "g"
      ],
      [
        "minced meat"
      ]
    ],
    [
      [
        "100"
      ],
      [
        "ml"
      ],
      [
        "orange juice"
      ]
    ]
  ]
}
```


Εικόνα 4.4: *Flask API documentation*

Κεφάλαιο 5

Ανάπτυξη

5.1 Ανάπτυξη Front-End

5.1.1 Δομή Συστήματος και Ενδεικτικές Λειτουργίες Κλάσεων

Η εφαρμογή, όπως αναφέρθηκε στο σχεδιασμό, ξεκινάει με μια εισαγωγική οθόνη ενώ η πρώτη λειτουργική οθόνη που βλέπει ο χρήστης είναι αυτή της **HomeActivity**. Στον κώδικα παρακάτω βλέπουμε ενδεικτικά τη μέθοδο δημιουργίας της οθόνης και μετά τη μέθοδο διαχείρισης των δύο κουμπιών, με τα οποία μεταβαίνει σε άλλη οθόνη.

```
class HomeActivity : BaseActivity(), View.OnClickListener {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setFullScreen()
        setContentView(R.layout.activity_home)

    }

    override fun onClick(v: View) {
        when (v.id) {
            R.id.btnSubmit -> {
                if (isEditTextEmpty(searchFood)) setErrorFieldIsRequired(
↪ searchFood)
                else if ((application as App).isConnected()) {
                    startActivity(Intent(this, AllFoodsActivity::class.java
↪ ).putExtra(AllFoodsActivity.PARAM_FOOD, searchFood.text.toString()).
↪ addFlags(Intent.FLAG_ACTIVITY_NEW_TASK))
                } else Toast.makeText(this, "No internet connection!",
↪ Toast.LENGTH_SHORT).show()
            }
            R.id.btnRecent -> {
                startActivity(Intent(this, RecentParseFoodsActivity::class.
↪ java).addFlags(Intent.FLAG_ACTIVITY_NEW_TASK))
            }
        }
    }
}
```

Σαν παράδειγμα υλοποίησης της αρχιτεκτονικής Model - View - ViewModel, βλέπουμε τη λειτουργία της AllFoodsActivity. Αρχικά έχουμε το μοντέλο **FoodNutrition**:

```
class FoodNutrition(private val jsonObject: JsonObject) {

    var description: String = ""
    var energy: Float = 0f
    var energyUnitName = ""
    var water: Float = 0f
```

```
    var waterUnitName = ""
    var protein: Float = 0f
    var proteinUnitName = ""
    var totalLipid: Float = 0f
    var totalLipidUnitName = ""
    var carbohydrate: Float = 0f
    var carbohydrateUnitName = ""
    var sugars: Float = 0f
    var sugarsUnitName = ""
    var fattyAcidsTotalSaturated: Float = 0f
    var fattyAcidsTotalSaturatedUnitName = ""
    var fattyAcidsTotalMonounsaturated: Float = 0f
    var fattyAcidsTotalMonounsaturatedUnitName = ""
    var fattyAcidsTotalPolyunsaturated: Float = 0f
    var fattyAcidsTotalPolyunsaturatedUnitName = ""
    var cholesterol: Float = 0f
    var cholesterolUnitName = ""


    init {
        description = jsonObject.get("description").asString
.
.
.
```

Παρακάτω βλέπουμε συνοπτικά την κλάση τύπου view της **AllFoodsActivity**

```
@AndroidEntryPoint
class AllFoodsActivity : BaseActivity(), View.OnClickListener {

    @Inject
    lateinit var allFoodsAdapter: AllFoodsAdapter

    private val viewModel: AllFoodsViewModel by viewModels()

    companion object {
        var  instance: AllFoodsActivity? = null
        const val PARAM_FOOD = "food"
    }

    override fun onCreate(savedInstanceState: Bundle?) {...}

    override fun onClick(v: View) {...}

    override fun onStart() {...}

    override fun onDestroy() {...}

    private fun prepareViews() {
        val layoutManager = LinearLayoutManager(this)
        layoutManager.orientation = RecyclerView.VERTICAL
        recyclerView.layoutManager = layoutManager
        recyclerView.adapter = allFoodsAdapter
    }

    private fun parseFoods() {
        if ((application as App).isConnected()) {
            viewModel.parseFood(getParseFoodsRequestParams()).observe(this,
↪   Observer {
                when (it.status) {
                    Resource.Status.SUCCESS -> {
                        progressBar.visibility = View.GONE
                        if (it.data?.has("parsed") == true) {
                            val parsed = it.data.getAsJsonArray("parsed")
                            var isResponseOkay = true
                            for (pos in 0 until parsed.size()) {
                                val quantity = parsed.get(pos).asJsonArray.
↪ get(0).asJsonArray.get(0).asString
                                if (!isNumeric(quantity)) {
```

```
                                    isResponseOkay = false
                                    break
                                }
                            }

                            if (!isResponseOkay){
                                AlertDialog.Builder(this)
                                    .setMessage("Please use numbers for the
    ↪  quantity of foods.")
                                    .setTitle("Issue")
                                    .setPositiveButton("Okay") { dialog,
    ↪ which ->
                                        onBackPressed()
                                        dialog.dismiss()
                                    }.show()
                                return@Observer
                            }
                            val parsedFoodsList = ParseFood.fromJson(it.
    ↪ data)
                            allFoodsAdapter.addNewData(parsedFoodsList)
                            allFoodsAdapter.notifyDataSetChanged()
                            updateTotalFood()
                        }
                    }
                    Resource.Status.LOADING -> {
                        progressBar.visibility = View.VISIBLE
                    }
                    Resource.Status.ERROR -> {
                        progressBar.visibility = View.GONE
                    }
                }
            })
        }
    }

    private fun getParseFoodsRequestParams(): RequestBody {
        val map = HashMap<String, String>()
        map["text"] = intent.getStringExtra(PARAM_FOOD).toString()
        return RequestBody.create(
            MediaType.parse("application/json; charset=utf-8"),
            JSONObject(map as Map<*, *>).toString()
        )
    }

    public fun updateNutrition(position: Int, parseFood: ParseFood) {
        allFoodsAdapter.updateData(parseFood, position)
        allFoodsAdapter.notifyItemChanged(position)
        updateTotalFood()
    }

    private fun updateTotalFood() {
        val parseFood = allFoodsAdapter.getTotalNutrition()
        energyNutrition.text = "Energy: " + parseFood.energy + " " +
    ↪ parseFood.energyUnitName
        waterNutrition.text = "Water: " + parseFood.water + " " + parseFood
    ↪ .waterUnitName

.
.
.
```

Σαν binder λειτουργεί η κλάση **AllFoodsAdapter**, η οποία αυτοματοποιεί την επικοινωνία μεταξύ της View και των ιδιοτήτων της ViewModel.

```
class AllFoodsAdapter @Inject constructor(private var parseFoodList:
    ↪ ArrayList<ParseFood>) :
    RecyclerView.Adapter<RecyclerView.ViewHolder>() {
```

```
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
↪ RecyclerView.ViewHolder {
    return AllFoodsViewHolder(
        LayoutInflater.from(parent.context).inflate(R.layout.
↪ item_all_food_rows, parent, false) )
}

override fun getItemCount(): Int {
    return parseFoodList.size
}

override fun onBindViewHolder(holder: RecyclerView.ViewHolder, position
↪ : Int) {
    (holder as AllFoodsViewHolder).bindViews(parseFoodList[position])
}

inner class AllFoodsViewHolder(itemView: View) : RecyclerView.
↪ ViewHolder(itemView) {
    init {
        itemView.setOnClickListener(View.OnClickListener {
        openNutritionResultActivity(parseFoodList[adapterPosition])
        })
    }

    fun bindViews(parseFood: ParseFood) {
        //Item Food Name
        if (parseFood.isInputError){
            itemView.itemName.text = parseFood.tempUnit
            itemView.itemInputError.visibility = View.VISIBLE
        }
        else
        {
            itemView.itemName.text = parseFood.food + " " + parseFood.
↪ quantity +" "+ parseFood.unit
            itemView.itemInputError.visibility = View.GONE
        }
        //Food Nutrition
        itemView.energyNutrition.text = "Energy: " + parseFood.energy +
↪  " " + parseFood.energyUnitName
        itemView.waterNutrition.text = "Water: " + parseFood.water + "
↪ " + parseFood.waterUnitName
        .
        .
        }

    private fun openNutritionResultActivity(food: ParseFood) {
        itemView.context.startActivity(
            Intent(itemView.context, NutritionResultActivity::class.
↪ java)
                .addFlags(Intent.FLAG_ACTIVITY_NEW_TASK)
                .putExtra(NutritionResultActivity.PARAM_POS,
↪ adapterPosition)
                .putExtra(NutritionResultActivity.PARAM_FOOD, food.food
↪ )
                .putExtra(NutritionResultActivity.PARAM_FOOD_QUANTITY,
↪ food.quantity)
                .putExtra(NutritionResultActivity.PARAM_FOOD_UNIT, food
↪ .unit) )
    }
}

fun addNewData(parseFoodList: ArrayList<ParseFood>) {
    this.parseFoodList = parseFoodList}

fun updateData(parseFood: ParseFood,position: Int){
    this.parseFoodList[position] = parseFood}

fun getTotalNutrition() : ParseFood{
    var energy: Float = 0f
    var energyUnitName = ""
```

```
        var water: Float = 0f
        .
        .

        for (food in parseFoodList){
            energy += food.energy
            water += food.water
            .
            .
            .
        }

         return ParseFood(0,"","","","", false, energy, energyUnitName,
    ↪ water, waterUnitName, protein, proteinUnitName, totalLipid,
    ↪ totalLipidUnitName, carbohydrate, carbohydrateUnitName, sugars,
    ↪ sugarsUnitName, fattyAcidsTotalSaturated,
    ↪ fattyAcidsTotalSaturatedUnitName, fattyAcidsTotalMonounsaturated,
    ↪ fattyAcidsTotalMonounsaturatedUnitName,
    ↪ fattyAcidsTotalPolyunsaturated,
    ↪ fattyAcidsTotalPolyunsaturatedUnitName, cholesterol,
    ↪ cholesterolUnitName)
    }
}
```

Τέλος, η κλάση **AllFoodsViewModel**

```
class AllFoodsViewModel @ViewModelInject constructor(private val repository
    ↪ : ParseFoodRepository) : ViewModel(){
    fun parseFood(@Body params: RequestBody) = repository.parseFood(params)
}
```

καθώς και η **NutritionResultViewModel**, η οποία αποτελεί πιο ενδεικτικό παράδειγμα μιας κλάσης τύπου ViewModel.

```
class NutritionResultViewModel @ViewModelInject constructor( private val
    ↪ repository: SearchNutritionRepository, private val
    ↪ parseFoodRepository: ParseFoodRepository ) : ViewModel() {
    fun searchNutrition(api_key: String,query: String) = repository.
    ↪ searchNutrition(api_key, query)

    fun saveParseFood(parseFood: ParseFood) = parseFoodRepository.
    ↪ saveParseFood(parseFood)
}
```

5.1.2 Διεπαφή

Η διεπαφή σχεδιάστηκε με το δυναμικό εργαλείο Layout Editor του Android Studio το οποίο επεξεργάζεται για κάθε οθόνη το αντίστοιχο αρχείο XML. Παρακάτω φαίνεται ενδεικτικά κώδικας από την αρχική οθόνη.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/white"
    android:orientation="vertical">


    <RelativeLayout
        android:id="@+id/action_bar_menus"
        android:layout_width="match_parent"
```

```
        android:background="@color/main_bcg"
        android:layout_height="?actionBarSize"
        android:orientation="horizontal"
        android:visibility="visible">


        <com.foodnutrition.views.customviews.CustomTextViewBold
            android:id="@+id/tvHeader"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_centerVertical="true"
            android:text="Home"
            android:textColor="@android:color/white"
            android:textSize="20sp" />

    </RelativeLayout>
.
.
.
```

5.1.3 Επικοινωνία με το Back-End

Για την επικοινωνία με το Back-End, δημιουργήσαμε την κλάση **AppModule**, η οποία περιέχει τις βασικές πληροφορίες για τα API endpoints,

```
@Module
@InstallIn(SingletonComponent::class)
object AppModule {

    @Singleton
    @Provides
    @Named("Normal")
    fun provideRetrofit(@ApplicationContext appContext: Context,gson: Gson)
    ↪ : Retrofit = Retrofit.Builder()
        .baseUrl("https://food-model-service-wwtxo274zq-lm.a.run.app/api/")
        .addConverterFactory(GsonConverterFactory.create(gson))
        .build()

    @Singleton
    @Provides
    @Named("Nutrition")
    fun provideRetrofitForNutritionDatabase(@ApplicationContext appContext:
    ↪  Context,gson: Gson):      Retrofit = Retrofit.Builder()
        .baseUrl("https://api.nal.usda.gov/fdc/v1/foods/")
        .addConverterFactory(GsonConverterFactory.create(gson))
        .build()


    @Provides
    fun provideGson(): Gson = GsonBuilder().setLenient().create()

    @Provides
    fun provideParseFoodApiServices(@Named("Normal") retrofit: Retrofit):
    ↪ ParseFoodApiServices =
        retrofit.create(ParseFoodApiServices::class.java)

    @Provides
    fun provideNutritionApiServices(@Named("Nutrition") retrofit: Retrofit)
    ↪ : NutritionApiServices =
        retrofit.create(NutritionApiServices::class.java)

    @Singleton
    @Provides
    fun provideRemoteDataSource(parseFoodApiServices: ParseFoodApiServices,
    ↪ nutritionApiServices:
```

```
    NutritionApiServices) = RemoteDataSource(parseFoodApiServices,
    ↪ nutritionApiServices)

    @Singleton
    @Provides
    fun provideDatabase(@ApplicationContext appContext: Context) =
        LocalDataSource.getDatabase(appContext)

.
.
.
```

καθώς και τη **RemoteDataSource**, η οποία παρέχει πρόσβαση στις συναρτήσεις DAO μέσα από την οποία καλούνται

```
class RemoteDataSource @Inject constructor(private val parseFoodApiServices
    ↪ : ParseFoodApiServices, private val nutritionApiServices:
    ↪ NutritionApiServices) : BaseDataSource() {

    //Parse Food
    suspend fun parseFood(@Body params: RequestBody) = getResult {
    ↪ parseFoodApiServices.parseFood(params) }
    //Search Food Nutrition
    suspend fun searchNutrition(api_key: String,query: String) = getResult
    ↪ { nutritionApiServices.searchNutrition(api_key, query) }
}
```

Python Web App

Η επικοινωνία με το Python Flask API για την κατηγοριοποίηση των τροφών γίνεται με τη **ParseFoodApiServices** και ολοκληρώνεται με την ParseFoodRepository, που παρουσιάζεται αμέσως μετά.

```
interface ParseFoodApiServices {
    @POST("parse_food")
    suspend fun parseFood(@Body  params: RequestBody): Response<JsonObject>
}
```

Τοπική Βάση Δεδομένων

Η πρόσβαση στα DAO της τοπικής βάσης δεδομένων, καθώς και η πρόσβαση στην παραπάνω συνάρτηση της ParseFoodApiServices, επιτυγχάνεται με την κλάση **ParseFoodRepository**.

```
class ParseFoodRepository  @Inject constructor( private val
    ↪ remoteDataSource: RemoteDataSource, private val parseFoodDao:
    ↪ ParseFoodDao ) {

    fun parseFood(@Body params: RequestBody) =
        performOperationFromRemoteOnly( networkCall = {remoteDataSource.
    ↪ parseFood(params)} )

    fun saveParseFood(parseFood: ParseFood) =
        performSaveOperation( saveCallResult = {parseFoodDao.insert(
    ↪ parseFood)} )

    fun getParseFoods() =
        performOperationFromLocalOnly( databaseQuery = {parseFoodDao.
    ↪ getParseFoods()} )
```

```
    fun getParseFoods(id: Int) =
        performOperationFromLocalOnly( databaseQuery = {parseFoodDao.
    ↪ getParseFoods(id)})
}
```

Online Βάση Δεδομένων

Η επικοινωνία με το API της FoodData Central για την ανάκτηση των καταχωρήσεων για την τροφή που διάλεξε ο χρήστης γίνεται με τη **NutritionApiServices**,

```
interface NutritionApiServices {
    @GET("search")
    suspend fun searchNutrition(@Query(value="api_key") api_key: String,
    @Query(value="query") query: String) : Response<JsonObject>
}
```

και την **SearchNutritionRepository**, επιτυγχάνοντας έτσι δομημένη πρόσβαση και δημιουργόντας ένα στρώμα ενθυλάκωσης.

```
class SearchNutritionRepository  @Inject constructor( private val
    ↪ remoteDataSource: RemoteDataSource ) {

    fun searchNutrition(api_key: String,query: String) =
    ↪ performOperationFromRemoteOnly( networkCall = { remoteDataSource.
    ↪ searchNutrition(api_key, query) } )
}
```

5.2 Ανάπτυξη Back-End

5.2.1 Τοπική Βάση Δεδομένων

Για την αποθήκευση προσφάτων αναζητήσεων δημιουργήθηκαν οι παρακάτω οντότητες (entities).

- **ParseFood**: Η σημασία των πεδίων της επεξηγείται από το ίδιο το όνομα τους.

```
@Entity(tableName = "tbl_parse_food")
data class ParseFood(
    @PrimaryKey(autoGenerate = true)
    val id: Int,
    val food: String,
    val quantity: String,
    val unit: String,
    val tempUnit: String="",
    val isInputError: Boolean = false,
    val energy: Float = 0f,
    val energyUnitName: String = "",
    val water: Float = 0f,
    val waterUnitName: String = "",
    val protein: Float = 0f,
    val proteinUnitName: String = "",
    val totalLipid: Float = 0f,
    val totalLipidUnitName: String = "",
    val carbohydrate: Float = 0f,
    val carbohydrateUnitName: String = "",
    val sugars: Float = 0f,
    val sugarsUnitName: String = "",
    val fattyAcidsTotalSaturated: Float = 0f,
    val fattyAcidsTotalSaturatedUnitName: String = "",
    val fattyAcidsTotalMonounsaturated: Float = 0f,
    val fattyAcidsTotalMonounsaturatedUnitName: String = "",
    val fattyAcidsTotalPolyunsaturated: Float = 0f,
    val fattyAcidsTotalPolyunsaturatedUnitName: String = "",
    val cholesterol: Float = 0f,
    val cholesterolUnitName: String = ""
)
```

- **LocalDataSource**: Είναι τύπου RoomDatabase και αποτελείται από οντότητες τύπου ParseFood.

Επίσης, για την πρόσβαση σε αυτά, ακόμα και για σκοπούς εμφάνισης των αποτελεσμάτων από την online βάση δεδομένων, χρησιμοποιείται το αντικείμενο πρόσβασης σε δεδομένα (DAO) **parseFoodDao**:

```
@Dao
interface ParseFoodDao {

    @Query("SELECT * FROM tbl_parse_food order by id DESC LIMIT 20")
    fun getParseFoods() : LiveData<List<ParseFood>>

    @Query("SELECT * FROM tbl_parse_food where id = :id")
    fun getParseFoods(id: Int) : LiveData<List<ParseFood>>

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertAll(parseFoodList: List<ParseFood>)

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insert(parseFood: ParseFood)
}
```

5.2.2 Python Web App και επεξεργασία εισόδου

Στην εφαρμογή Python, η επεξεργασία του κειμένου γίνεται από την ενότητα **Ingredients.py**. Περιέχει τις παρακάτω συναρτήσεις με την τελευταία να καλείται από την ενότητα run.py.

```
def is_measure_word(word)
def is_cooking_tool(word)
def is_color(word)
def is_ignore_list(word)
def normalize_ingredient_name(ingredient_name)
def is_ingredient(word)
def tokenize_ingredient(text)
```

Η διαδικασία που ακολουθεί είναι αυτή ακριβώς που περιγράφηκε στην ενότητα *4.5*.

```
def tokenize_ingredient(text):
    collected_data=[]

    text=text.replace("and",",")
    text=text.replace("with",",")

    chunks= text.split(",")

    for chunk in chunks:

        chunk=chunk.replace('.',' ')
        chunk=chunk.replace('-',' ')
        chunk=chunk.replace('_',' ')
        tokenized = word_tokenize(chunk)

        ingredients = []
        ing_string= " "
        measure_=[]
        units_=[]

        measure_word = False
        ignore = False
        prev_token=""

        for token in tokenized:
            wordsList = nltk.word_tokenize(token)
            tagged = nltk.pos_tag(wordsList)

            for tag in tagged:

                if tag[1]=='CD':
                    units_.append(normalize_ingredient_name(tag[0]))
                if tag[1] in ['NN','NNS','NNPS','VBN']:
                    base_word=normalize_ingredient_name(tag[0])

                    if is_measure_word(base_word) or base_word in wordlists
↪ .units_of_measure:
                        measure_.append(base_word)

                    else:
                        if  is_ingredient(base_word) and not
↪ is_measure_word(base_word) and not is_ignore_list(base_word)  and
↪ base_word not in  wordlists.units_of_measure:

                            if base_word not in ingredients:
                                if len(prev_token)>0:

↪ ingredients.append(prev_token+" "+base_word.strip())
                                    prev_token=""
                                else:
```

```
                                ingredients.append(base_word.strip())
                        elif base_word in wordlists.food_adjectives:
                            print(base_word)
                            prev_token=base_word

        collected_data.append([units_,measure_,[ing_string.join(ingredients
    ↪ ).strip()]])

    return collected_data
```

Η normalize_ingredient_name πραγματοποιεί Lemmatization με τη χρήση της βιβλιοθήκης WordNet. Οι συναρτήσεις is_measure_word, is_cooking_tool, is_color και is_ignore_list πραγματοποιούν τις απαραίτητες συγκρίσεις για την ταξινόμηση των λέξεων-συμβολοσειρών, με στόχο τον εντοπισμό ποσότητας και λέξεων μέτρησης. Ενδεικτικά οι δύο από αυτές:

```
def is_cooking_tool(word):
    tools1 = ['[Pp]ot','[Kk]nife','[Pp]an.?','[Kk]nives','[Gg]rater','[Bb]
    ↪ oard','[Oo]pener','[Cc]up.?','[Ss]poon.?','[Bb]owl.?','[Cc]olander.?
    ↪ ','[Pp]eeler.?','[Mm]asher.?','[Ww]hisk.?','[Ss]pinner.?','[Gg]rater
    ↪ .?','[Ss]hear.?','[Jj]uicer','[Pp]ress','[Ss]teel','[Ss]harpener.?']
    tools2 = ['[Ff]oil','skillet','plate.?','[Ss]patula.?','[Ss]poon.?','[
    ↪ Tt]ong.?','[Ll]adle.?','[Mm]itt.?','[Oo]ven','[Tt]rivet.?','[Gg]uard
    ↪ ','[Tt]hermometer.?','[Bb]lender.?','[Ss]cale.?','[Cc]ontainer.?']

    for tool in tools1:
        if re.search(tool,word):
            return True

    for tool in tools2:
        if re.search(tool,word):
            return True

    return False

def is_color(word):
    colors = dict(mcolors.BASE_COLORS, **mcolors.CSS4_COLORS)

    if word in list(colors.keys()) or word[0:-1] in list(colors.keys()):

        return True
    else:
        return False
```

Τέλος, η χρήση των synsets για την ταξινόμηση των λέξεων-συμβολοσειρών σε τροφές και μη, γίνεται με τη συνάρτηση is_ingredient. Ο κώδικας της δίνεται παρακάτω:

```
def is_ingredient(word):
    skip_list=['cut','non','eat','drink']

    if word in skip_list:
        return True
    reject_synsets = ['meal.n.01', 'meal.n.02', 'dish.n.02', 'vitamin.n.01'
    ↪ ]
    reject_synsets = set(wordnet.synset(w) for w in reject_synsets)
    accept_synsets = ['chemical.n.01','creating_from_raw_materials.n.01','
    ↪ natural_object.n.01',  'living_thing.n.01','mixture.n.01','food.n.01
    ↪ ', 'food.n.02','fruit.n.01','fruit.n.02',  'plant.n.01','plant.n.02'
    ↪ ,'starch.n.01','edible_fat.n.01']
    accept_synsets = set(wordnet.synset(w) for w in accept_synsets)

    for word_synset in wordnet.synsets(word, wordnet.NOUN):
        all_synsets = set(word_synset.closure(lambda s: s.hypernyms()))
```

```
        all_synsets.add(word_synset)
        for synset in accept_synsets:
            if synset in all_synsets:
                return True

        for synset in reject_synsets:
            if synset in all_synsets:
                return False

    if len(wordnet.synsets(word, wordnet.NOUN))==0:
        return True

    return False
```

Εδώ εισάγονται και βιβλιοθήκες όπως η re και η fractions, καθώς και μέρη της πλατφόρμας βιβλιοθηκών NLTK όπως η word_tokenize, η stopwords και η WordNet. Η re παρέχει τη δυνατότητα πράξεων μεταξύ κανονικών εκφράσεων και η fractions μεταξύ ρητών αριθμών, ενώ οι υπόλοιπες αναλύθηκαν σε προηγούμενο κεφάλαιο. Επίσης, κάποιες από τις λίστες που χρησιμοποιούνται βρίσκονται σε μορφή .txt στην ενότητα Wordlists στα αρχεία food.txt, food_adjectives.txt κ.λπ.

5.2.3 Flask Framework

Το Flask framework τρέχει από την κύρια ενότητα **run.py** ενώ χρησιμοποιήθηκε η έκδοση 2.0.1. Εδώ δημιουργείται το endpoint. Το ορίζουμε να καλείται από τον σύνδεσμο '/api/parse_food' με μέθοδο 'POST' και τύπο json. Εδώ επίσης καλείται η συνάρτηση tokenize_ingredient και επιστρέφονται τα αποτελέσματα με μορφή JSON.

```
from flask import Flask, request, jsonify

from Ingredients import tokenize_ingredient

app = Flask(__name__)

# root
@app.route("/")
def index():
    """
    this is a root dir of my server
    :return: str
    """
    return "This is root!!!!"


# POST
@app.route('/api/parse_food', methods=['POST'])
def get_text_prediction():
    """
    predicts requested text whether it is ham or spam
    :return: json
    """
    json = request.get_json()
    print(json)
    if len(json['text']) == 0:
        return jsonify({'error': 'invalid input'})

    results=tokenize_ingredient(json['text'])

    return jsonify({'parsed':results})

# running web app in local machine
```

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080)
```

5.2.4 Docker

Για τις docker υπηρεσίες που απαιτεί η εφαρμογή και το deployment τους στο Google Cloud Run χρησιμοποιήθηκε το ακόλουθο **dockerfile**:

```
# Use the official lightweight Python image https://hub.docker.com/_/python
FROM python:3.8.6

# Allow statements and log messages to immediately appear in the Knative
    ↪ logs
ENV PYTHONUNBUFFERED True

# Copy local code to the container image.
ENV APP_HOME /app
WORKDIR $APP_HOME
COPY . ./

# Install production dependencies.
RUN pip install -r requirements.txt
RUN python -m nltk.downloader punkt -d /usr/local/nltk_data
RUN python -m nltk.downloader averaged_perceptron_tagger -d /usr/local/
    ↪ nltk_data
RUN python -m nltk.downloader wordnet -d /usr/local/nltk_data
RUN pip install gunicorn


# Run the web service on container startup. Here we use the gunicorn
# webserver, with one worker process and 8 threads.
# For environments with multiple CPU cores, increase the number of workers
# to be equal to the cores available.
# Timeout is set to 0 to disable the timeouts of the workers to allow Cloud
    ↪  Run to handle instance scaling.
CMD exec gunicorn --bind :$PORT --workers 1 --threads 8 --timeout 0 run:app
```

Πρώτα καθορίζουμε τη έκδοση της Python που θέλουμε να τρέχει το περιβάλλον όπου θα φιλοξενείται το docker image. Έπειτα, αντιγράφεται ο κώδικας της εφαρμογής στο container image και εγκαθίστανται τα υπόλοιπα πακέτα που απαιτούνται, όπως το NLTK και το Flask καθώς και το gunicorn. Το **gunicorn** είναι ένας Python WSGI HTTP Server για UNIX. Είναι ιδιαίτερα σημαντικός επειδή είναι σταθερός, ευρέως συμβατός με διάφορα web frameworks, απλά υλοποιημένος, ελαφρύς στη χρήση πόρων του διακομιστή και αρκετά γρήγορος. Με την τελευταία γραμμή, εκκινούμε τον gunicorn server κάνοντας τις απαραίτητες ρυθμίσεις.

Το αρχείο **requirements.txt** περιλαμβάνει τις παρακάτω γραμμές-πακέτα:

```
Flask~=2.0.1
nltk~=3.6.2
numpy
```

Επίσης,για το localstack του Docker Desktop, χρησιμοποιούμε το **.dockerignore** που καθορίζει τα αρχεία που πρέπει να αγνοήσει.

```
Dockerfile
*.pyc
__pycache__
```

5.2.5 Ρύθμιση Cloud Run

Για το deployment του Web App σαν service στο Cloud Run, αφού δημιουργήθηκε το project στο Google Cloud χρησιμοποιήθηκαν οι παρακάτω γραμμές στο terminal του IntelliJ IDEA. Η πρώτη δημιουργεί το containter image με τη χρήση του εργαλείου Cloud Build της Google και με βάση το Dockerfile. Η δεύτερη κάνει το deploy του κατασκευασμένου image. Οι οδηγίες αυτές βρίσκονται στο αρχείο **README.md**.

```
gcloud builds submit --tag gcr.io/storied-scarab-316618/food-model-service
    ↪  --project=storied-scarab-316618
gcloud builds submit --tag gcr.io/Project Id/Service Name  --project=
    ↪ Project Id

```

```

gcloud run deploy --image gcr.io/storied-scarab-316618/food-model-service
    ↪ --platform managed --project=storied-scarab-316618
gcloud run deploy --image gcr.io/Project Id/Service Name --platform managed
    ↪   --project=Project Id
```

Κεφάλαιο 6

Αποτίμηση και Έλεγχος

Στο κεφάλαιο αυτό, θα γίνει έλεγχος της καλής λειτουργίας της εφαρμογής και θα απεικονιστούν με στιγμιότυπα οθόνης, οι δύο λειτουργίες της, με δύο βασικά σενάρια-παραδείγματα χρήσης. Η σειρά οθονών και οι ενέργειες του χρήστη παρουσιάζονται όπως περιγράφονται στην ενότητα *4.3*.

6.1 Παραδείγματα Χρήσης

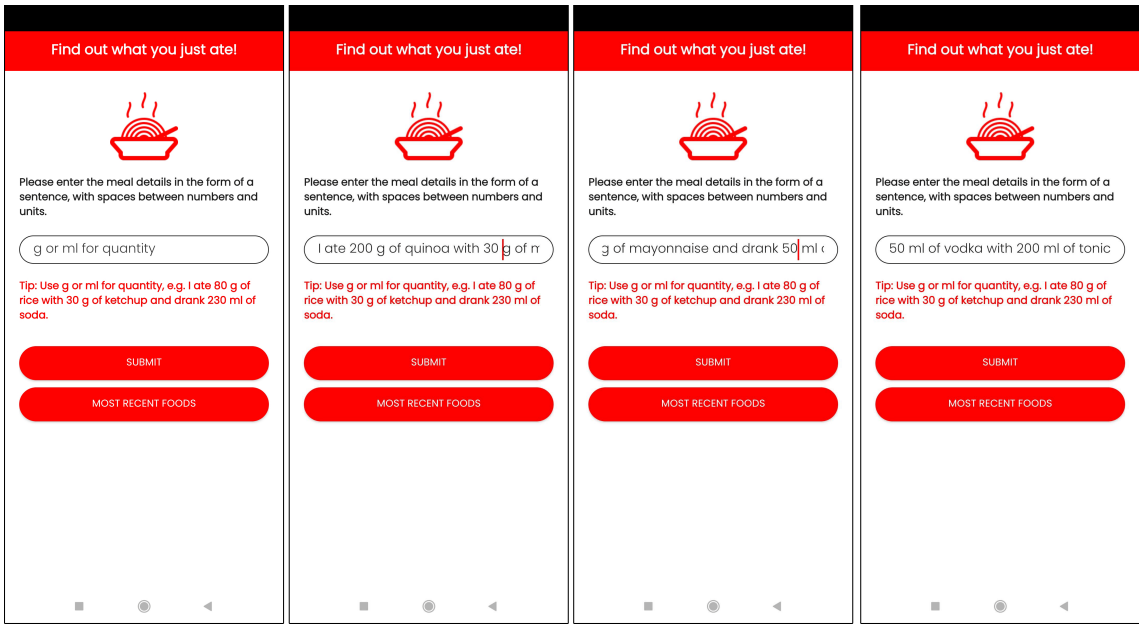
6.1.1 Αναζήτηση διατροφικών στοιχείων νέου γεύματος

Συνοπτικά στο παράδειγμα αυτό:

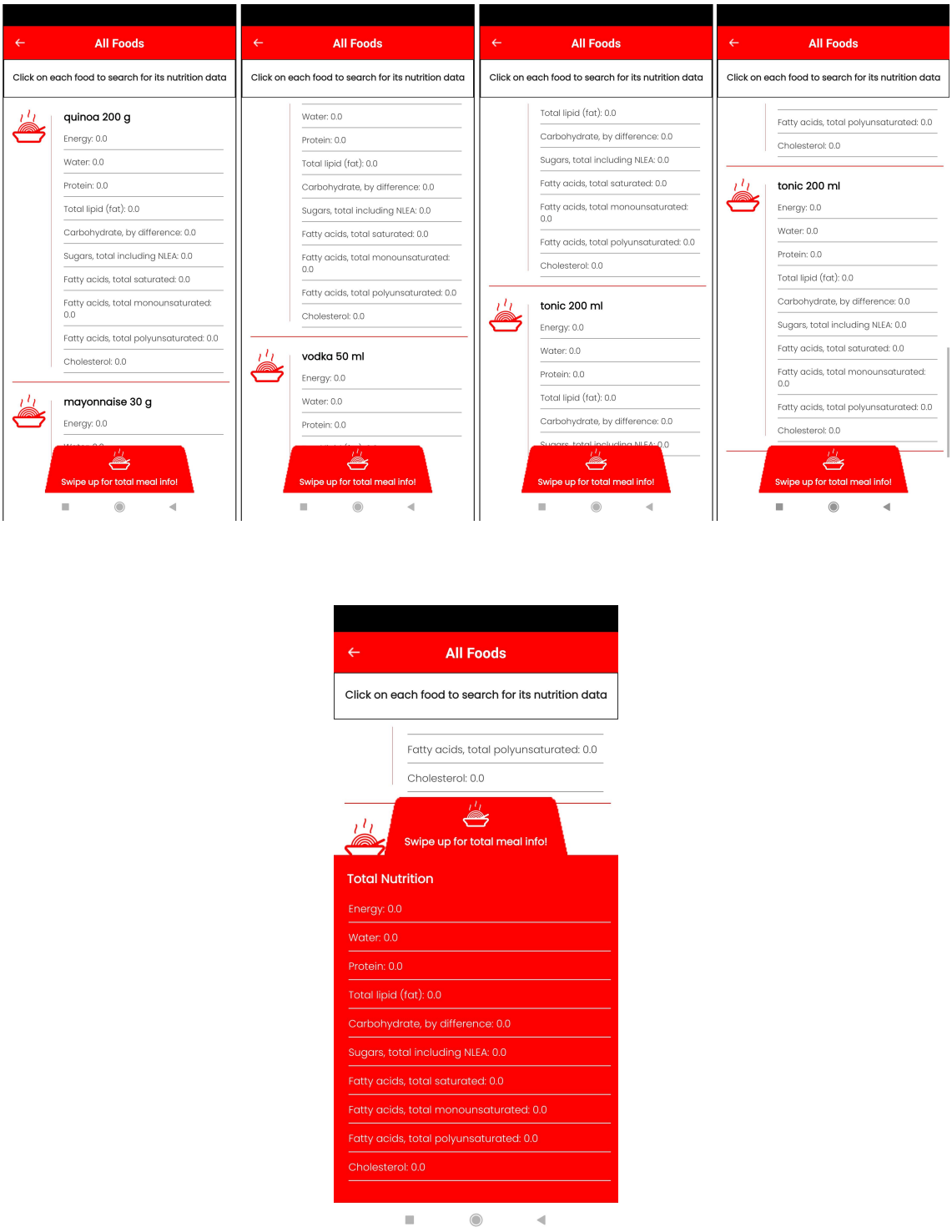
1. ο χρήστης πληκτρολογεί σε μορφή πρότασης φυσικής γλώσσας το γεύμα που είχε
2. του παρουσιάζονται όλες οι τροφές που αυτό περιείχε, όπου μπορεί να διαλέξει να κάνει αίτημα για κάποια από αυτές
3. διαλέγει και του παρουσιάζονται οι καταχωρήσεις που υπάρχουν στην online βάση δεδομένων για τη συγκεκριμένη τροφή
4. επιλέγει ποια καταχώρηση του ταιριάζει, προσθέτοντάς τα διατροφικά στοιχεία της στο σύνολο του γεύματος
5. επιστρέφει στην οθόνη με όλες τις τροφές και τα συνολικά διατροφικά στοιχεία όπου επαναλαμβάνει τα τελευταία 2 βήματα και για τις υπόλοιπες τροφές

Εκτέλεση σεναρίου

Εικόνα 6.1: *Εισαγωγική Οθόνη με το logo*

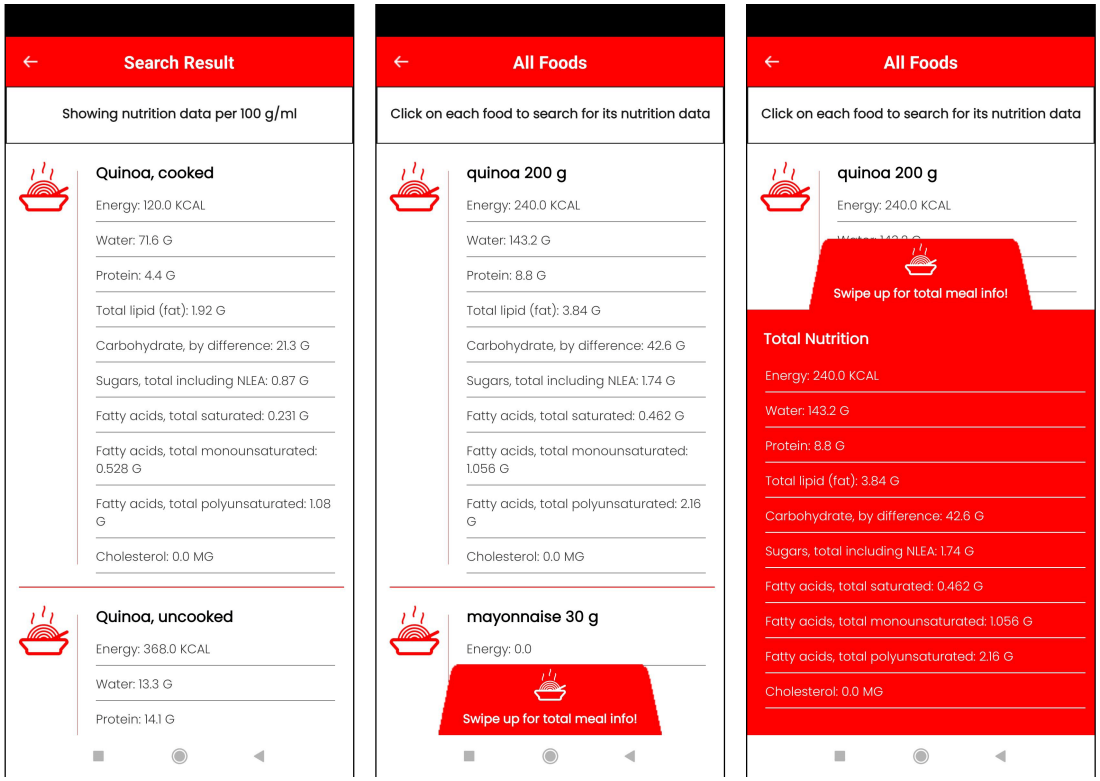


Εικόνα 6.2: *Αρχική οθόνη όπου εισάγεται η είσοδος "I ate 200 g of quinoa with 30 g of mayonnaise and drank 50 ml of vodka with 200 ml of tonic"*



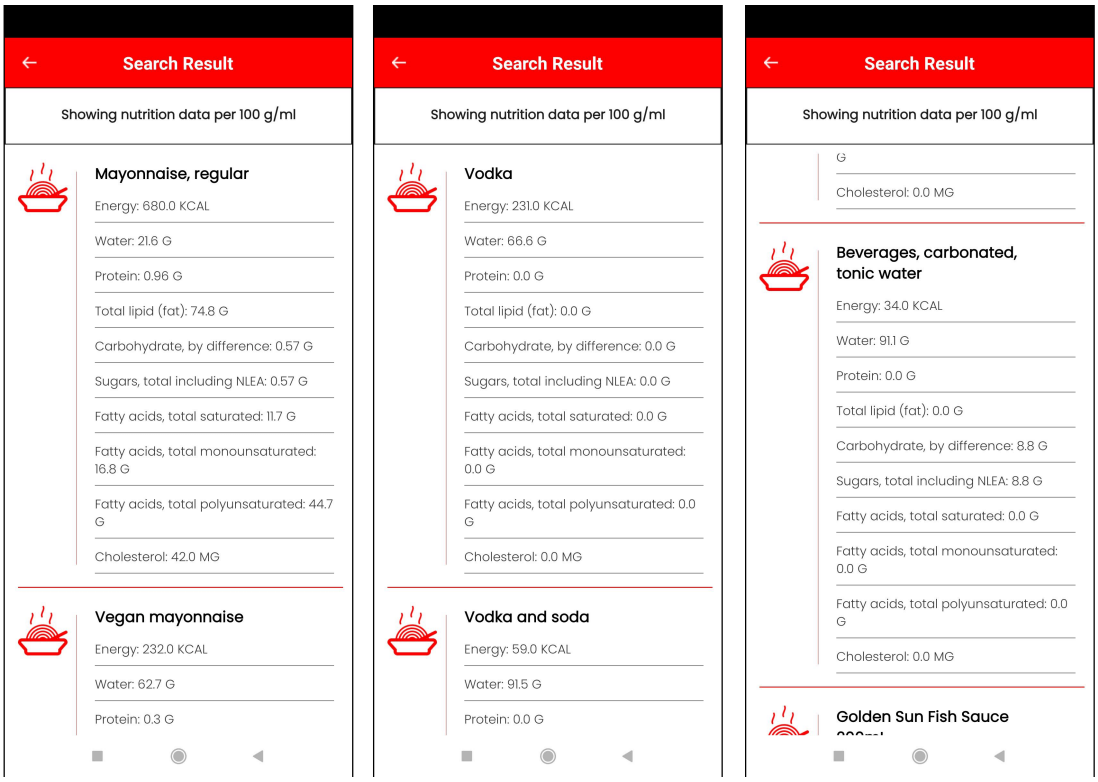
Εικόνα 6.3: *Οθόνη εμφάνισης λίστας με τις τροφές εισόδου καθώς και τα διατροφικά τους στοιχεία (τα οποία είναι μηδενικά γιατί δεν έχει επιλεχθεί καμία καταχώρηση από τη βάση δεδομένων)*

Διαλέγοντας κάποια από τις τροφές, πραγματοποιεί αίτημα στη βάση δεδομένων για αυτήν. Στην επόμενη οθόνη του εμφανίζονται τα αποτελέσματα από τη FoodData Central με τα διατροφικά τους στοιχεία ανά 100 g/ml, κι αφού ο χρήστης επιλέξει την καταχώρηση που του ταιριάζει, επιστρέφει στην προηγούμενη οθόνη, εμφανίζοντας όμως τώρα τα διατροφικά στοιχεία της τροφής αυτής για την ποσότητα που λάβαμε στην είσοδο και έχοντας προσθέσει αυτά στα συνολικά στοιχεία του γεύματος.

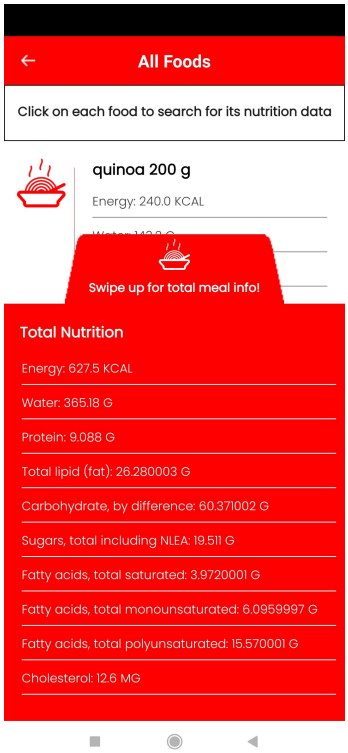


Εικόνα 6.4: *Οθόνη αποτελεσμάτων στη βάση δεδομένων για "quinoa" κι επιστροφή στην προηγούμενη οθόνη*

Στη συνέχεια φαίνονται οι καταχωρήσεις από τη FoodData Central και για τις υπόλοιπες τροφές, και συγκεκριμένα αυτές που επιλέχθηκαν.

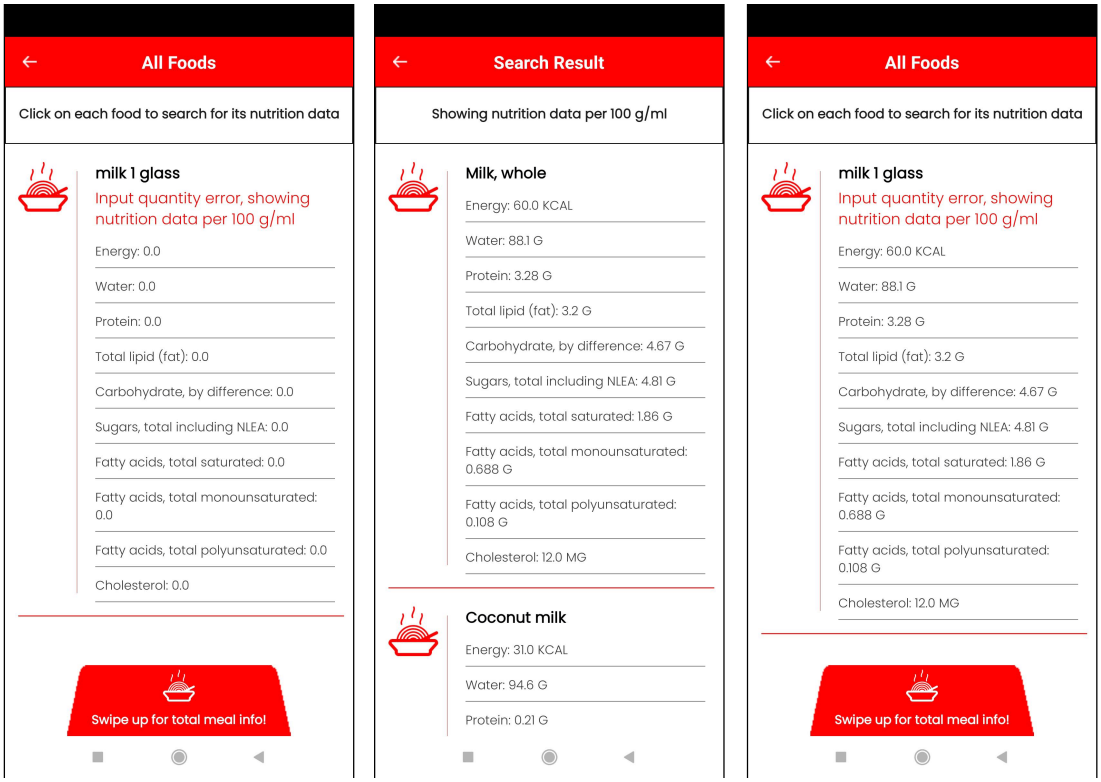


Εικόνα 6.5: *Καταχωρήσεις από τη FoodData Central για "mayonnaise","vodka","tonic"*



Εικόνα 6.6: *Συνολικά διατροφικά στοιχεία γεύματος με βάση τις καταχωρήσεις που επιλέξαμε*

Στην περίπτωση που ο χρήστης εισάγει ποσότητα αλλά σε μη έγκυρη μορφή π.χ. "1 glass of wine", η εφαρμογή δείχνει την τροφή αλλά παρουσιάζει τα διατροφικά της στοιχεία ανά 100 g/ml.



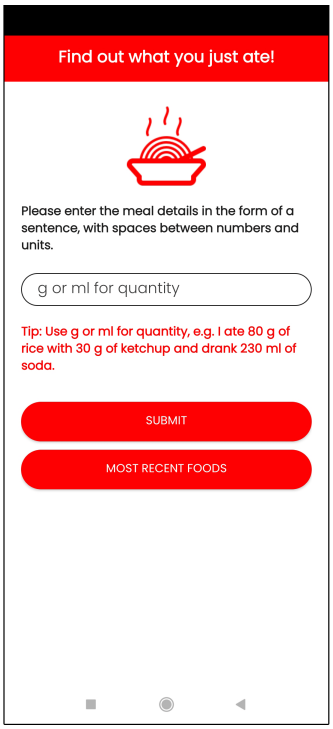
Εικόνα 6.7: *Παρουσίαση διατροφικών στοιχείων ανά 100 g/ml λόγω μη μετρήσιμης ποσότητας*

6.1.2 Προβολή διατροφικών στοιχείων προσφάτως αναζητημένης τροφής

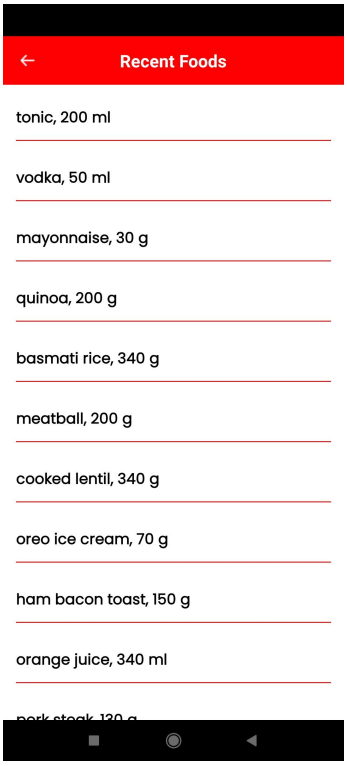
Συνοπτικά στο παράδειγμα αυτό:

1. ο χρήστης πατάει το «κουμπί» που τον οδηγεί στην οθόνη με τις αποθηκευμένες τροφές στην τοπική βάση δεδομένων
2. εκεί βλέπει τις καταχωρήσεις που έχει διαλέξει πρόσφατα για τροφές από τα πρόσφατα γεύματά του, όπου μπορεί να διαλέξει για ποια επιθυμεί να δει τα διατροφικά της στοιχεία
3. του παρουσιάζονται τα διατροφικά στοιχεία για την τροφή που επέλεξε

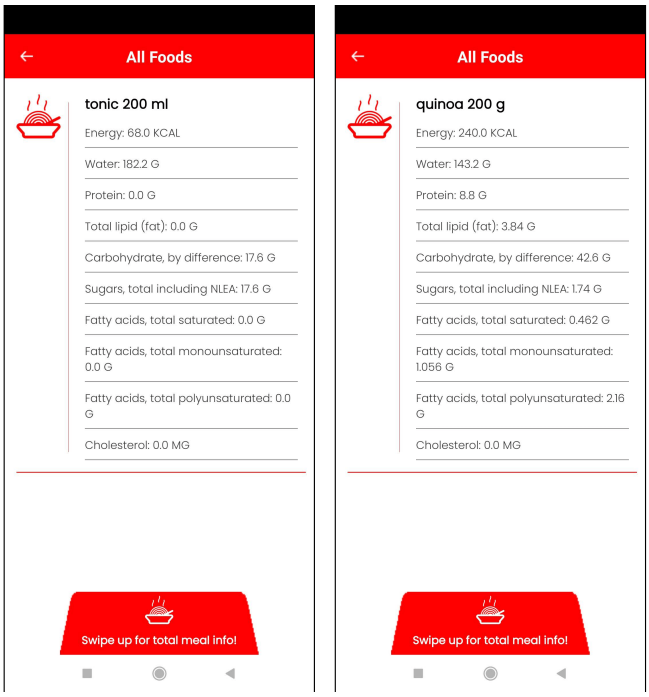
Εκτέλεση σεναρίου



Εικόνα 6.8: *Από την αρχική οθόνη, πατάει το κουμπί "Most Recent Foods"*



Εικόνα 6.9: *Οθόνη με όλες τις πρόσφατες καταχωρήσεις*



Εικόνα 6.10: *Καταχωρήσεις "tonic 200ml" και "quinoa 200g"*

Κεφάλαιο 7

Επίλογος

7.1 Συμπεράσματα

Η όλη διαδικασία της διπλωματικής εργασίας και της δημιουργίας της εφαρμογής αποτέλεσε για εμένα μια εμπειρία ανακάλυψης και ενασχόλησης με νέες τεχνολογίες, ανάλυσης και σύνθεσης ενός ενιαίου συστήματος καθώς και εξοικείωσης με τη διαδικασία που μπορεί να ακολουθήσει κάποιος που έχει στόχο τη δημιουργία μιας επιτυχημένης κι εύχρηστης εφαρμογής.

Μέσα από την αναζήτηση καινούριων εργαλείων και τεχνολογιών, συνειδητοποίησα ουσιαστικά το άπειρο των δυνατοτήτων που έχει κάποιος στη διάθεση του για τη δημιουργία συστημάτων και την αντιμετώπιση προβλημάτων. Ταυτόχρονα, είναι πολλές και οι δυσκολίες και τα εμπόδια που συναντάει κάποιος στην κατασκευή ενός ολοκληρωμένου συστήματος, τόσο κατά τη σχεδίαση όσο και κατά την ανάπτυξη του.

Ένα πλήρες σύστημα-εφαρμογή έχει συνήθως ενδελεχής ανάλυση των απαιτήσεων και των διαθέσιμων μέσων για την πλήρωση αυτών, σχεδίαση και σύνθεση αυτών για την επίτευξη του στόχου, καθώς και ανάπτυξη. Πλέον, ο μέσος χρήστης κινητών συσκευών έχει πολλή μεγαλύτερη ευχέρεια, άνεση και κατανόηση για τη λειτουργία και τη χρήση της συσκευής. Παράλληλα όμως, οι απαιτήσεις παραμένουν υψηλές καθώς αναζητά την πιο εύχρηστη, ομαλή και φυσική λειτουργία σε μια εφαρμογή, πράγμα που καθορίζει εν τέλει και το κατά πόσο θα συνεχίσει να τη χρησιμοποιεί. Μία εφαρμογή που απλά ικανοποιεί μια ανάγκη του χρήστη μπορεί να γνωρίσει από τον ίδιο μερική ή και καθόλου αποδοχή. Είναι αναγκαίο να εστιάζουμε στην εμπειρία που έχει ο χρήστης και πόσο φιλική και εύχρηστη είναι η διεπαφή, εφόσον σκοπός είναι να τον ενθαρρύνει και να τον εκπαιδεύει στη χρήση της. Γι' αυτό πρέπει να δίνεται βάση στις λεπτομέρειες και τον τρόπο με τον οποίο πετυχαίνει το στόχο της.

Όπως γνωρίζουμε, οι πληροφορίες στις οποίες παρέχει πρόσβαση η εφαρμογή, είναι ήδη διαθέσιμες στο διαδίκτυο και μπορεί οποιοσδήποτε να τις κοιτάξει με τη χρήση ενός περιηγητή. Η συνεισφορά της παρούσας εφαρμογής βρίσκεται στην κατασκευή ενός συστήματος-εφαρμογής για την παροχή τους στον τελικό χρηστή με τρόπο απλό και κατανοητό, ακόμα και για χρήστες χωρίς εμπειρία. Αντί ένας χρήστης να πρέπει να χρησιμοποιήσει κάποιο σύνδεσμο όπου θα πληκτρολογεί και θα ψάχνει τα στοιχεία κάθε τροφής ξεχωριστά, μπορεί με την εφαρμογή να γράψει εύκολα σε αγγλική γλώσσα το σύνολο που κατανάλωσε και η εφαρμογή να του δώσει τη δυνατότητα με το πάτημα ενός κουμπιού να ψάξει για κάθε τροφή και να δει τα συνολικά διατροφικά στοιχεία του γεύματος σε μία οθόνη. Αντιλήφθηκα ότι, δεδομένου πως

η εφαρμογή στοχεύει να διευκολύνει το χρήστη καθημερινά, παίζουν μεγάλο ρόλο και η σταθερότητα και η αξιοπιστία των υπηρεσιών της εφαρμογής, προκειμένου ο χρήστης να μπορεί να την εμπιστευτεί για την κάλυψη των αναγκών του. Τέλος, έβγαλα και το συμπέρασμα ότι σαν μια εφαρμογή που απευθύνεται σε ένα ευρύ κοινό, προκειμένου να μπορεί να ικανοποιεί τις ανάγκες των χρηστών της συνεχώς, πρέπει να μπορεί να εξελίσσεται και συνεπώς να είναι εύκολα επεκτάσιμη από οποιονδήποτε developer.

Συμπερασματικά, η εφαρμογή που σχεδιάστηκε στα πλαίσια αυτής της διπλωματικής αποτελεί ένα σύστημα που πληρεί τις παραπάνω προϋποθέσεις και μπορεί να καλύψει τις γνωστικές ανάγκες του χρήστη σχετικά με τα διατροφικά στοιχεία των γευμάτων του.

7.2 Μελλοντικές Επεκτάσεις

Το σύστημα που αναπτύχθηκε στα πλαίσια αυτής της διπλωματικής εργασίας θα μπορούσε να βελτιωθεί και να επεκταθεί περαιτέρω. Παρά το γεγονός ότι η εφαρμογή σχεδιάστηκε ως standalone σύστημα, θα μπορούσε μελλοντικά να εξελιχθεί ώστε να καλύπτει καλύτερα και περισσότερες ανάγκες που έχει ένας χρήστης που τον ενδιαφέρει η παρακολούθηση και ρύθμιση της διατροφής του. Συγκεκριμένα, αναφέρονται τα ακόλουθα:

- Ένα σημαντικό μέρος της εφαρμογής είναι το πρόγραμμα Python που εξάγει πληροφορίες σχετικά με τις τροφές από την είσοδο. Στην περίπτωση που δοκιμαστεί σε μεγαλύτερη κλίμακα, θα μπορούσε να βελτιωθεί διορθώνοντας περιπτώσεις που δε λειτουργεί όπως πρέπει, οι οποίες δεν έχουν εντοπισθεί ακόμα. Ταυτόχρονα, θα μπορούσε να υλοποιηθεί κάποιο μοντέλο αναγνώρισης προφορικού λόγου ώστε ο χρήστης να εισάγει δεδομένα κατά αυτόν τον τρόπο, πετυχαίνοντας την περαιτέρω διευκόλυνσή του. Τέλος, θα ήταν δυνατή η εγκατάσταση και υλοποίηση μοντέλου για αναγνώριση τροφών μέσω κάμερας, όμως ένα εμπόδιο που συναντάται σε αυτήν την κατεύθυνση είναι η αναγνώριση της ακριβής ποσότητας μέσω του όγκου.

- Η ευχρήστοτητα της εφαρμογής αυτής έγκειται και στην οργάνωση των γευμάτων. Αυτή τη στιγμή αποθηκεύονται οι πρόσφατες καταχωρήσεις τροφών ώστε να μπορεί ο χρήστης να έχει πρόσβαση σε αυτές εύκολα. Θα μπορούσε αυτό να βελτιωθεί οργανώνοντας τις πρόσφατες καταχωρήσεις με κάποιο συγκεκριμένο τρόπο (π.χ. πρωινό, μεσημεριανό, βραδινό, δεκατιανό), αποθηκεύοντας ολόκληρα τα γεύματα που πραγματοποίησε ή αποθηκεύοντας σε κάθε τροφή όλες τις διαφορετικές καταχωρήσεις που έχει διαλέξει από την online βάση δεδομένων. Κάτι τέτοιο θα απαιτούσε επέκταση και προσθήκη μοντέλων και εγγραφών στη τοπική βάση δεδομένων.

- Τέλος, μια δυνατή επέκταση θα ήταν η δυνατότητα προγραμματισμού γευμάτων π.χ. για μια μέρα ολόκληρη ώστε να μπορεί να μοιράσει καλύτερα και να αποφασίσει για τα γεύματα του.

Βιβλιογραφία

[1] *A Guide to Google's recommended Architecture for Android Apps*. https://www.scalablepath.com/blog/recommended-architecture-for-android-apps, 2021. Ημερομηνία πρόσβασης: 10-09-2021.

[2] *World Health Organization: Nutrition*. https://www.who.int/health-topics/nutrition. Ημερομηνία πρόσβασης: 28-06-2021.

[3] *WebMD: Healthy Eating for Weight Loss*. https://www.webmd.com/women/guide/nutrition-101-how-to-eat-healthy. Ημερομηνία πρόσβασης: 29-06-2021.

[4] *World Health Organization: Benefits of a balanced diet*. https://www.euro.who.int/en/health-topics/disease-prevention/nutrition/a-healthy-lifestyle/benefits-of-a-balanced-diet. Ημερομηνία πρόσβασης: 28-06-2021.

[5] John D Fernstrom και RJ Wurtman. *Brain serotonin content: physiological dependence on plasma tryptophan levels*. *Science*, 173(3992):149–152, 1971.

[6] Ted Wilson, Norman J. Temple, George A. Bray και Marie Boyle Struble. *Nutrition Guide for Physicians*. Springer, 2010.

[7] Eleni Vasara, Georgios Marakis, Joao Breda, Petros Skepastianos, Maria Hassapidou, Anthony Kafatos, Nikolaos Rodopaios, Alexandra A Koulouri και Francesco P Cappuccio. *Sodium and potassium intake in healthy adults in Thessaloniki Greater Metropolitan Area—The salt intake in Northern Greece (sing) study*. *Nutrients*, 9(4):417, 2017.

[8] JD Rockefeller. *Diabetes: symptoms, causes, treatment and prevention*. JD Rockefeller, 2015.

[9] Nia S Mitchell, Victoria A Catenacci, Holly R Wyatt και James O Hill. *Obesity: overview of an epidemic*. *Psychiatric clinics*, 34(4):717–732, 2011.

[10] Ted Wilson, Norman J. Temple, George A. Bray και Marie Boyle Struble. *Nutrition Guide for Physicians*. Springer, 2010.

[11] *Pew Research Center: Demographics of Mobile Device Ownership and Adoption in the United States*. https://www.pewresearch.org/internet/fact-sheet/mobile, 2021. Ημερομηνία πρόσβασης: 18-08-2021.

[12] *Pew Research Center: Smartphone ownership in advanced economies higher than in emerging*. https://www.pewresearch.org/global/2019/02/05/smartphone-

ownership-is-growing-rapidly-around-the-world-but-not-always-equally/pg_global-technology-use-2018_2019-02-05_0-01, 2019. Ημερομηνία πρόσβασης: 18-08-2021.

[13] Christian Montag και Sarah Diefenbach. *Towards homo digitalis: important research issues for psychology and the neurosciences at the dawn of the internet of things and the digital society. Sustainability*, 10(2):415, 2018.

[14] James L. Sipes. *Database Dynamics. Landscape Architecture*, 93(3):46–91, 2003.

[15] David Sepkoski. *DATABASES*, σελίδες 392–396. Princeton University Press, 2021.

[16] *The history of Android: The evolution of the biggest mobile OS in the world*. https://www.androidauthority.com/history-android-os-name-789433/, 2021. Ημερομηνία πρόσβασης: 20-08-2021.

[17] *Definition of Android Studio*. https://searchmobilecomputing.techtarget.com/definition/Android-Studio, 2018. Ημερομηνία πρόσβασης: 24-08-2021.

[18] *Everything you need to build on Android*. https://developer.android.com/studio/features, 2021. Ημερομηνία πρόσβασης: 28-08-2021.

[19] *Develop Android apps with Kotlin*. https://developer.android.com/kotlin, 2021. Ημερομηνία πρόσβασης: 27-10-2021.

[20] *Kotlin for Android*. https://kotlinlang.org/docs/android-overview.html, 2021. Ημερομηνία πρόσβασης: 27-10-2021.

[21] *What is IntelliJ IDEA?* https://www.jetbrains.com/idea/features, 2021. Ημερομηνία πρόσβασης: 20-09-2021.

[22] *General Python FAQ*. https://docs.python.org/3/faq/general.html, 2021. Ημερομηνία πρόσβασης: 29-08-2021.

[23] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[24] Steven Bird, Ewan Klein και Edward Loper. *Natural language processing with Python*. OReilly, Beijing, 1η έκδοση, 2009.

[25] *Natural Language Toolkit*. https://www.nltk.org/, 2021. Ημερομηνία πρόσβασης: 30-08-2021.

[26] *NTLK Sentiment Analysis: Text Mining & Analysis in Python*. https://www.datacamp.com/community/tutorials/text-analytics-beginners-nltk, 2019. Ημερομηνία πρόσβασης: 30-08-2021.

[27] *NLTK :: nltk.tokenize package*. https://www.nltk.org/api/nltk.tokenize.html, 2021. Ημερομηνία πρόσβασης: 25-11-2021.

[28] *Python for Data Science and Machine Learning Bootcamp*. https://www.udemy.com/course/python-for-data-science-and-machine-learning-bootcamp/, 2020. Ημερομηνία πρόσβασης: 02-05-2021.

[29] *NLTK :: nltk.tag package*. https://www.nltk.org/api/nltk.tag.html, 2021. Ημερομηνία πρόσβασης: 25-11-2021.

[30] *Penn Treebank P.O.S. Tags*. https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html, 2021. Ημερομηνία πρόσβασης: 25-11-2021.

[31] *Princeton University "About WordNet."*. https://wordnet.princeton.edu, 2010. Ημερομηνία πρόσβασης: 31-08-2021.

[32] Miguel Grinberg. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.", 2018.

[33] *Google Cloud Platform - Cloud Run*. https://cloud.google.com/run, 2021. Ημερομηνία πρόσβασης: 20-08-2021.

[34] *U.S. Department of Agriculture, Agricultural Research Service. FoodData Central*. https://fdc.nal.usda.gov, 2019. Ημερομηνία πρόσβασης: 01-09-2021.

[35] *Androidx.room: Android Developers*. https://developer.android.com/reference/androidx/room/package-summary, 2021. Ημερομηνία πρόσβασης: 12-06-2021.

[36] *Android Developers: Guide to app architecture*. https://developer.android.com/jetpack/guide, 2021. Ημερομηνία πρόσβασης: 10-09-2021.

[37] *Introduction to JSON-Java (org.json)*. https://www.baeldung.com/java-org-json, 2021. Ημερομηνία πρόσβασης: 06-07-2021.

[38] *NLTK :: nltk.tokenize*. https://www.nltk.org/_modules/nltk/tokenize.html, 2021. Ημερομηνία πρόσβασης: 27-11-2021.

[39] *NLTK :: nltk.tokenize.punkt*. https://www.nltk.org/_modules/nltk/tokenize/punkt.html, 2021. Ημερομηνία πρόσβασης: 27-11-2021.

[40] *A Good Part-of-Speech Tagger in about 200 Lines of Python*. https://explosion.ai/blog/part-of-speech-pos-tagger-in-python, 2013. Ημερομηνία πρόσβασης: 25-11-2021.