Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

**Μεταπτυχιακή Διατριβή**

| | |
|---|---|
| Τίτλος Διατριβής | **ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ ANDROID - ΘΕΡΜΙΔΟΜΕΤΡΗΤΗΣ**<br>**DEVELOPMENT OF AN APPLICATION FOR ANDROID - CALORIE COUNTER** |
| Ονοματεπώνυμο Φοιτητή | **ΛΕΩΝΙΔΑΣ ΜΑΡΟΥΛΗΣ** |
| Πατρώνυμο | **ΙΩΑΝΝΗΣ** |
| Αριθμός Μητρώου | **ΜΠΠΛ/ 12032** |
| Επιβλέπων | **ΕΥΘΥΜΙΟΣ ΑΛΕΠΗΣ, ΛΕΚΤΟΡΑΣ** |

Ημερομηνία Παράδοσης **Μήνας Έτος**

**Τριμελής Εξεταστική Επιτροπή**

| | | |
|---|---|---|
| (υπογραφή) | (υπογραφή) | (υπογραφή) |
| Όνομα Επώνυμο<br>Βαθμίδα | Όνομα Επώνυμο<br>Βαθμίδα | Όνομα Επώνυμο<br>Βαθμίδα |

# Πίνακας περιεχομένων

# Πίνακας εικόνων

# 1. Περίληψη

Σκοπός αυτής της εργασίας είναι να περιγράψει τα στάδια δημιουργίας ενός ολοκληρωμένου λογισμικού για την πλατφόρμα Android. Το Android είναι μια πλατφόρμα που αποτελείται από ένα λειτουργικό σύστημα και ένα κιτ ανάπτυξης λογισμικού SDK για φορητές συσκευές. Η εφαρμογή θα πρέπει να επικοινωνεί με τον παγκόσμιο ιστό και συγκεκριμένα με μια απομακρυσμένη βάση δεδομένων η οποία θα δημιουργείται με την χρήση ενός φιλικού προς χρήση διαχειριστικού περιβάλλοντος, του όποιου τα στοιχεία θα γνωρίζει ο ιδιοκτήτης και οι διαχειριστές της εφαρμογής. Έτσι θα υπάρχει μια μόνιμη επικοινωνία και αλληλεπίδραση μεταξύ της κινητής συσκευής και της απομακρυσμένης βάσης. Η εφαρμογή είναι ένας θερμιδομετρητής και σαν αντικείμενο θα έχει να βοηθήσει άτομα που θέλουν να κρατάνε ιστορικό της ημερήσιας κατανάλωσης φαγητών, καθώς και να βοηθήσει εκείνους που θέλουν να αλλάξουν ή να διατηρήσουν το σωματικό τους βάρος να πραγματοποιήσουν τον στόχο τους. Κύριο στοιχείο της εφαρμογής είναι η δημιουργία προφίλ, ο υπολογισμός χρήσιμων δεικτών για την υγεία και η αποθήκευση τόσο των δεδομένων που θα συλλέγονται από την βάση δεδομένων όσο και των επιλογών του χρήστη σε μια τοπική βάση δεδομένων του κινητού (SQLite) έτσι ώστε να είναι δυνατή η χρήση της εφαρμογής και offline.

The purpose of this paper is to describe the steps which is needed to create an complete software for Android platform. Android is a platform consisting of an operating system and a Software Development Kit SDK for handheld devices.. The application should communicate with the Web and more specifically to a remote database which is created by using a user friently management environment whose elements will know the owner and admins of the application. So there will be a permanent communication and interaction between the mobile device and the remote database. The application is a calorie counter and aim to help people who want to keep track of daily consumption of food and to help those who want to change or maintain their body weight to succeed. A key element of the application is the creation of profiles, calculate indicators of the health and save both the data collected from the database and user choice in a local database (SQLite) so that would be possible the use of the application offline.

# 2. Εισαγωγή

Οι τεχνολογικές εξελίξεις στον τομέα της Πληροφορικής έχουν μεταβάλει σημαντικά την ποιότητα της ζωής μας σε όλες τις επιστήμες και ιδιαίτερα στην ιατρική. Συνεχώς δημιουργούνται τεχνολογικές καινοτομίες στον κλάδο της ιατρικής οι οποίες βελτιώνουν τα ποσοστά επιτυχίας των επεμβάσεων καθώς και την στην εξεύρεση νέων μεθόδων έγκαιρης διάγνωσης. Σε αυτές τις καινοτομίες μπορούμε να προσθέσουμε εφαρμογές λογισμικού και πληροφοριακά συστήματα με θέμα στο χώρο της διατροφής και της ιατρικής.

Είναι γεγονός πως ο σύγχρονος τρόπος ζωής με τους γρήγορους ρυθμούς και την καθιστική ζωή, που αρκετά μεγάλο ποσοστό του πληθυσμού έχει υιοθετήσει, κάνει την ανάγκη ενημέρωσης για θέματα υγείας και διατροφής μεγαλύτερη. Με την έλευση των κινητών τεχνολογιών η ενημέρωση γίνεται πιο εύκολα καθώς οι πλειοψηφία του κόσμου σήμερα χρησιμοποίει έξυπνα κινητά.

Σκοπός της παρούσας διπλωματικής ήταν να δημιουργηθεί ένα λογισμικό, φιλικό στην χρήση, το οποίο θα ενημερώσει και θα βοηθήσει όλους του χρήστες της να υπολογίζουν την ημερήσια κατανάλωση θερμίδων. Παρόλο που γίνεται χρήση έγκυρων επιστημονικών δεικτών για να βγουν αποτελέσματα ο αριθμός των θερμίδων είναι κατά προσέγγιση καθώς παίζουν ρόλο αρκετοί παράγοντες σε αυτόν τον υπολογισμό.

Η εφαρμογή θερμιδομετρητής υλοποιήθηκε για την πλατφόρμα Android. Το Android είναι ένα λειτουργικό σύστημα ( OS ) για κινητά που βασίζεται στον πυρήνα (kernel) του λειτουργικού συστήματος Linux και αναπτύσσεται σήμερα από την Google. Το Android θεωρείται σήμερα το πιο δημοφιλές περιβάλλον για κινητά τηλέφωνα καθώς κατέχει το μεγαλύτερο μερίδιο αγοράς. Το μερίδιο αγοράς του Android συνεχώς αυξάνεται όπως φανερώνουν στοιχεία της Kantar Worldpanel Comtech, τα οποία αφορούν τους πρώτους τρεις μήνες του 2014. Συγκεκριμένα στην αμερικανική αγορά, η Google με το Android σκαρφάλωσε 7,3 μονάδες επιτυγχάνοντας μερίδιο 59,1%. Στις κορυφαίες αγορές της Ευρώπης το Android εμφανίζει άνοδο 1,7% ελέγχοντας το 72,4% της συνολικής αγοράς.

Ένα από τα βασικότερα χαρακτηριστικά της εφαρμογής είναι η δημιουργία προφίλ με βάση τα χαρακτηριστικά του χρήστη (όπως ηλικία, σωματικό βάρος, αθλητική δραστηριότητα κλπ). Αφού συλλέξει κάποιες από τις παραπάνω πληροφορίες για τον χρήστη η εφαρμογή θα βγάλει κάποια αποτελέσματα τα όποια θα είναι σημαντικά για την πραγματοποίηση του στόχου του χρήστη. Τα αποτελέσματα αυτά είναι ο δείκτης μάζας σώματος (BMI) αλλά και ο Βασικός Μεταβολικός Ρυθμός (BMR Basal Metabolic Rate). Η εφαρμογή αφού συλλέξει και υπολογίσει αυτές τις πληροφορίες, την πρώτη φορά που θα ανοίξει, θα τις αποθηκεύσει στο κινητό του χρήστη.

Οι πληροφορίες για τα φαγητά (θερμίδες, γραμμάρια πρωτεϊνών κλπ) θα δημιουργούνται από τον διαχειριστή της εφαρμογής και θα φτάνουν στο κινητό μέσω ενός service που το κινητό θα είναι προγραμματισμένο να διαβάζει. Το service αυτό επιστρέφει αποτελέσματα σε μορφή JSON ή XML έτσι ώστε να είναι εύκολη η ανάγνωση των δεδομένων από το κινητό - Client. Για την σύνδεση στο διαχειριστικό web κομμάτι της εφαρμογής χρησιμοποιήθηκε authentication μέθοδος για ασφάλεια έτσι ώστε να μην είναι δυνατή η σύνδεση και η αλλαγή δεδομένων από οποιονδήποτε. Ο διαχειριστής έχει δικαίωμα να δημιουργήσει, διαγράψει και να επεξεργαστεί τις πληροφορίες των φαγητών - κατηγοριών, να δημιουργήσει καινούργιους χρήστες (χρήστες με λιγότερα δικαιώματα από αυτών), να πάρει αντίγραφο ασφαλείας της βάσης καθώς και να αλλάξει τις ρυθμίσεις του συστήματος.

Ένας από τους λόγους που δημιουργήθηκε ένα τόσο πολυσύνθετο περιβάλλον διαχείρισης ήταν για να βοηθήσει αυτόν που θα διαχειρίζεται την εφαρμογή να μπορεί εύκολα να κάνει αλλαγές χωρίς να είναι προγραμματιστής αλλά και χωρίς να απαιτείται να βγάλει καινούργια έκδοση της εφαρμογής στο market. Η εφαρμογή στο κινητό είναι ρυθμισμένη να ενημερώνει την τοπική βάση κάθε μια εβδομάδα αλλά παρέχει και στον χρήστη να την ενημερώσει όποτε αυτός κρίνει αναγκαίο.

Το παραπάνω σύστημα διαχείρισης θα μπορούσε πολύ εύκολα να δημιουργήσει και άλλου τύπου εφαρμογές οι οποίες θα είναι τεχνικά ίδιες αλλά θα αλλάζουν τα δεδομένα και οι πληροφορίες οι οποίες παρέχουν στον χρήστη. Για παράδειγμα θα μπορούσε να δημιουργηθεί

μια εφαρμογή αποθήκης, την οποία θα μπορούν να κατεβάσουν οι πελάτες για να βλέπουν το απόθεμα και τις αγορές από το κινητό τους.

Στο κεφάλαιο 3 θα γίνει μια αναλυτική επισκόπηση των εφαρμογών υγείας και διατροφής που υπάρχουν στο εμπόριο. Πιο συγκεκριμένα γίνεται ανασκόπηση των εφαρμογών καταμέτρησης θερμίδων τόσο στην Ελλάδα όσο και στο εξωτερικό.

Στο κεφάλαιο 4 γίνεται αναλυτική παρουσίαση της εφαρμογής και δίνονται οδηγίες για το πως πρέπει να χρησιμοποιηθεί από τον χρήστη. Αυτό το κεφάλαιο χωρίζεται σε 2 ενότητες, η πρώτη ενότητα αφορά την διαχείριση της απομακρυσμένης σελίδας και το δεύτερο μέρος αφορά την χρήση της εφαρμογής στις κινητές συσκευές.

Το 5 και τελευταίο κεφάλαιο μας αναλύει πως υλοποιήθηκε η εφαρμογή, τι τεχνολογίες χρησιμοποιήθηκαν καθώς και κάποια σημεία που για το πως υπολογίζονται οι δείκτες μέσα στην εφαρμογή. Γίνεται ανάλυση όλων των πινάκων που έχουν χρησιμοποιηθεί τόσο απομακρυσμένα όσο και στο κινητό.

Τέλος στο κεφάλαιο 6 αναλύονται τα τελικά συμπεράσματα και προτείνονται οι μελλοντικές επεκτάσεις που μπορούν να γίνουν.

# 3. Ανασκόπηση πεδίου

Στις μέρες μας υπάρχει ολοένα αυξανόμενο ενδιαφέρον για τη χρήση τεχνολογιών πληροφορικής στον έλεγχο και τη διαχείριση του σωματικού βάρους. Ο σύγχρονος τρόπος ζωής και η έλλειψη χρόνου μας έχει κάνει να τρώμε junk foods χωρίς να υπολογίζουμε τις επιπτώσεις που μπορεί να έχουν στην υγεία μας αλλά και σωματικό μας βάρος. Ως γνωστών οι κινητές τεχνολογίες λόγω της φύσης τους διαδραματίζουν έναν σπουδαίο ρόλο για στην ενημέρωση θεμάτων υγείας. Τα παραπάνω έχουν δημιουργήσει την ανάγκη μιας εύχρηστης εφαρμογής για αυτούς που θέλουν να καταγράφουν το τι τρώνε. Παρακάτω θα παρουσιάσω κάποιες από τις εφαρμογές που έχουν δημιουργηθεί σε Ελλάδα και εξωτερικό.

## 3.1 Εφαρμογές στην Ελλάδα

***1. Θερμιδομετρητής (Android by Pinapps.com)***

Αυτή η εφαρμογή παρέχει ένα κατάλογο με θερμίδες τις οποίες ο χρήστης μπορεί να καταχωρήσει στα αγαπημένα είτε στα φαγητά που κατανάλωσε μέσα στην ημέρα. Τέλος υπάρχει δυνατότητα φιλτραρίσματος της λίστας από τον χρήστη. Η παραπάνω εφαρμογή ανήκει στην κατηγορία "Υγεία και φυσική κατάσταση" και τελευταία ανανέωση έγινε στις 6 Μαΐου 2014.


**Εικόνα 1 - Θερμιδομετρητής**

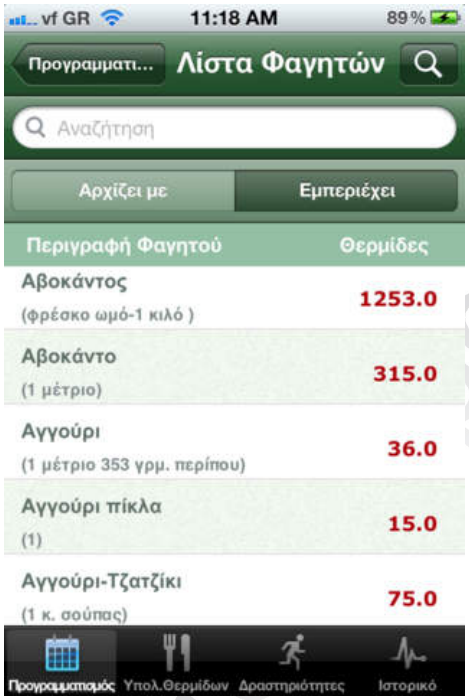
***2. Υπολογιστής Θερμίδων (Android by gdo.com )***

Η εφαρμογή αυτή παρέχει μόνο μια οθόνη με τον κατάλογο των φαγητών υπάρχει η δυνατότητα να υπολογίσεις τις θερμίδες που έφαγες χρησιμοποιώντας το σύμβολο + είτε να τις αφαιρέσεις. Δεν γίνεται αποθήκευση των παραπάνω πληροφοριών και κάθε φορά που ανοίγει η εφαρμογή θα πρέπει να υπολογίζει ο χρήστης από την αρχή. Δεν υπάρχει δυνατότητα αναζήτησης αλλά υπάρχει δυνατότητα φιλτραρίσματος αλφαβητικά. Η εφαρμογή ανήκει στην κατηγορία "Υγεία και φυσική κατάσταση" και τελευταία ενημέρωση έγινε στις 23 Ιουλίου 2013.

**Εικόνα 2- Υπολογιστής Θερμίδων**

**3. Θερμιδομετρητής (iOS by Teokoul)**

Σε αυτή την εφαρμογή υπάρχει δυνατότητα αναζήτησης -φιλτραρίσματος των φαγητών καθώς και υπολογισμού του δείκτη BMI. Η εφαρμογή είναι για κινητά με λειτουργικό iOS και ανήκει στην κατηγορία Reference. Τελευταία ενημέρωση έγινε 19 Ιουλίου 2012.

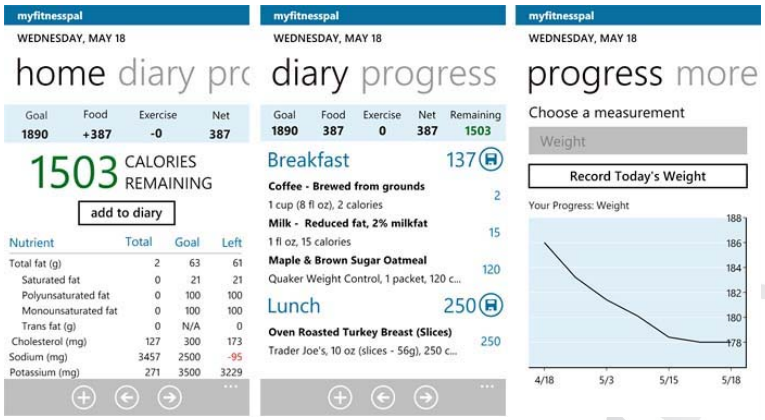



**Εικόνα 3 - Θερμιδομετρητής**

## 3.2 Εφαρμογές στο εξωτερικό

**1. Calorie Counter - MyFitnessPal (android & iOS by MyFitnessPal, Inc.)**

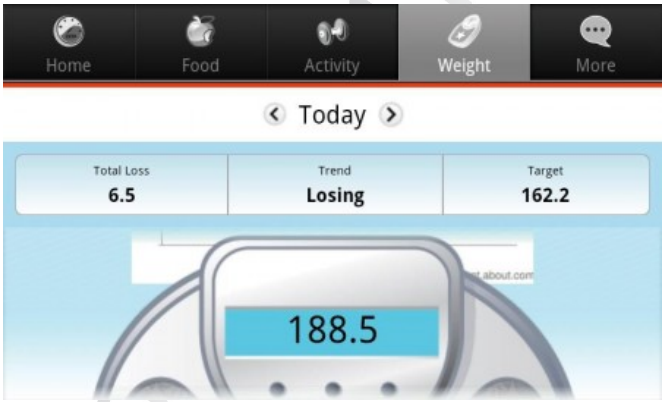
Η πιο δημοφιλής εφαρμογή του χώρου στην κατηγορία fitness και διατροφή με πάνω από 3 εκατομμύρια τροφές στην βάση δεδομένων της. Η εφαρμογή καθοδηγεί τον χρήστη από το πρώτο άνοιγμα βοηθώντας τον να υπολογίσει δείκτες (BMR κλπ) και ημερήσια θερμιδική αξία που πρέπει να λαμβάνει ώστε να πετύχει τον στόχο του. Επιτρέπει στον χρήστη να δημιουργεί τα δικά του φαγητά σε περίπτωση που δεν υπάρχουν όπως επίσης και ημερήσιες δραστηριότητες (όπως περπάτημα, τρέξιμο κλπ) που στο τέλος αφαιρούντε από το σύνολο των θερμίδων που κατανάλωσε. Βασικό στοιχείο της εφαρμογής είναι οι συμβουλές που παρέχει στον χρήστη σε συγκεκριμένα στάδια καθώς και η μέτρηση του στόχου με τον καιρό. Η εφαρμογή ανήκει στην κατηγορία "Υγεία και φυσική κατάσταση" στο Android και στην κατηγορία Health & Fitness στο iOS. Τελευταία ενημέρωση ήταν 24 Σεπτεμβρίου 2014.




**Εικόνα 4 - Calorie Counter - My FitnessPal**

### 2. Calorie Counter (android by CalorieCount.com )

Αυτή η εφαρμογή έχει 250000 φαγητά και χρειάζεται σύνδεση για να χρησιμοποιηθεί (έχει δυνατότητα σύνδεσης και φωνητικά αφού δημιουργηθεί λογαριασμός). Βασικό χαρακτηριστικό της εφαρμογής είναι η σάρωση και αναζήτηση τροφών με barcode και σε περίπτωση που κάποιος κωδικός δεν είναι διαθέσιμος αποθήκευση πληροφοριών του φαγητού από τον χρήστη. Τέλος η εφαρμογή περιέχει διάφορα ενημερωτικά γραφήματα σχετικά με την πρόοδο του χρήστη. Η εφαρμογή είναι διαθέσιμη για Android, BlackBerry και iOS.




**Εικόνα 5 - Calorie Counter**

### 3. Calorie Counter by FatSecret (Android)

Ακόμα μια εφαρμογή για καταγραφή θερμίδων με δυνατότητα εισαγωγής φαγητών από τον χρήστη. Παρέχετε κατηγοριοποίηση των φαγητών με βάση την κατηγορία, αλυσίδων γνωστών εστιατορίων και supermarket καθώς και ξεχωριστά όλα τα δημοφιλή προϊόντα. Η εφαρμογή παροτρύνει τον χρήστη να δημιουργήσει προφίλ στο ξεκίνημα και να καθωρίσει τον στόχο του.


**Εικόνα 6 - Calorie Counter by FatSecret**

# 4. Παρουσίαση εφαρμογής - Θερμιδομετρητής

Παρακάτω θα γίνει μια αναλυτική αναφορά χρήσης τόσο της διαδικτυακής εφαρμογής από τους διαχειριστές του συστήματος όσο και της mobile εφαρμογής από τους χρήστες.

Η διαδικτυακή εφαρμογή υλοποιήθηκε χρησιμοποιώντας το PHP framework Codeigniter το οποίο ακολουθεί το MVC pattern (Model - View - Controller). Το front end κομμάτι της εφαρμογής είναι responsive (για την σωστή προβολή σε όλες τις αναλύσεις) με την βοήθεια του CSS framework bootstrap. Η σελίδα λειτουργεί σωστά σε οποιοσδήποτε σύγχρονο browser.

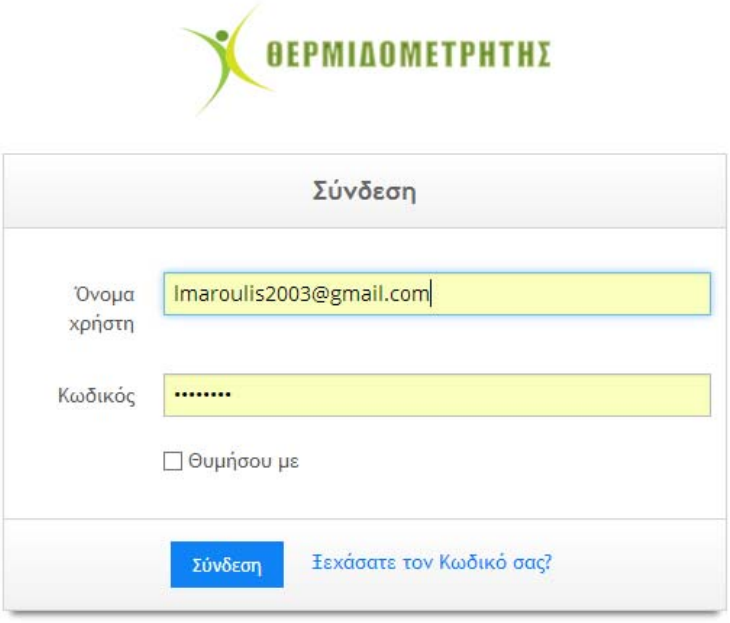
Το mobile κομμάτι της εφαρμογής δημιουργήθηκε με την γλώσσα Java και χρησιμοποιώντας το τελευταίο Android SDK που παρέχει η Google.

## 4.1 Διαχειριστικό θερμιδομετρητή

Παρακάτω θα παρουσιάσουμε όλες τις ενέργειες που μπορεί να κάνει ο διαχειριστής στο διαδικτυακό περιβάλλον.

### 4.1.1 Σύνδεση διαχειριστή

Για την είσοδο στο διαχειριστικό μέρος της εφαρμογής απαιτείται σύνδεση όπως φαίνεται στην παρακάτω εικόνα.

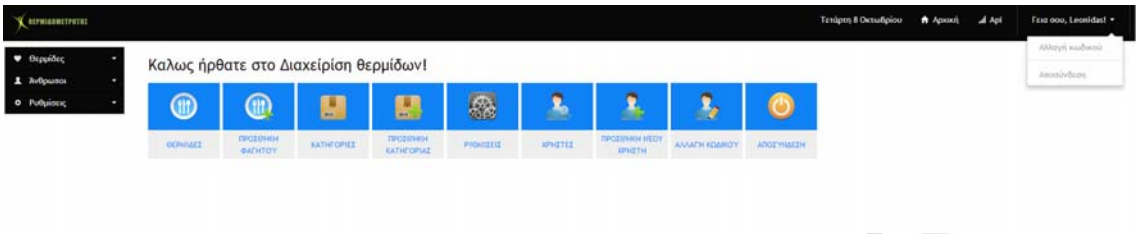



**Εικόνα 7 - Σύνδεση**

Στο όνομα χρήστη πρέπει να εισαχθεί το email του χρήστη ενώ στο πεδίο κωδικός πρέπει να μπει ο κωδικός. Σε περίπτωση που ο κωδικός χαθεί υπάρχει η επιλογή να ξανά σταλεί στο email.

### 4.1.2 Είσοδος στο Dashboard

Αφού εισαχθούν σωστά τα στοιχεία στην φόρμα σύνδεσης η πρώτη οθόνη που θα εμφανιστεί στον διαχειριστή είναι η αρχική.




**Εικόνα 8 - Dashboard**

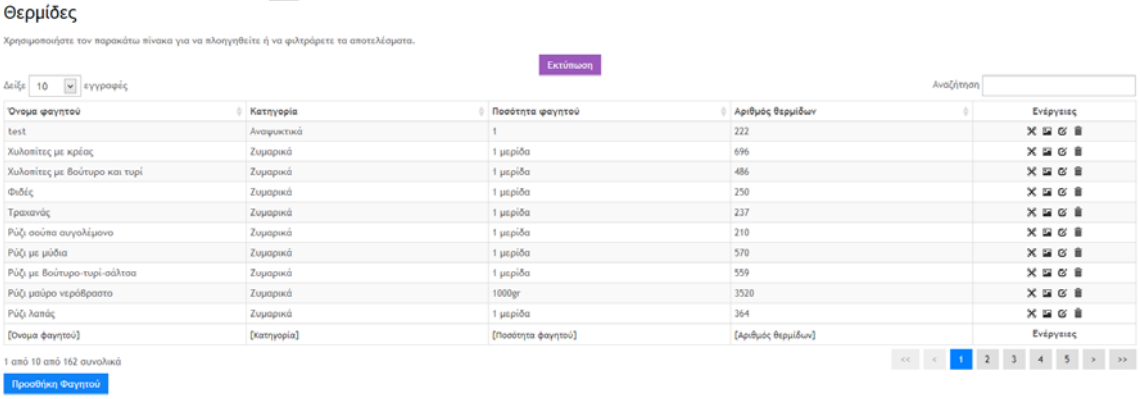
Στο πάνω μέρος δεξιά έχει την τρέχουσα ημερομηνία όπως επίσης και τους συνδέσμους για ανακατεύθυνση στην αρχική σελίδα και στην σελίδα του Api. Τέλος υπάρχει χαιρετισμός προς τον χρήστη που είναι συνδεδεμένος και με το πάτημά του χαιρετισμού βγαίνουν οι επιλογές για αλλαγή κωδικού και αποσύνδεση.

Αριστερά υπάρχει το μενού το οποίο έχει τις ίδιες επιλογές με τα κουτάκια που φαίνονται στο κέντρο της οθόνης. Οι επιλογές είναι οι ακόλουθες:

- ✓ Θερμίδες. Εδώ μπορούμε να διαχειριστούμε όλα τα φαγητά (επεξεργασία, διαγραφή και δημιουργία).
- ✓ Προσθήκη Φαγητού. Φόρμα για εισαγωγή ενός καινούργιου φαγητού στην βάση.
- ✓ Κατηγορίες. Διαχείριση κατηγοριών για εισαγωγή διαγραφή και επεξεργασία.
- ✓ Προσθήκη κατηγορίας. Συντόμευση για την φόρμα εισαγωγής κατηγορίας.
- ✓ Ρυθμίσεις. Εδώ υπάρχουν κάποιες βασικές ρυθμισεις του περιβάλλοντος (θέμα κλπ) οι οποίες είναι διαθέσιμες μόνο στον χρήστη με δικαιώματα ιδιοκτήτη.
- ✓ Χρήστες. Λίστα με όλους του χρήστες (εμφανίζεται μόνο στον ιδιοκτήτη).
- ✓ Προσθήκη νέου χρήστη. Φόρμα για δημιουργία χρήστη (εμφανίζεται μόνο στον ιδιοκτήτη).
- ✓ Αλλαγή κωδικού. Επιλογή για αλλαγή κωδικού χρήστη.
- ✓ Αποσύνδεση από την εφαρμογή.

### 4.1.3 Λίστα φαγητών

Στο συγκεκριμένο section είναι δυνατή η διαχείριση όλων των φαγητών.




**Εικόνα 9 - Λίστα θερμίδων**

Πάνω αριστερά υπάρχει η δυνατότητα να επιλέξει ο χρήστης πόσες εγγραφές φαγητών θέλει να βλέπει (προεπιλογή είναι το 10). Στο κέντρο υπάρχει το κουμπί εκτύπωση το οποίο εκτυπώνει τα αποτελέσματα που βλέπει ο χρήστης (λειτουργεί και σε περίπτωση φιλτραρίσματος). Δεξιά υπάρχει η αναζήτηση η οποία βοηθάει τον χρήστη να ψάξει σε όλα τα

δεδομένα (όνομα φαγητού, κατηγορία κλπ). Η αναζήτηση λειτουργεί και αν εισαχθεί κάτι στο τέλος κάθε στήλης όπου τότε γίνεται φιλτράρισμα μόνο της στήλης.

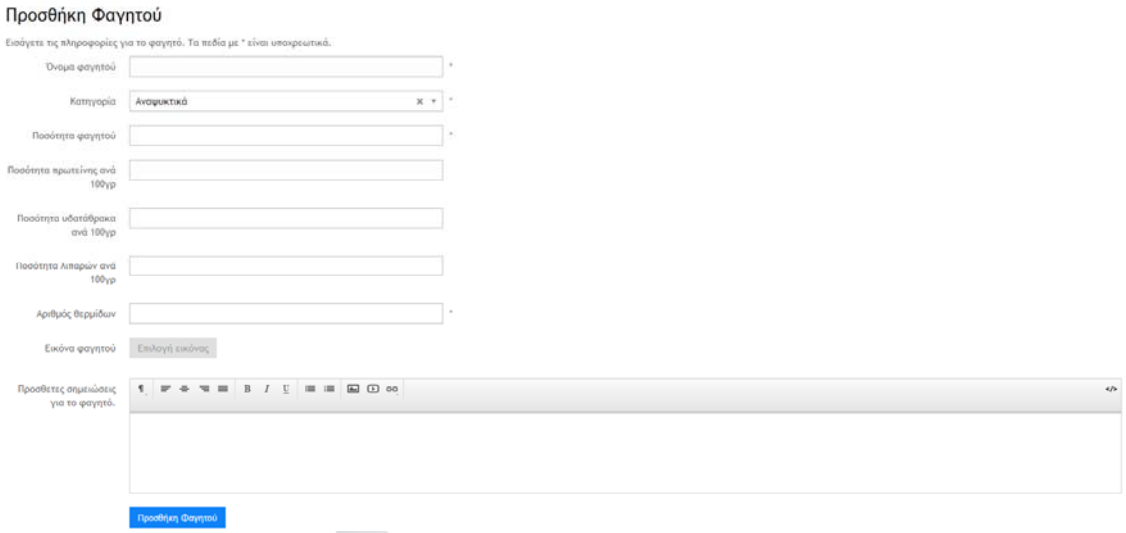
Μετά από κάθε γραμμή υπάρχουν οι ενέργειες που μπορεί να κάνει ο χρήστης από αριστερά προς τα δεξιά είναι οι ακόλουθες:

1. Να δει σε μικρό παράθυρο κάποιες λεπτομέρειες για το φαγητό.
2. Να δει την φωτογραφία του φαγητού αν υπάρχει.
3. Να επεξεργαστεί - ενημερώσει τα δεδομένα του φαγητού.
4. Να διαγράψει το προϊόν.

Κάτω αριστερά υπάρχει το κουμπί προσθήκη φαγητού και δεξιά δυνατότητα να μετακινηθεί στις υπόλοιπες σελίδες.

### 4.1.4 Δημιουργία φαγητού

Η επιλογή προσθήκη φαγητού μας ανοίγει την παρακάτω φόρμα στην οποία μπαίνουν όλα τα δεδομένα του φαγητού.



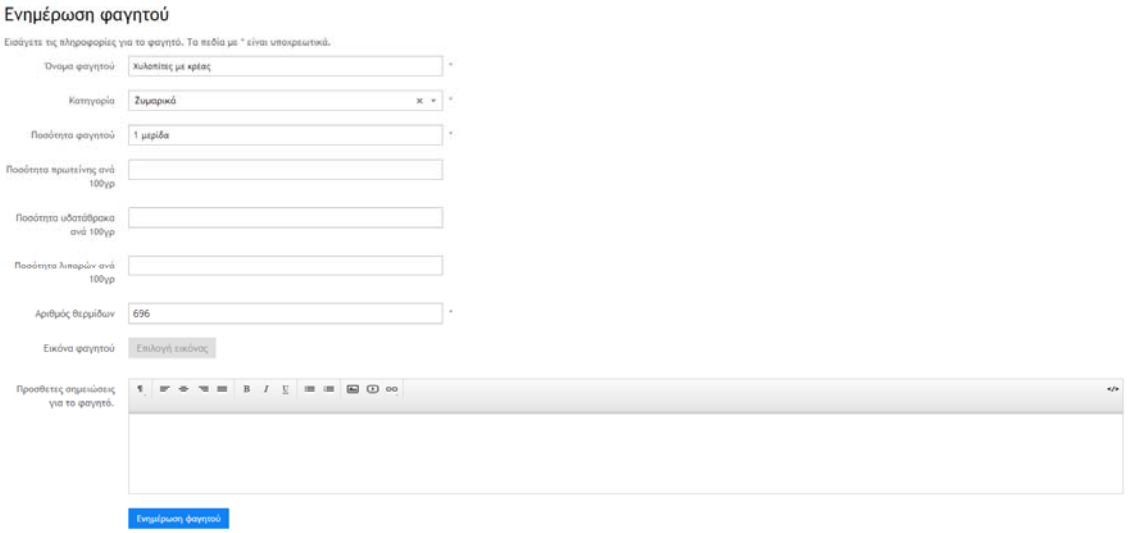

**Εικόνα 10 - Φόρμα δημιουργίας φαγητού**

Τα δεδομένα που μπορεί να εισάγει ο διαχειριστής είναι τα παρακάτω:

- ✓ Όνομα φαγητού (Υποχρεωτικό πεδίο).
- ✓ Κατηγορία που ανήκει το φαγητό (Υποχρεωτικό πεδίο).
- ✓ Ποσότητα φαγητού (Υποχρεωτικό πεδίο).
- ✓ Ποσότητα πρωτείνης ανά 100γρ.
- ✓ Ποσότητα υδατάθρακα ανά 100γρ.
- ✓ Ποσότητα λιπαρών ανά 100γρ.
- ✓ Αριθμός θερμίδων (Υποχρεωτικό πεδίο - επιτρέπονται μόνο αριθμοί).
- ✓ Εικόνα φαγητού.
- ✓ Σημειώσεις σχόλια που αφορούν το συγκεκριμένο φαγητό.

### 4.1.5 Επεξεργασία φαγητού

Η παρακάτω φόρμα μας δείχνει την φόρμα με όλα τα στοιχεία ενός φαγητού τα οποία ο διαχειριστής μπορεί να επεξεργαστεί.

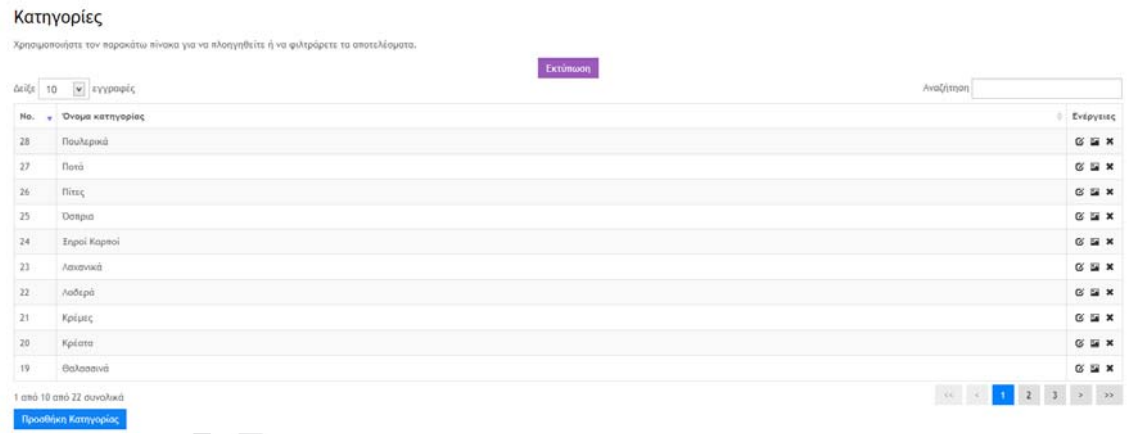



**Εικόνα 11 - Φόρμα επεξεργασίας φαγητού**

Η φόρμα έχει τα ίδια στοιχεία που έχει και η φόρμα προσθήκης με την μόνη διαφορά ότι εδώ φαίνονται τα δεδομένα που υπάρχουν στην βάση. Ο διαχειριστής μπορεί εύκολα να αλλάξει τα στοιχεία η να προσθέσει στοιχεία που είχε αφήσει κενά στο παρελθόν. Όταν τελειώσουν οι αλλαγές θα πρέπει να πατήσει το κουμπί Ενημέρωση φαγητού που βρίσκεται κάτω αριστερά.

### 4.1.6 Λίστα κατηγοριών

Με την ίδια λογική που φαίνονται τα φαγητά ο χρήστης μπορεί να δει και τις κατηγορίες.




**Εικόνα 12 - Λίστα κατηγοριών**

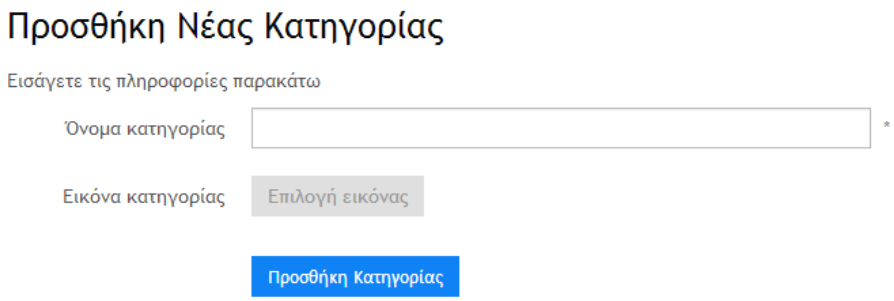
Πάνω αριστερά μπορεί ο χρήστης να επιλέξει πόσες κατηγορίες ανά σελίδα θέλει να βλέπει με προεπιλογή το 10. Υπάρχει δυνατότητα φιλτραρίσματος των κατηγοριών πάνω δεξιά στο πεδίο που λέει αναζήτηση. Στην μέση το κουμπί εκτύπωση έχει ως σκοπό να εκτυπώνει τα δεδομένα του πίνακα (και μετά το φιλτράρισμα).

Κάτω αριστερά το κουμπί Προσθήκη Κατηγορίας μας πηγαίνει στην φόρμα για προσθήκη κατηγορίας και τέλος δεξιά υπάρχουν οι διαθέσιμες σελίδες για περιήγηση σε όλες τις διαθέσιμες κατηγορίες.

Τέλος οι ενέργειες στο πλάι είναι για επεξεργασία, διαγραφή κατηγοριών αλλά υπάρχει και κουμπί για γρήγορη προβολή της εικόνας της κατηγορίας.

### 4.1.7 Εισαγωγή κατηγορίας

Παρακάτω βλέπουμε την φόρμα για εισαγωγή κατηγορίας.

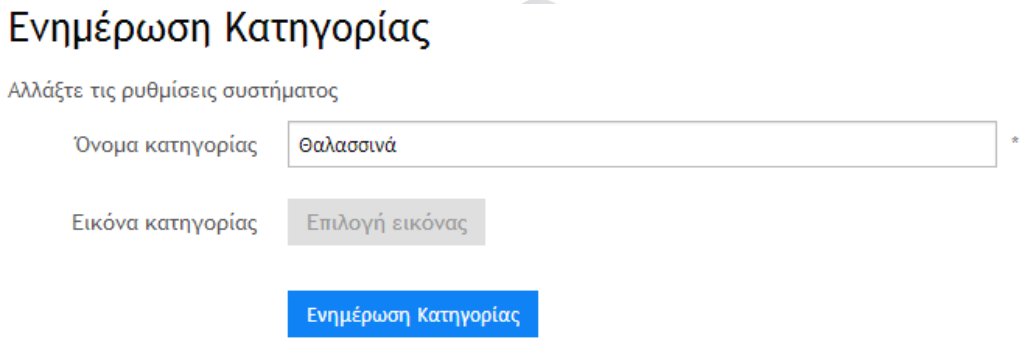



**Εικόνα 13 - Εισαγωγή κατηγορίας**

Σε αυτή την φόρμα θα πρέπει υποχρεωτικά να εισάγουμε τον όνομα της κατηγορίας και προαιρετικά μια εικόνα. Αφού τελειώσουμε πατάμε το κουμπί Προσθήκη Κατηγορίας και η εγγραφή καταχωρείτε.

### 4.1.8 Ενημέρωση κατηγορίας

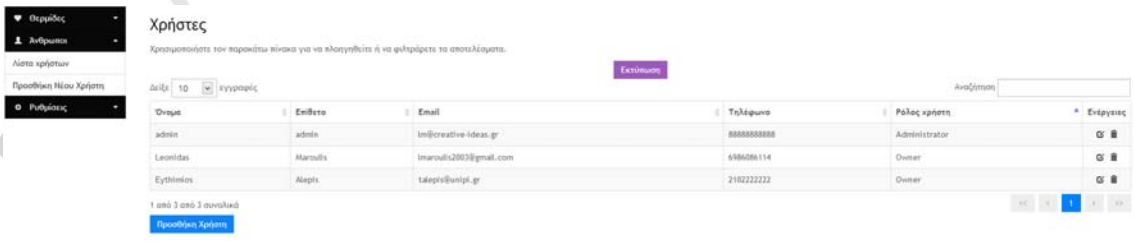
Εδώ μπορούμε να αλλάξουμε το όνομα και την εικόνα από μια κατηγορία που υπάρχει ήδη διαθέσιμη σαν εγγραφή.




**Εικόνα 14 - Ενημέρωση κατηγορίας**

### 4.1.9 Λίστα χρηστών

Επιλέγοντας αριστερά στο μενού Άνθρωποι -> Λίστα χρηστών, εμφανίζονται όλοι οι διαθέσιμοι χρήστες.

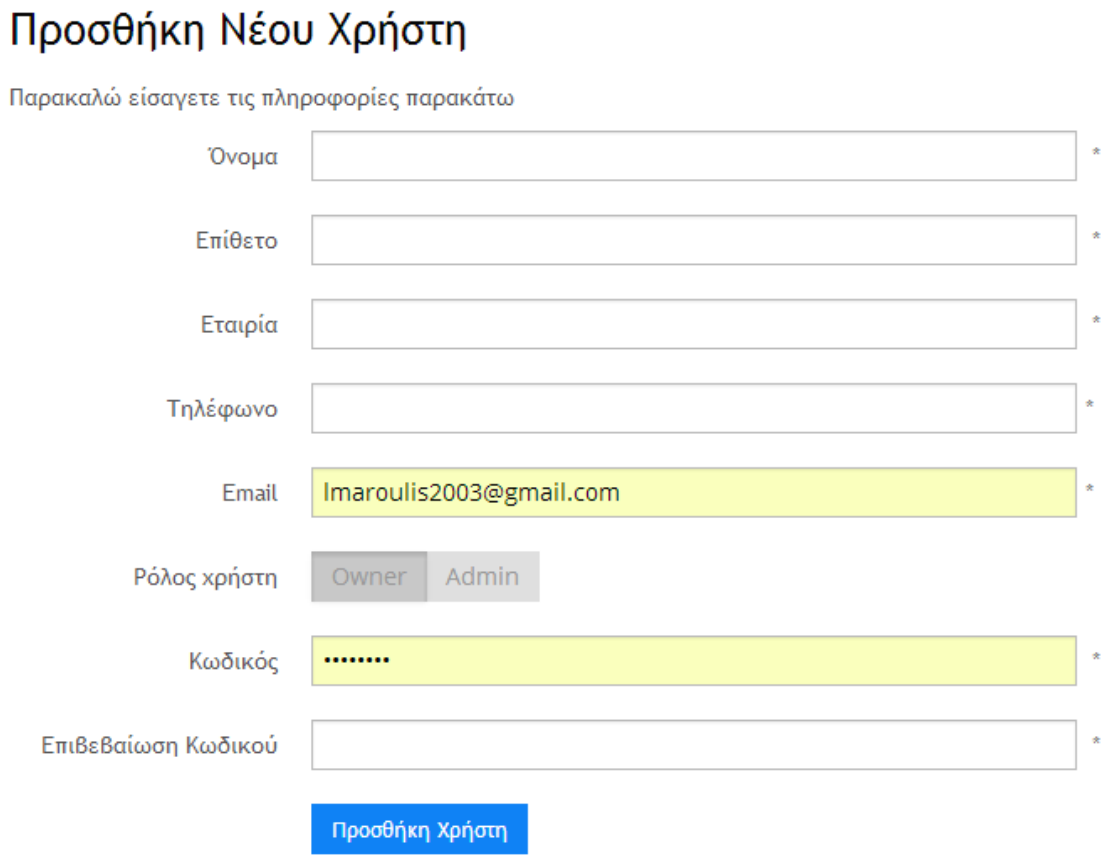



**Εικόνα 15 - Λίστα χρηστών**

Αντίστοιχα εδώ όπως και στις προηγούμενες ενότητες μπορούμε να φιλτράρουμε εκτυπώσουμε είτε να επεξεργαστούμε - διαγράψουμε χρήστες.

### 4.1.10 Προσθήκη νέου χρήστη

Εδώ μπορεί ο διαχειριστής να προσθέσει έναν καινούργιο χρήστη.




**Εικόνα 16 - Προσθήκη νέου χρήστη**

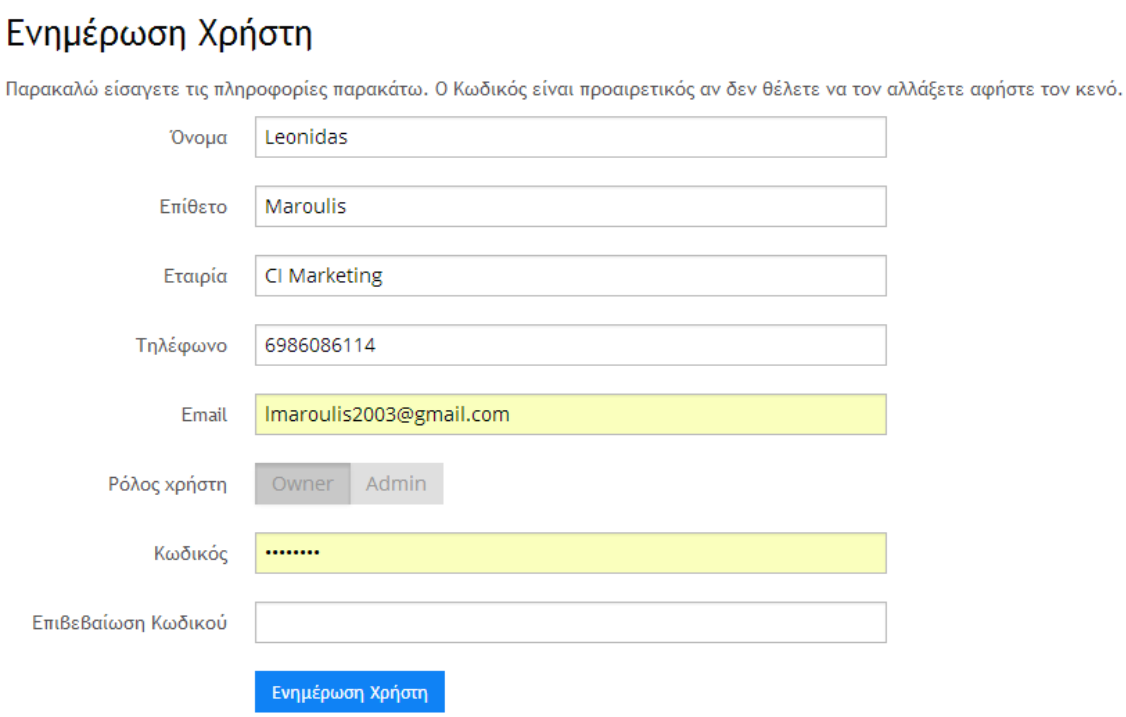
Η πληροφορίες που πρέπει να εισαχθούν είναι οι παρακάτω και είναι όλες υποχρεωτικές:

- ✓ Όνομα χρήστη
- ✓ Επίθετο χρήστη
- ✓ Εταιρία
- ✓ Τηλέφωνο
- ✓ Email (Το οποίο χρησιμοποιήτε για την είσοδο στο σύστημα)
- ✓ Ρόλος χρήστη. Εδώ υπάρχουν 2 ειδών χρήστες ο ιδιοκτήτης ο οποίος έχει πρόσβαση σε όλες τις λειτουργίες του συστήματος και ο διαχειριστής (admin) ο οποίος έχει πρόσβαση μόνο στα φαγητά.
- ✓ Κωδικός
- ✓ Επιβεβαίωση κωδικού

Αφού εισαχθούν τα παραπάνω στοιχεία πατάμε προσθήκη χρήστη και γυρνάμε στην λίστα χρηστών.

### 4.1.11 Επεξεργασία χρήστη

Ο διαχειριστής έχει την δυνατότατα να επεξεργαστή τις πληροφορίες όλων των χρηστών καθώς και τις δικές του όπως φαίνεται παρακάτω.

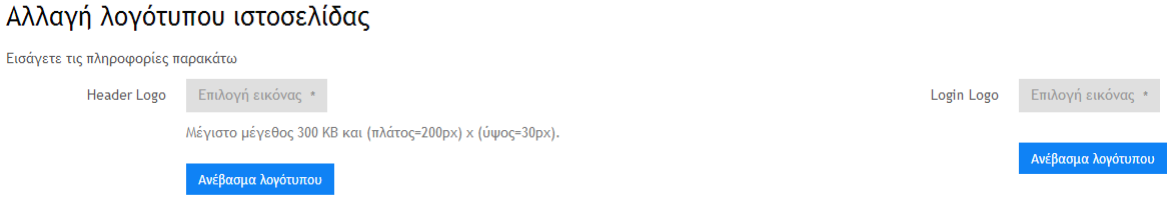



**Εικόνα 17 - Επεξεργασία - Ενημέρωση χρήστη**

Σε περίπτωση που δεν θέλει να αλλάξει τον κωδικό μπορεί να τον αφήσει κενό και έτσι το σύστημα θα διατηρήσει το υπάρχον.

### 4.1.12 Αλλαγή λογοτύπου

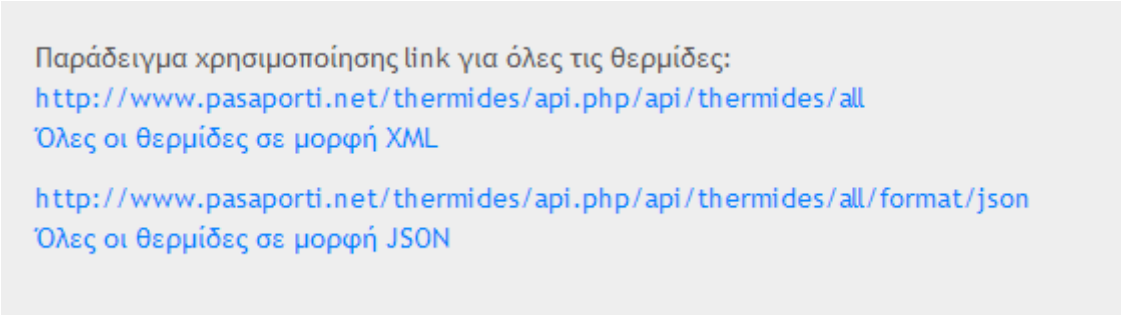
Ο ιδιοκτήτης του συστήματος έχει επίσης την δυνατότητα να αλλάξει το λογότυπο της οθόνης σύνδεσης αλλά και το βασικό που φαίνεται πάνω αριστερά.




Το μέγεθος του βασικού λογοτύπου δεν θα πρέπει να υπερβαίνει τα 200px πλάτος και τα 30px ύψος.

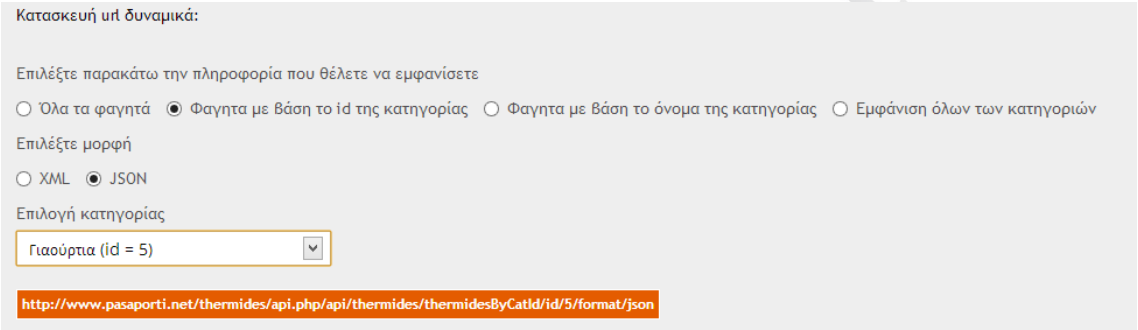
## 4.2 API εφαρμογής

Το API η αλλιώς **Διεπαφή Προγραμματισμού Εφαρμογών (Application Programming Interface)** είναι η διεπαφή προγραμματιστικών διαδικασιών που μια εφαρμογή επιτρέπει προς αυτή αιτήσεις από άλλα προγράμματα και ανταλλαγές δεδομένων. Το συγκεκριμένο API σκοπό είχε να κάνει ευκολότερη την επικοινωνία μεταξύ mobile συσκευής και online βάσης δεδομένων. Τα δεδομένα που προσφέρει είναι όλα τα φαγητά φαγητά με βάση το id της κατηγορίας και εμφάνιση όλων των κατηγοριών. Δεν είναι εφικτό να γίνει εισαγωγή ή διαγραφή στοιχείων από το API (για λόγους ασφαλείας) αλλά μπορεί να γίνει φιλτράρισμα των αποτελεσμάτων. Παρακάτω βλέπουμε ένα παράδειγμα για το πως μπορεί να χρησιμοποιηθεί το API.




**Εικόνα 18 - Παράδειγμα url για εμφάνιση όλων των θερμίδων**

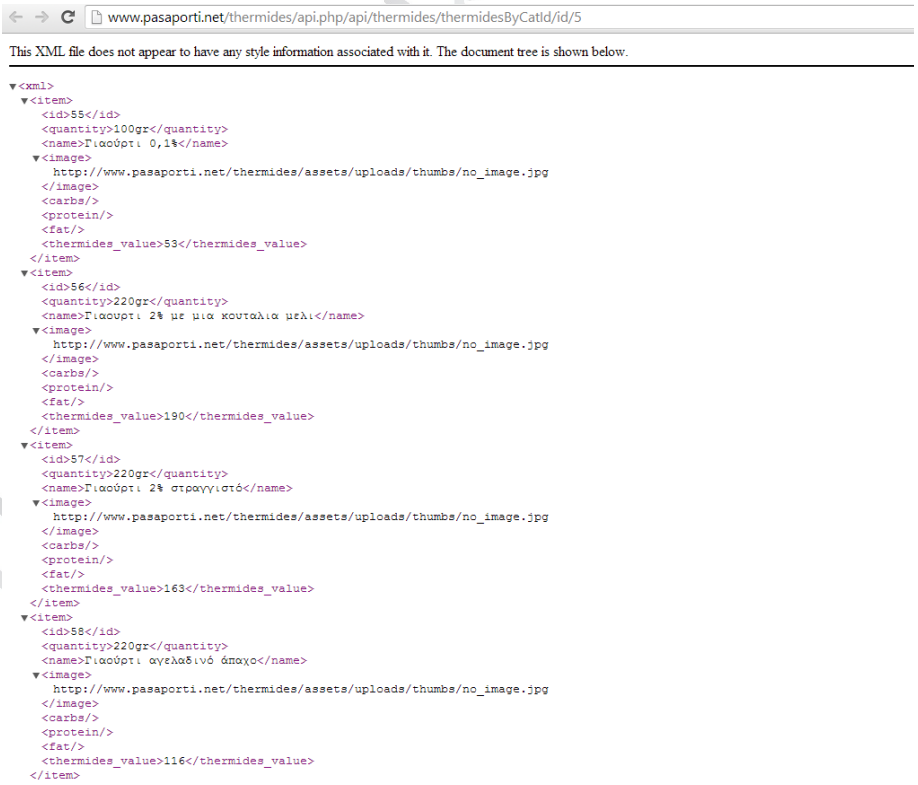
Η παρακάτω φόρμα έχει φτιαχτεί για να γίνει πιο εύκολη η δημιουργία url.




**Εικόνα 19 - Δυναμική κατασκευή URL**

Χρησιμοποιώντας την παραπάνω φόρμα μπορεί ο χρήστης να δημιουργήσει link με βάση τις απαιτήσεις του και να φιλτράρει τα αποτελέσματα που θέλει.

Ένα παράδειγμα αποτελέσματος που μπορεί να γυρίσει το service, σε μορφή XML, φαίνεται παρακάτω:




**Εικόνα 20 - XML αποτέλεσμα (με βάση το category Id)**

Παραπάνω βλέπουμε ένα αρχείο τύπου xml μέσα στο οποίο υπάρχουν <items> και μεταφράζονται σε φαγητά που παρέχουν τις εξής πληροφορίες:

- ✓ Id του φαγητού (μοναδικό)
- ✓ Ποσότητα - Quantity
- ✓ Όνομα φαγητου - Name
- ✓ Εικόνα - image του φαγητού (αν υπάρχει αλλιώς μπαίνει η default εικόνα)
- ✓ Υδατάνθρακες - Carbs
- ✓ Πρωτεΐνες - Protein
- ✓ Λίπος - Fat
- ✓ Αριθμός θερμίδων - thermides_value

Έτσι η παραπάνω πληροφορία είναι εύκολο να ανακτηθεί από απομακρυσμένες εφαρμογές (όπως εφαρμογές κινητών) και να τις προβάλουν χωρίς να είναι απαραίτητο να γίνει σύνδεση με την βάση αλλά και χωρίς να χρειάζεται να γνωρίζουν το source code της εφαρμογής.

## 4.3 Χρήση εφαρμογής στο κινητό

Παρακάτω θα γίνει αναφορά στην χρήση της εφαρμογής από χρήση κινητού με Android.

### 4.3.1 Είσοδος στην εφαρμογή

Για να γίνει είσοδος στην εφαρμογή πρέπει ο χρήστης να πατήσει στο εικονίδιο - λογότυπο της εφαρμογής.

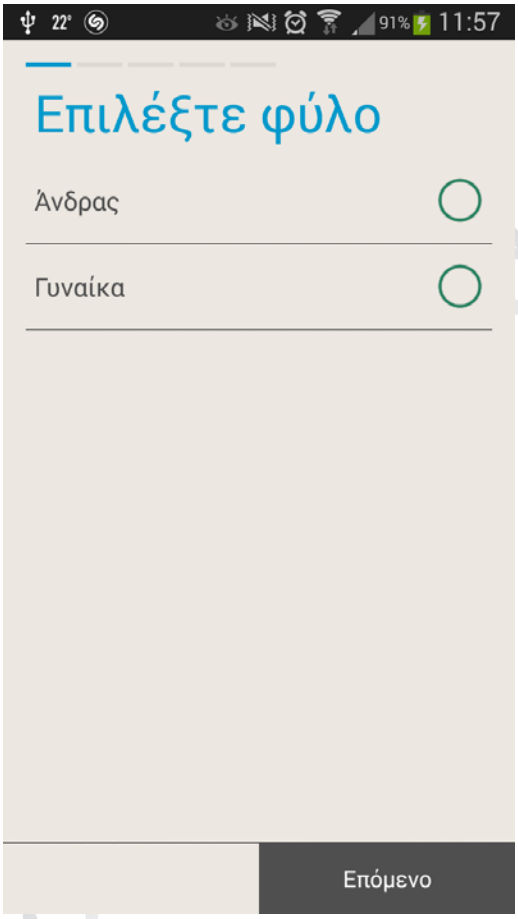
**Εικόνα 21- Εικονίδιο για έναρξη της εφαρμογής**

Με το που πατηθεί το παραπάνω εικονίδιο γίνεται έναρξη της εφαρμογής (launch) και ακολουθεί μια splash-intro οθόνη όπως φαίνεται παρακάτω.


**Εικόνα 22 - Splash/Intro οθόνη**

### 4.3.2 Δημιουργία προφίλ

Την πρώτη φορά που ανοίγει η εφαρμογή ο χρήστης είναι υποχρεωμένος να δημιουργήσει το προφίλ του. Για τον λόγο αυτό δημιουργήθηκε ένας οδηγός που συλλέγει κάποια στοιχεία του χρήστη για να βγάλει αποτελέσματα στο τέλος. Ο οδηγός λειτουργεί με βήματα (steps) και για την μετάβαση σε επόμενες οθόνες είναι απαραίτητη η συμπλήρωση όλων των στοιχείων.




**Εικόνα 23 - Πρώτη οθόνη οδηγού δημιουργίας προφίλ**

Ο οδηγός έχει 5 βήματα (steps) για την ολοκλήρωση δημιουργίας προφίλ. Παρακάτω θα αναφέρουμε αναλυτικά όλες τις οθόνες.

- ✓ Η πρώτη οθόνη ζητάει από τον χρήστη να εισάγει το φύλο του (Άνδρας ή Γυναίκα).
- ✓ Στην δεύτερη οθόνη ζητείται να προσδιοριστεί η φυσική δραστηριότητα. Οι επιλογές εδώ είναι: 1) Πολύ μικρή η οποία μεταφράζεται σε σπάνια άσκηση εώς καθόλου, 2) Μέτρια δηλαδή άσκηση μέχρι 3 φορές την εβδομάδα ή καθημερινές δουλείες στο σπίτι, 3) Έντονη που σημαίνει άσκηση 3-5 φορές την εβδομάδα 4) Καθημερινή άσκηση που προσδιορίζεται είτε με άσκήση στο γυμναστήριο κάθε μέρα είτε με κάποιο επάγγελμα που απαιτεί σωματική άσκηση (οικοδόμος, σερβιτόρος κλπ) 5) Αθλητική δηλαδή προπόνηση επαγγελματική.
- ✓ Η τρίτη οθόνη ζητάει τον προσδιορισμό του στόχου, δηλαδή αν ο χρήστης της εφαρμογής θέλει να διατηρήσει το βάρος του ή να το αλλάξει. Οι επιλογές εδώ είναι: χάσιμο ενός κιλού την εβδομάδα, χάσιμο 1 κιλού τον μήνα, χάσιμο 3 κιλών τον μήνα, διατήρηση βάρους, προσθήκη 1 κιλού την εβδομάδα, προσθήκη 1 κιλού τον μήνα, προσθήκη 3 κιλών τον μήνα.
- ✓ Η τέταρτη οθόνη ζητάει από τον χρήστη το όνομα του, την ηλικία του, το σωματικό του βάρος και το ύψος του.

- ✓ Η πέμπτη και τελευταία οθόνη κάνει ανασκόπηση των στοιχείων που έχει δώσει ο χρήστης και σε περίπτωση που κάτι έχει εισαχθεί λάθος μπορεί να πατηθεί και ο χρήστης θα μεταφερθεί στο αντίστοιχο βήμα (step).


**Εικόνα 24 - Οθόνη ανασκόπησης (πέμπτο βήμα)**

Με το που πατηθεί το κουμπί τέλος συλλέγονται όλα τα δεδομένα που έδωσε ο χρήστης και βγαίνουν αποτελέσματα χρήσιμα για τον χρήστη όπως ο δείκτης BMI και το πόσες θερμίδες πρέπει να καταναλώσει για την πραγματοποίηση του στόχου του.

### 4.3.3 Dashboard επιλογών εφαρμογής

Η οθόνη μετά την δημιουργία προφίλ είναι κεντρική της εφαρμογής και έχει όλες τις διαθέσιμες επιλογές για τον χρήστη.

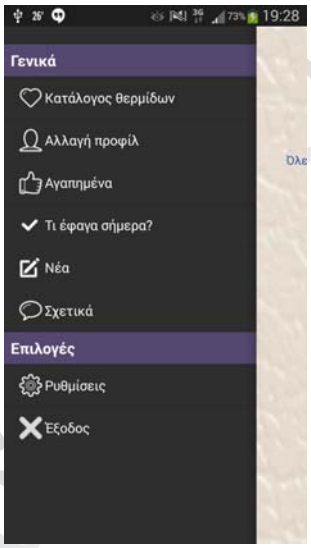

**Εικόνα 25 - Dashoboard εφαρμογής**

Οι επιλογές που έχει ο χρήστης σε αυτή την οθόνη είναι οι ακόλουθες:

- ✓ Όλες οι κατηγορίες. Περιήγηση σε όλες τις κατηγορίες.
- ✓ Ημερολόγιο. Να δει τί έφαγε την συγκεκριμένη μέρα.
- ✓ Νέα. Να δει τα τελευταία νέα από τον χώρο (πηγή in.gr).
- ✓ Αλλαγή προφίλ. Να ξανά εκκινήσει τον οδηγό δημιουργίας προφίλ σε περίπτωση που θέλει να αλλάξει κάποια πληροφορία (όπως το σωματικό του βάρος).
- ✓ Ρυθμίσεις. Να αλλάξει κάποιες βασικές λειτουργίες τις εφαρμογής.
- ✓ Σχετικά. Να βρει πληροφορίες σχετικά με την ανάπτυξη της εφαρμογής.

### 4.3.4 Μενού Επιλογών

Στην εφαρμογή έχει φτιαχτεί ένα μενού με επιλογές έτσι ώστε να είναι πιο εύκολη η περιήγηση μεταξύ των οθονών όπου και αν βρίσκεται ο χρήστης. Το μενού αυτό ανοίγει είτε πατώντας το κουμπί μενού της συσκευή είτε σύροντας το χέρι από αριστερά προς τα δεξιά.




**Εικόνα 26 - Μενού επιλογών εφαρμογής**

Οι επιλογές που έχει ο χρήστης στο μενού είναι:

- ✓ Κατάλογος θερμίδων. Περιήγηση σε όλες τις κατηγορίες.
- ✓ Αλλαγή προφίλ. Να ξανά εκκινήσει τον οδηγό δημιουργίας προφίλ σε περίπτωση που θέλει να αλλάξει κάποια πληροφορία (όπως το σωματικό του βάρος).
- ✓ Αγαπημένα. Λίστα με τα αγαπημένα φαγητά του χρήστη.
- ✓ Τι έφαγα σήμερα. Να δει τί έφαγε την συγκεκριμένη μέρα.
- ✓ Νέα. Να δει τα τελευταία νέα από τον χώρο (πηγή in.gr).
- ✓ Σχετικά. Να βρει πληροφορίες σχετικά με την ανάπτυξη της εφαρμογής.
- ✓ Ρυθμίσεις. Να αλλάξει κάποιες βασικές λειτουργίες τις εφαρμογής.
- ✓ Έξοδος. Έξοδος από την εφαρμογή.

### 4.3.5 Κατάλογος κατηγοριών

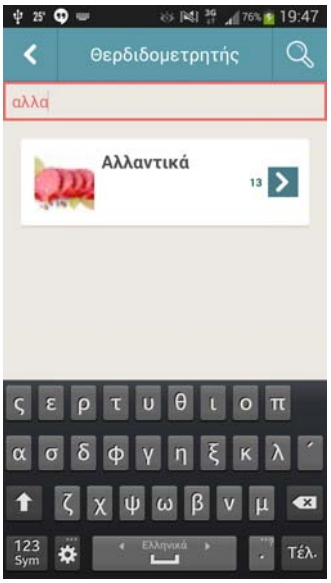
Το κουμπί "Όλες οι κατηγορίες" μας οδηγεί στον κατάλογο με τις διαθέσιμες κατηγορίες φαγητών.


**Εικόνα 27 - Κατάλογος κατηγοριών**

Ο χρήστης μπορεί να περιηγηθεί σε όλες τις διαθέσιμες κατηγορίες και με το που πατήσει σε κάποια από αυτές μπαίνει στην λίστα με τα φαγητά. Η λίστα αυτή δείχνει το όνομα της κατηγορίας την φωτογραφία της κατηγορίας (αριστερά) και τον αριθμό των φαγητών που έχει η κατηγορία.

Η παραπάνω λίστα είναι αποθηκευμένη στην τοπική βάση δεδομένων του κινητού για κάποιες ημέρες (ανάλογα με τις προτιμήσεις του χρήστη που έχουν οριστεί στις ρυθμίσεις τις εφαρμογής) αλλά ο χρήστης μπορεί να ανανεώσει την λίστα όποτε επιθυμεί επικοινωνόντας με την απομακρυσμένη βάση δεδομένων (οπότε είναι απαραίτητη η σύνδεση στο διαδίκτυο) πηγαίνοντας πάνω πάνω και σύροντας την λίστα προς τα κάτω (pull to refresh).

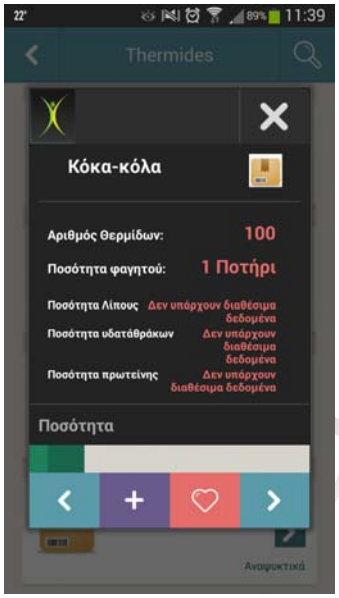
Τέλος υπάρχει η δυνατότητα φιλτραρίσματος της λίστας πατώντας το κουμπί με το εικονίδιο του φακού. Εδώ φιλτράρεται το πρώτο μέρος του ονόματος της κατηγορίας όπως φαίνεται στην παρακάτω εικόνα.




**Εικόνα 28 - Φιλτράρισμα λίστας**

### 4.3.6 Κατάλογος φαγητών / Αναλυτικές πληροφορίες φαγητού

Ο κατάλογος των φαγητών έχει τις ίδιες λειτουργίες με τον κατάλογο θερμίδων (αναζήτηση και σύρσιμο για ενημέρωση (εφόσον υπάρχει διαθέσιμη σύνδεση στο διαδίκτυο) και ο χρήστης όταν πατήσει σε κάποιο φαγητό ανοίγει η καρτέλα με τις διαθέσιμες πληροφορίες του.



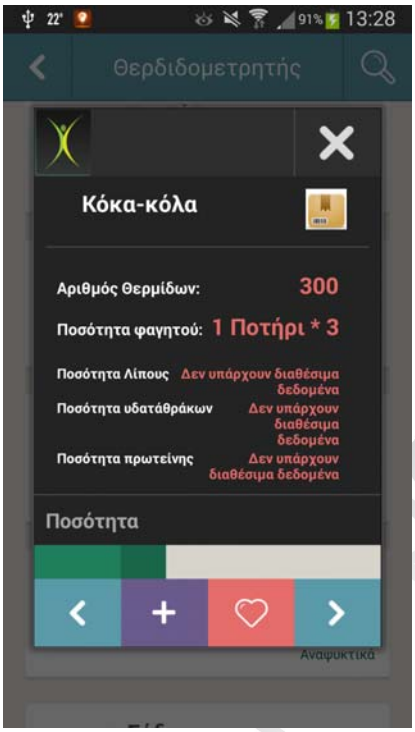

**Εικόνα 29 - Αναλυτικές Πληροφορίες φαγητού**

Στην καρτέλα αναλυτικών πληροφοριών της τροφής - φαγητού φαίνονται τα εξής:

- ✓ Όνομα της τροφής.
- ✓ Εικόνα της τροφής (δεξιά από τον όνομα).
- ✓ Αριθμός θερμίδων (με βάση την ποσότητα)
- ✓ Ποσότητα που αντιστοιχούν οι θερμίδες.
- ✓ Ποσότητα λίπους ανά 100γρ.
- ✓ Ποσότητα υδατανθράκων ανά 100γρ.
- ✓ Ποσότητα πρωτεΐνης ανά 100γρ.

Οι ενέργειες που μπορεί να κάνει στην συγκεκριμένη καρτέλα ο χρήστης είναι:

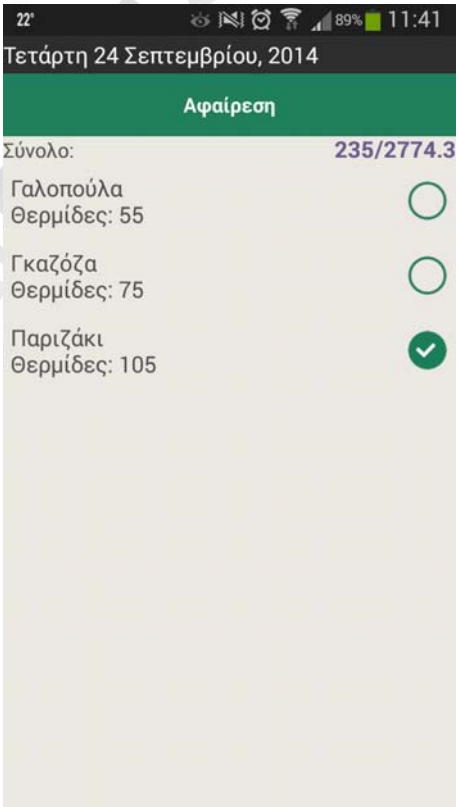
1. Να κλείσει την καρτέλα πατώντας πάνω στο εικονίδιο Χ.
2. Να περιηγηθεί στις υπόλοιπες τροφές της κατηγορίας χρησιμοποιώντας τα βελάκια αριστερά - δεξιά.
3. Να αποθηκεύσει το φαγητό στα αγαπημένα πατώντας πάνω στο εικονίδιο της καρδούλας.
4. Να αλλάξει την ποσότητα του φαγητού σέρνοντας την πράσινη μπάρα (είτε αριστερά, είτε δεξιά). Ανάλογα την ποσότητα που θα επιλέξει θα υπολογιστούν και οι αντίστοιχες θερμίδες που αντιστοιχούν στην επιλογή. Για παράδειγμα αν ο χρήστης επιλέξει 2 ποτήρια coca cola που το κάθε ποτήρι έχει 100 θερμίδες η εφαρμογή θα εμφανίσει 200 θερμίδες.
5. Να προσθέσει στο τι έφαγε σήμερα την συγκεκριμένη τροφή (με την ποσότητα που έχει ορίσει).




**Εικόνα 30- Αλλαγή ποσότητας από τον χρήστη**

### 4.3.7 Τι έφαγα σήμερα οθόνη

Η οθόνη τι έφαγα σήμερα δείχνει όλα τα φαγητά της ημέρας που έχει καταναλώσει ο χρήστης.




**Εικόνα 31 - Οθόνη τι έφαγα σήμερα**

Σε αυτή την οθόνη ο χρήστης μπορεί να δει το σύνολο τον θερμίδων που έχει καταναλώσει μέσα στην ημέρα (πάνω αριστερά) καθώς και τις θερμίδες που πρέπει να πάρει για να πετύχει τον στόχο του. Σε περίπτωση που έχει κάνει κάποια λάθος εισαγωγή φαγητού μπορεί να το επιλέξει και να πατήσει το κουμπί αφαίρεση.

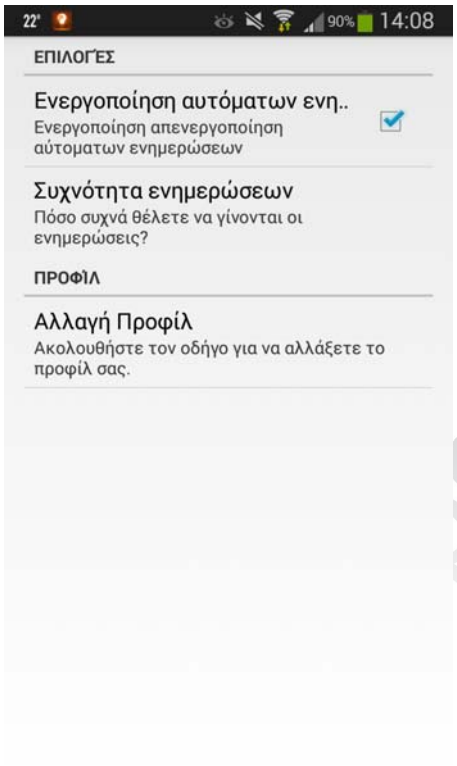
### 4.3.8 Οθόνη τελευταίων νέων

Η οθόνη με τα τελευταία νέα τραβάει τα τελευταία νέα από το in.gr και την κατηγορία υγεία και διατροφή. Για να δει αυτά τα νέα ο χρήστης είναι απαραίτητο να έχει σύνδεση στο internet.


**Εικόνα 32 - Τελευταία Νέα**

### 4.3.8 Οθόνη επιλογών ρυθμίσεων

Η οθόνη ρυθμίσεων δημιουργήθηκε για να υπάρχουν όλες οι διαθέσιμες επιλογές που μπορεί να κάνει ο χρήστης έτσι ώστε να παραμετροποιήσει την εφαρμογή. Στην συγκεκριμένη έκδοση της εφαρμογής ο χρήστης έχει μόνο μια επιλογή (αλλά σε μελλοντικές ενημερώσεις θα προστεθούν και άλλες). Η επιλογή που έχει ο χρήστης είναι να αλλάξει την συχνότητα των ενημερώσεων είτε να την αφαιρέσει τελείως.




**Εικόνα 33 - Οθόνη επιλογών**

Η επιλογές που έχει ο χρήστης εδώ είναι οι εξής:

1. Ενεργοποίηση - απενεργοποίηση αυτόματων ενημερώσεων σε συγκεκριμένο χρονικό διάστημα. Σε περίπτωση που απενεργοποιηθεί αυτή η δυνατότατα η εφαρμογή θα απαιτεί μόνιμη σύνδεση για την προβολή των φαγητών καθώς θα πρέπει να επικοινωνεί κάθε φορά με την απομακρυσμένη βάση δεδομένων.
2. Συχνότητα ενημερώσεων. Εδώ ο χρήστης μπορεί να ορίσει κάθε πότε θέλει η εφαρμογή να ενημερώνεται. Διαθέσιμες επιλογές είναι κάθε μέρα, κάθε εβδομάδα, κάθε 2 εβδομάδες και κάθε μήνα. Η επιλογή αυτή είναι εξαρτώμενη από την πιο πάνω επιλογή (δηλαδή αν απενεργοποιηθούν οι αυτόματες ενημερώσεις απενεργοποιείτε και αυτή η επιλογή).
3. Αλλαγή προφίλ. Ο χρήστης από εδώ έχει επιλογή να ανοίξει τον οδηγό δημιουργίας προφίλ για να αλλάξει το προφίλ του.

# 5. Υλοποίηση εφαρμογής - Αρχιτεκτονική Συστήματος

Η ανάπτυξη κάθε λογισμικού πρέπει να περνάει από κάποια στάδια. Αυτά τα στάδια βοηθάνε τον προγραμματιστή να μετατρέψει τις απαιτήσεις του πελάτη σε εφικτές λύσεις. Παρακάτω θα περιγράψουμε κάποια από τα στάδια για την δημιουργία της εφαρμογής Θερμιδομετρητής όπως επίσης τι τεχνολογίες χρησιμοποιήθηκαν.

## 5.1 Ανάλυση απαιτήσεων και στόχων

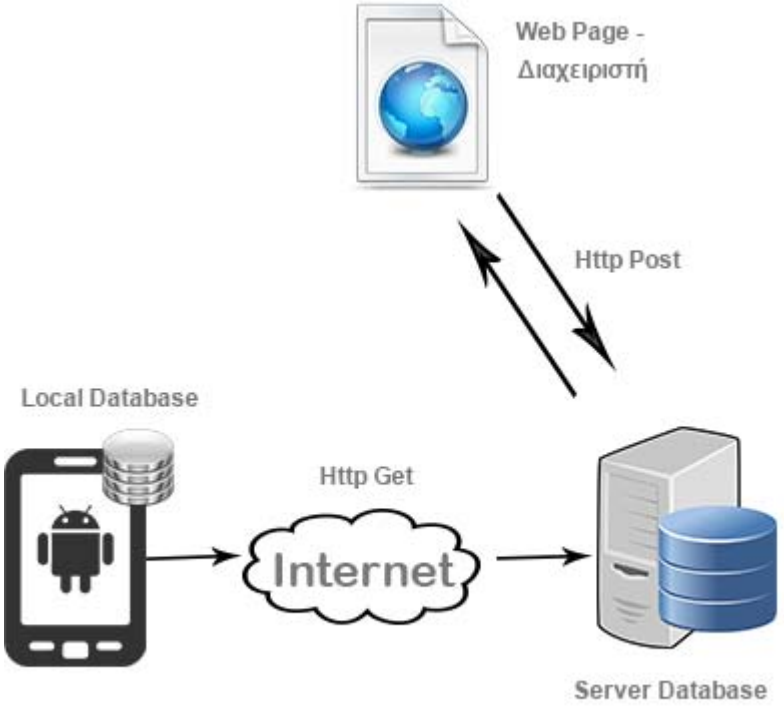
Στόχος του θερμιδομετρητή είναι η σωστή ενημέρωση και καθοδήγηση του χρήστη σχετικά με την ημερήσια κατανάλωση θερμίδων. Ο χρήστης θα μπορεί να δημιουργεί στόχους και να παρακολουθεί την πορεία αυτών. Πιο συγκεκριμένα οι απαιτήσεις που θα πρέπει να ικανοποιούνται είναι οι εξής:

- ✓ *Φιλικό περιβάλλον (User Friendly):* Σκοπός της εφαρμογής είναι να βοηθήσει και να καθοδηγήσει τους χρήστες στο να χρησιμοποιούν την εφαρμογή καθημερινά. Για να γίνει αυτό εφικτό θα πρέπει να δημιουργηθεί ένα περιβάλλον το οποίο θα είναι εύκολο στην χρήση από μη έμπειρους χρήστες.
- ✓ *Δημιουργία προφίλ.* Σκοπός της εφαρμογής δεν είναι να γίνει μια απλή αναφορά στα στοιχεία των φαγητών αλλά να βοηθήσει τον χρήστη να καθορίσει τον στόχο του με βάση τα στοιχεία που θα εισάγει. Η εφαρμογή δεν θα πρέπει να είναι σε θέση να προχωρήσει σε επόμενες οθόνες αν δεν δημιουργηθεί προφίλ.
- ✓ *Αποθήκευση δεδομένων και αποτελεσμάτων προφίλ*. Η εφαρμογή θα πρέπει να αποθηκεύει τα αποτελέσματα που βγήκαν μετά την συμπλήρωση του προφίλ έτσι ώστε ο χρήστης να μην χρειάζεται να το συμπληρώνει συνεχεία. Ο χρήστης θα πρέπει να είναι σε θέση να αλλάξει το προφίλ όποτε αυτός επιθυμεί (πχ αλλαγή σωματικού βάρους).
- ✓ *Δημιουργία διαχειριστικού συστήματος για τον ιδιοκτήτη*. Ο ιδιοκτήτης της εφαρμογής δεν θα είναι απαραίτητο να είναι προγραμματιστής για να αλλάξει τις εγγραφές. Για τον λόγο αυτό είναι απαραίτητο να δημιουργηθεί ένα περιβάλλον εύκολο στην χρήση από κάποιον μη γνώστη.
- ✓ *Η εφαρμογή θα μπορεί να δουλεύει και εκτός σύνδεσης*. Η επικοινωνία με τον server θα γίνεται την πρώτη φορά που θα ανοίγει. Μετά η εφαρμογή θα πρέπει να είναι σε θέση να αποθηκεύει τα δεδομένα έτσι ώστε να μην χρειάζεται να ξανασυνδεθεί. Μετά από συγκεκριμένο χρονικό διάστημα η εφαρμογή θα πρέπει να είναι σε θέση να ενημερώνει ξανά τα στοιχεία με το που βρει διαθέσιμη σύνδεση.
- ✓ *Ενημέρωση στα δεδομένα του κινητού από τον χρήστη*. Ο χρήστης μπορεί να κάνει update όποτε επιθυμεί αρκεί να έχει διαθέσιμη σύνδεση στο internet.
- ✓ *Αναζήτηση*. Ο χρήστης θα μπορεί να φιλτράρει τα φαγητά. Γράφοντας στην αναζήτηση θα εμφανίζονται μόνο τα φαγητά/κατηγορίες που ταιριάζουν στην λίστα.
- ✓ *Αποθήκευση ημερήσιας κατανάλωσης φαγητών*. Ο χρήστης θα είναι σε θέση να αποθηκεύει στην εφαρμογή το τι έφαγε όλη την ημέρα (όπως επίσης και να αφαιρεί σε περίπτωση που έκανε λάθος) και με βάση τις ποσότητες που έφαγε θα εμφανίζεται το σύνολο τον θερμίδων.
- ✓ *Προσθήκη στα αγαπημένα.* Ο χρήστης θα πρέπει να είναι σε θέση να έχει αγαπημένα φαγητά (φαγητά που καταναλώνει συχνότερα) έτσι ώστε να είναι πιο εύκολη η περιήγηση.
- ✓ *Υπολογισμός θερμίδων με βάση την ποσότητα*. Επειδή δεν είναι εφικτό να περαστούν στην βάση όλοι οι πιθανοί συνδυασμοί ποσοτήτων, η εφαρμογή θα πρέπει να είναι σε θέση να υπολογίζει τις θερμίδες με βάση την ποσότητα που βάζει ο χρήστης.

- ✓ Τελευταία νέα του χώρου. Η εφαρμογή πρέπει να παρέχει στον χρήστη τις τελευταίες εξελίξεις σε θέματα διατροφής και υγείας. Αυτό θα πρέπει να επιτευχθεί χρησιμοποιώντας μια αξιόπιστη πηγή με νέα.

## 5.2 Τεχνικές Προδιαγραφές

Για τις ανάγκες της διπλωματικής εργασίας και του θερμιδομετρητή αναπτύχθηκε ένα διαχειριστικό σύστημα το οποίο θα κάνει εφικτό στον ιδιοκτήτη της εφαρμογής την ενημέρωση της. Το σύστημα αυτό αποτελείται από μια εφαρμογή στην μεριά του πελάτη (client) και ένα ιστοχώρο που θα είναι συνδεμένος στην βάση από την μεριά του εξυπηρετητή (server). Παρακάτω μπορούμε να δούμε το διάγραμμα της τοπολογίας του συστήματος:




**Εικόνα 34 - Τοπολογία συστήματος**

Το **διαχειριστικό μέρος** της πτυχιακής (η ιστοσελίδα) στήθηκε χρησιμοποιώντας ως βάση το δημοφιλές **PHP framework Code Igniter**. Το framework αυτό χρησιμοποίει PHP 5 ή οποία είναι αντικειμενοστραφής και την λογική του **MVC** (Model View Controller). Η λογική MVC προσφέρει αρκετά πλεονεκτήματα όπως επεκτασιμότητα, ασφάλεια, ευκολία στον κώδικα, ταχύτερη ανάπτυξη (λόγω πολλών βιβλιοθηκών) και φορητότητα. Για να γίνει εφικτή η προβολή της ιστοσελίδας από όλες τις αναλύσεις έγινε χρήση του responsive **css framework bootstrap**. Η βάση δεδομένων που χρησιμοποιήθηκε είναι η **MySQL** της Oracle και η δοκιμές έγιναν σε τοπικό περιβάλλον χρησιμοποιώντας το πρόγραμμα WAMP.

Τα libraries (βιβλιοθήκες) που χρησιμοποιήθηκαν στην ιστοσελίδα και είναι απαραίτητα για την σωστή προβολή είναι τα παρακάτω:

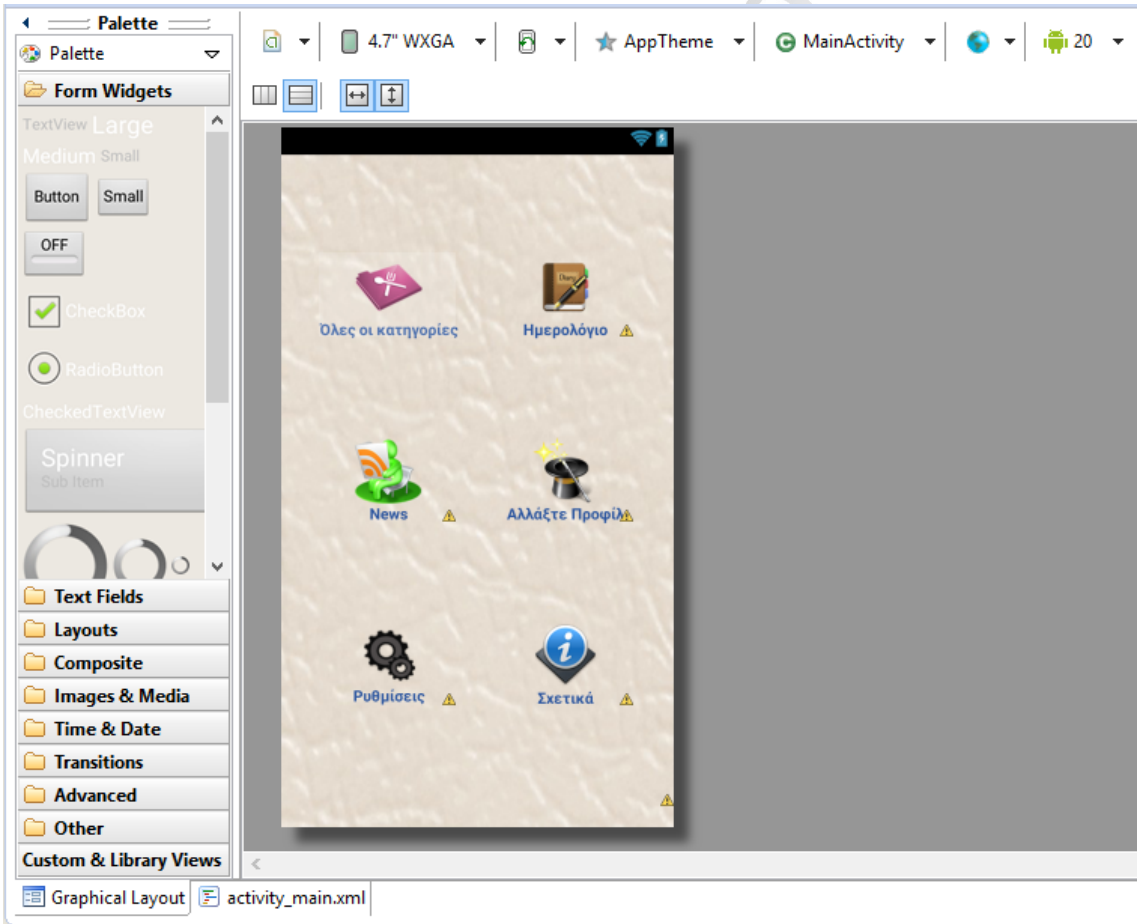
- ✓ *Form Validation Library*. Για server side validation
- ✓ *CodeIgniter Rest Server.* Για την δημιουργία του service
- ✓ *Ignited Datatables.* Για την προβολή λιστών (φαγητά, χρήστες, κατηγορίες κλπ).

Η εφαρμογή στο κινητό υλοποιήθηκε στην πλατφόρμα **Android** και θα παίζει σε όλα τα κινητά με έκδοση Android πάνω από 2.2 (Froyo). Η γλώσσα που χρησιμοποιήθηκε είναι η Java με την βοήθεια του **Android SDK** (software development kit) το οποίο παρέχει κάποια εργαλεία

για ευκολότερη ανάπτυξη (όπως debugger, emulator κλπ). Το IDE (Integrated Development Environment) που έγινε η ανάπτυξη είναι το **eclipse** με προεγκατεστημένο το ADT plugin (Android Development Tools). Η εφαρμογή σχεδιάστηκε για να παίζει σωστά στις περισσότερες και πιο δημοφιλείς οθόνες. Οι δοκιμές έγιναν σε κινητό με έκδοση Android 4.3 και με ανάλυση οθόνης 4.7 ίντσες. Η προγραμματιστική γλώσσα που χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής είναι η Java. Οι απαραίτητες εξωτερικές βιβλιοθήκες για την εφαρμογή είναι οι ακόλουθες:

1. ListView Animations. Παροχή διαφόρων εφε για τις λίστες.
2. Sliding Menu. Αυτή η βιβλιοθήκη δίνει την δυνατότητα να ανοίγει το menu από το πλάι και σε παλαιότερες εκδώσεις.
3. Joda Time. Βιβλιοθήκη για χρόνο και ώρα της Java.

Για τον σχεδιασμό των οθονών χρησιμοποιήθηκε ο designer του eclipse ενώ κάποια τμήματα κώδικα γράφτηκαν κατευθείαν στην γλώσσα XML με την βοήθεια του editor του eclipse.




**Εικόνα 35 - Designer Eclipse**

Στην εφαρμογή υπάρχουν οι παρακάτω βασικές οθόνες η οποίες σχεδιάστηκαν ξεχωριστά:

- ✓ Λίστα κατηγοριών. Είναι η λίστα με όλες οι κατηγορίες και με το πάτημα μιας κατηγορίας ο χρήστης πηγαίνει στα φαγητά της συγκεκριμένης κατηγορίας.
- ✓ Λίστα φαγητών. Η λίστα με τα φαγητά ανά κατηγορία.
- ✓ Λίστα με αγαπημένα. Λίστα για εμφάνιση των αγαπημένων φαγητών του χρήστη.
- ✓ Αναλυτική οθόνη φαγητού. Σε αυτή την οθόνη φαίνονται όλες οι πληροφορίες των φαγητών (θερμιδική αξία κλπ).

- ✓ Ημερολόγιο με τα φαγητά που έχει καταναλώσει ο χρήστης σήμερα.
- ✓ Τελευταία νέα από τον χώρο της υγείας και της διατροφής.
- ✓ Οδηγός προφίλ. Ο χρήστης μπορεί όποτε θέλει να επαναλάβει τον οδηγό για αλλαγή του προφίλ του.
- ✓ Ρυθμίσεις συστήματος. Κάποιες βασικές ρυθμίσεις της εφαρμογής.
- ✓ Σχετικά. Οθόνη με λίγα λόγια για τον προγραμματιστή.

### 5.2.1 Απομακρυσμένη Βάση Δεδομένων

Για το διαχειριστικό περιβάλλον χρησιμοποιήθηκε η MySQL της oracle η οποία στην συνέχεια συνδέθηκε με το PHP framework code-igniter. Η δομή της βάσης δεδομένων φαίνεται στην παρακάτω εικόνα.

| Table | Action | Rows | Type | Collation | Size | Overhead |
|---|---|---|---|---|---|---|
| categories | Browse Structure Search Insert Empty Drop | ~22 | InnoDB | utf8_general_ci | 32 KiB | - |
| groups | Browse Structure Search Insert Empty Drop | ~2 | InnoDB | utf8_general_ci | 16 KiB | - |
| settings | Browse Structure Search Insert Empty Drop | ~0 | InnoDB | utf8_general_ci | 16 KiB | - |
| thermides | Browse Structure Search Insert Empty Drop | ~162 | InnoDB | utf8_general_ci | 80 KiB | - |
| users | Browse Structure Search Insert Empty Drop | ~3 | InnoDB | utf8_general_ci | 16 KiB | - |
| users_groups | Browse Structure Search Insert Empty Drop | ~3 | InnoDB | utf8_general_ci | 16 KiB | - |
| 6 table(s) | Sum | ~18 | InnoDB | utf8_general_ci | 176 KiB | 0 B |

**Εικόνα 36- Δομή απομακρυσμένης βάσης**

Οι πίνακες της βάσης δεδομένων είναι οι ακόλουθοι:

1. Πίνακας categories. Αυτός ο πίνακας αποθηκεύει όλες τις κατηγορίες των θερμίδων.
2. Πίνακας groups. Πίνακας με όλα τα διαθέσιμα user group της εφαρμογής (admin και owner.
3. Πίνακας settings. Εδώ αποθηκεύονται οι προτιμήσεις του ιδιοκτήτη (θέμα ιστοσελίδας, page title κλπ).
4. Πίνακας thermides. Ο πιο βασικός πίνακας της εφαρμογής που αποθηκεύει τα φαγητά με τις πληροφορίες.
5. Πίνακας users. Όλοι οι λογαριασμοί χρηστών με τα στοιχεία τους (όπως τηλέφωνο, εταιρία κλπ).
6. Πίνακας user_groups. Εδώ γίνεται σύνδεση των πινάκων users και groups με αντιστοίχηση του user id με το group id για να δηλωθούν οι ρόλοι.

Ο πιο σημαντικός πίνακας του διαχειριστικού είναι ο πίνακας με τα φαγητά. Παρακάτω φαίνονται οι στήλες του.

| | id | name | image | category_id | quantity | thermides_value | details | carbs | fat | protein |
|---|---|---|---|---|---|---|---|---|---|---|
| Edit Copy Delete | 5 | Γαλοπούλα | no_image.jpg | 2 | 1 Φέτα | 55 | | NULL | NULL | NULL |
| Edit Copy Delete | 6 | Γαλοπούλα Καπνιστή | no_image.jpg | 2 | 1 Φέτα του τόστ | 16 | | NULL | NULL | NULL |
| Edit Copy Delete | 7 | Ζαμπόν διαίτης | no_image.jpg | 2 | 100gr | 150 | | NULL | NULL | NULL |
| Edit Copy Delete | 8 | Ζαμπόν μετρίου πάχους | no_image.jpg | 2 | 100γρ | 340 | | NULL | NULL | NULL |
| Edit Copy Delete | 10 | Βυσσινάδα | no_image.jpg | 1 | 1 Ποτήρι | 150 | | NULL | NULL | NULL |
| Edit Copy Delete | 11 | Γκαζόζα | no_image.jpg | 1 | 1 Ποτήρι | 75 | | NULL | NULL | NULL |
| Edit Copy Delete | 12 | Κόκα-κόλα | no_image.jpg | 1 | 1 Ποτήρι | 100 | | NULL | NULL | NULL |
| Edit Copy Delete | 13 | Λεμονάδα | no_image.jpg | 1 | 1 Ποτήρι | 75 | | NULL | NULL | NULL |
| Edit Copy Delete | 14 | Πορτοκαλάδα κόκκινη (με ανθρακικό) | no_image.jpg | 1 | 1 Ποτήρι | 85 | | NULL | NULL | NULL |
| Edit Copy Delete | 15 | Σόδα | no_image.jpg | 1 | 1 Ποτήρι | 5 | | NULL | NULL | NULL |
| Edit Copy Delete | 16 | Λακέρδα | no_image.jpg | 2 | 100gr | 284 | | NULL | NULL | NULL |
| Edit Copy Delete | 17 | Λουκάνικα Βιέννης βραστά | no_image.jpg | 2 | 1 | 35 | | NULL | NULL | NULL |
| Edit Copy Delete | 18 | Λουκάνικα Φρανκφούρτης βραστά | no_image.jpg | 2 | 1 | 110 | | NULL | NULL | NULL |

**Εικόνα 37 - themides table**

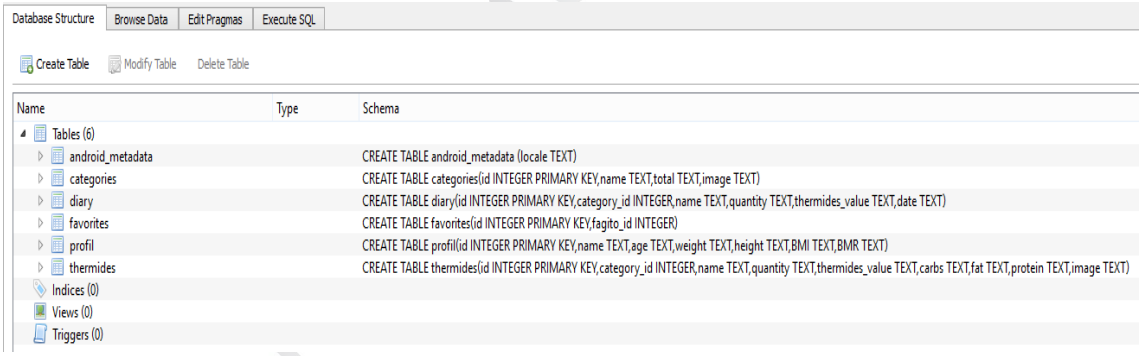
Ξεκινώντας από τον πίνακα categories στο πρώτο πεδίο του πίνακα βλέπουμε το (id) της κατηγορίας το οποίο είναι μοναδικό και διαχωρίζει τις κατηγορίες μεταξύ τους. Στην συνέχεια αποθηκεύονται το όνομα της κατηγορίας (name) και το μέγεθος σε αριθμό των φαγητών που ανήκουν στην συγκεκριμένη κατηγορία (total). Τέλος αποθηκεύεται το όνομα της εικόνας που έχει αποθηκευτεί τοπικά. Η προεπιλεγμένη τιμή είναι μια απλή εικόνα με όνομα no_image.jpg.

Ο πίνακας groups που αποθηκεύει όλα τα διαθέσιμα group που υπάρχουν στην εφαρμογή έχει σαν πεδία το group id το οποίο είναι πρωτεύον κλειδί και τα πεδία name και description.

Ο πίνακας thermides αποθηκεύει όλες τις πληροφορίες για τα φαγητά και περιέχει τα παρακάτω πεδία: id του φαγητού, id της κατηγορίας που ανήκει το φαγητό (catid), το όνομα του φαγητού (name), ποσότητα που αναλογούν οι θερμίδες (quantity), τιμή θερμίδων (thermides_value), ποσότητα λίπους ανά 100 γραμμάρια (fat), ποσότητα υδαταθράκων ανά 100 γραμμάρια (carbs), ποσότητα πρωτεΐνης ανά 100 γραμμάρια (protein), όνομα εικόνας για το συγκεκριμένο φαγητό (image) Παρακάτω φαίνονται κάποιες εγραφές του πίνακα thermides.

### 5.2.2 Βάση Δεδομένων στο κινητό

Τα περισσότερα δεδομένα από την απομακρυσμένη βάση εισάγονται με την βοήθεια του REST service το οποίο συνδέετε με την απομακρυσμένη βάση δεδομένων και εμφανίζει τα δεδομένα σε μορφή XML και JSON. Το κινητό από την άλλη μεριά ζητάει συγκεκριμένο URL (ανάλογα με τις ενέργειες του χρήστη). Έτσι ο χρήστης με το που επιλέξει, για παράδειγμα, κατηγορίες στην εφαρμογή θα γίνει μια κλήση (request) στο URL που επιστρέφει όλες τις κατηγορίες. Η εφαρμογή αφού κατεβάσει όλα τα δεδομένα θα τα αποθηκεύσει στην τοπική βάση δεδομένων έτσι ώστε να μην χρειαστεί να συνδεθεί ο χρήστης στο διαδίκτυο την επόμενη φορά που θα την ανοίξει. Η βάση δεδομένων που χρησιμοποιείται στο κινητό είναι η SQLite η οποία είναι συμβατή με το πρότυπο ISO SQL. Το όνομα της βάσης δεδομένων που δημιουργείται στην εφαρμογή είναι το themidesDB. Η δομή της βάσης δεδομένων φαίνεται παρακάτω:




**Εικόνα 38 - Δομή βάσης δεδομένων κινητού**

Ο σκελετός της βάσης όπως φαίνεται παραπάνω διαφέρει λίγο από την αντίστοιχη απομακρυσμένη βάση ενώ οι πίνακες thermides και categories παραμένουν ίδιοι. Πιο συγκεκριμένα οι πίνακες που χρειάζονται στην εφαρμογή είναι οι παρακάτω:

1. Πίνακας android_metadata. Ο πίνακας αυτός δημιουργείται αυτόματα από το λειτουργικό σύστημα για να αποθηκεύσει την περιοχή (locale).
2. Πίνακας categories. Ο πίνακας αυτός αποθηκεύει όλα τα στοιχεία των κατηγοριών.
3. Πίνακας diary. Ο πίνακας αυτός αποθηκεύει ότι έφαγε ο χρήστης με βάση την ημερομηνία.
4. Πίνακας favorites. Ο χρήστης έχει την δυνατότητα να έχει τα αγαπημένα του φαγητά σε ξεχωριστή λίστα.
5. Πίνακας profil. Εδώ αποθηκεύονται όλα τα στοιχεία του χρήστη (όπως σωματικό βάρος, ηλικία, στόχος).

6. Πίνακας thermides. Ο πίνακας αυτός αποθηκεύει όλα τα φαγητά που γυρνάει το service.

Ο πίνακας categories αποθηκεύει όλες τις κατηγορίες και παρακάτω μπορούμε να δούμε κάποιες εγγραφές.

| | id | name | total | image |
|---|---|---|---|---|
| | Filter | Filter | Filter | Filter |
| 1 | 1 | Αναψυκτικά | 7 | anapsiktika.jpg |
| 2 | 2 | Αλλαντικά | 14 | allantika.jpg |
| 3 | 3 | Βούτυρα | 8 | no_image.jpg |
| 4 | 4 | Γάλατα | 19 | no_image.jpg |
| 5 | 5 | Γιαούρτια | 7 | no_image.jpg |
| 6 | 6 | Γλυκά | 49 | no_image.jpg |
| 7 | 7 | Γλυκά Κουταλιού | 5 | no_image.jpg |
| 8 | 8 | Γλυκαντικά | 5 | no_image.jpg |
| 9 | 9 | Δημητριακά | 17 | no_image.jpg |

**Εικόνα 39 - Εγγραφές Πίνακα Categories**

Στο πρώτο πεδίο του πίνακα αποθηκεύεται το (id) της κατηγορίας το οποίο είναι μοναδικό και διαχωρίζει τις κατηγορίες μεταξύ τους. Στην συνέχεια αποθηκεύονται το όνομα της κατηγορίας (name) και το μέγεθος σε αριθμό των φαγητών που ανήκουν στην συγκεκριμένη κατηγορία (total). Τέλος αποθηκεύεται το όνομα της εικόνας που έχει αποθηκευτεί τοπικά. Η προεπιλεγμένη τιμή είναι μια απλή εικόνα με όνομα no_image.jpg.

Ο πίνακας diary αποθηκεύει το όνομα (name) του φαγητού που έφαγε ο χρήστης καθώς επίσης και το (category_id) για προβολή της κατηγορίας που ανήκει το φαγητό. Τέλος αποθηκεύει την θερμιδική αξία του φαγητού (thermides_value) με βάση την ποσότητα (quantity) που έχει ορίσει ο χρήστης και την ημερομηνία (date) που έγινε το insert .

Ο πίνακας favorites αποθηκεύει μόνο το id του φαγητού που έχει προστεθεί στα αγαπημένα από τον χρήστη. Επίσης διαθέτει και μια το id του favorite το οποίο είναι auto increment (ανεβαίνει αυτόματα με το που γίνει μια εγγραφή).

Ο πίνακας profil αποθηκεύει τα δεδομένα του χρήστη. Στο πρώτο πεδίο του πίνακα αποθηκεύεται το όνομα (name) του χρήστη και ακολουθούν η ηλικία (age), το βάρος (weight), το ύψος (height), ο δείκτης μάζας σώματος (BMI) και ο Βασικός Μεταβολικός Ρυθμός (BMR).

Ο τελευταίος πίνακας της βάσης του κινητού και ο πιο σημαντικός είναι ο thermides ο οποίος είναι ακριβώς ο ίδιος με τον πίνακα του διαχειριστικού και περιέχει τα παρακάτω πεδία: id του φαγητού, id της κατηγορίας που ανήκει το φαγητό (catid), το όνομα του φαγητού (name), ποσότητα που αναλογούν οι θερμίδες (quantity), τιμή θερμίδων (thermides_value), ποσότητα λίπους ανά 100 γραμμάρια (fat), ποσότητα υδαταθράκων ανά 100 γραμμάρια (carbs), ποσότητα πρωτεΐνης ανά 100 γραμμάρια (protein), όνομα εικόνας για το συγκεκριμένο φαγητό (image) Παρακάτω φαίνονται κάποιες εγραφές του πίνακα thermides.

.

| id | category_id | name | quantity | thermides_value | carbs | fat | protein | image |
|---|---|---|---|---|---|---|---|---|
| Filter | Filter | Filter | Filter | Filter | Filter | Filter | Filter | Filter |
| 5 | 2 | Γαλοπούλα | 1 Φέτα | 55 | null | null | null | no_image.jpg |
| 6 | 2 | Γαλοπούλα Καπνιστή | 1 Φέτα του τόστ | 16 | null | null | null | no_image.jpg |
| 7 | 2 | Ζαμπόν διαίτης | 100gr | 150 | null | null | null | no_image.jpg |
| 8 | 2 | Ζαμπόν μετρίου πάχους | 100γρ | 340 | null | null | null | no_image.jpg |
| 10 | 1 | Βυσσινάδα | 1 Ποτήρι | 150 | null | null | null | no_image.jpg |
| 11 | 1 | Γκαζόζα | 1 Ποτήρι | 75 | null | null | null | no_image.jpg |
| 12 | 1 | Κόκα-κόλα | 1 Ποτήρι | 100 | null | null | null | no_image.jpg |
| 13 | 1 | Λεμονάδα | 1 Ποτήρι | 75 | null | null | null | no_image.jpg |
| 14 | 1 | Πορτοκαλάδα κόκκινη (με ανθρακικό) | 1 Ποτήρι | 85 | null | null | null | no_image.jpg |
| 15 | 1 | Σόδα | 1 Ποτήρι | 5 | null | null | null | no_image.jpg |
| 16 | 2 | Λακέρδα | 100gr | 284 | null | null | null | no_image.jpg |
| 17 | 2 | Λουκάνικα Βιέννης βραστά | 1 | 35 | null | null | null | no_image.jpg |
| 18 | 2 | Λουκάνικα Φρανκφούρτης βραστά | 1 | 110 | null | null | null | no_image.jpg |
| 19 | 2 | Λουκάνικα χωριάτικα ψητά | 1 πηρουνιά | 48 | null | null | null | no_image.jpg |
| 20 | 2 | Μορταδέλα | 100gr | 450 | null | null | null | no_image.jpg |
| 22 | 2 | Μπέικον αλατισμένο | 100gr | 740 | null | null | null | no_image.jpg |
| 23 | 2 | Μπέικον καπνιστό | 100gr | 700 | null | null | null | no_image.jpg |
| 24 | 2 | Παριζάκι | 1 φέτα | 105 | null | null | null | no_image.jpg |
| 25 | 2 | Παστουρμάς | 100gr | 350 | null | null | null | no_image.jpg |
| 26 | 2 | Σαλάμι | 100gr | 450 | null | null | null | no_image.jpg |
| 27 | 3 | Βούτυρο | 1 κουταλιά | 70 | null | null | null | no_image.jpg |
| 28 | 3 | Βούτυρο γάλακτος | 100gr | 750 | null | null | null | no_image.jpg |
| 29 | 3 | Βούτυρο φυστικιού | 100gr | 600 | null | null | null | no_image.jpg |
| 30 | 3 | Βούτυρο φυτικό Μαργαρίνη | 100gr | 720 | null | null | null | no_image.jpg |

**Εικόνα 40 - Εγγραφές Πίνακα thermides**

## 5.3 Λεπτομέρειες σχετικά με τον υπολογισμό των θερμίδων

Παρακάτω θα αναλύσω τον τρόπο και την λογική που γίνεται ο υπολογισμός των δεικτών.

### 5.3.1 Ο δείκτης BMR και ο δείκτης δραστηριότητας

Ο δείκτης BMR (Basal Metabolic Rate) είναι η εκτίμηση για το πόσες θερμίδες θα κάψει ο οργανισμός την ημέρα σε περίπτωση που δεν κάνουμε τίποτα (ξεκούραση 24 ώρες). Αντιπροσωπεύει το ελάχιστο ποσό ενέργειας που απαιτείται για να κρατηθεί η σωστή λειτουργία του σώματος, συμπεριλαμβανομένου των σωστών σφυγμών, της σωστής αναπνοής και της σωστής θερμοκρασίας του σώματος. Ο δείκτης αυτός συνδέεται με την ηλικία, το βάρος και το ύψος. Για παράδειγμα αν μειωθεί το βάρος μειώνετε και ο δείκτης BMR, δηλαδή η ποσότητα των θερμίδων που απαιτείται για την διατήρηση του βάρους μας. Αντίστοιχα όταν αυξάνεται η ηλικία μας ο δείκτης αυτός μειώνεται.

Όπως αναφέραμε ο δείκτης BMR αντιπροσωπεύει τις θερμίδες που καίει ο οργανισμός σε κατάσταση ηρεμίας. Επειδή όμως κανείς δεν ξεκουράζεται για 24 ώρες, υπάρχει και η ανάγκη υπολογισμού των θερμίδων που καίγονται από διάφορες δραστηριότητες (όπως γυμναστική, περπάτημα κλπ). Αυτό γίνεται με τον πολλαπλασιασμό του BMR με έναν συντελεστή δραστηριότητας ο οποίος ονομάστηκε δείκτης μέτρησης δραστηριότητας (activity factor) (McArdle et al 1996).

Παρακάτω φαίνεται ο πίνακας που περιγράφει τους δείκτες δραστηριότητες με βάση την άσκηση.

| Δείκτης Δραστηριότητας | Κατηγορία | Περιγραφή |
|---|---|---|
| **1.2** | **Πολύ Μικρή** | Σπάνια έως καθόλου Άσκηση. Καθιστική ζωή - δουλεία γραφείου |
| **1.375** | **Μέτρια** | Άσκηση ή κάποιο άθλημα μέχρι 3 φορές την εβδομάδα |
| **1.55** | **Έντονη** | Άσκηση 3-5 φορές την εβδομάδα |
| **1.725** | **Καθημερινή** | Άσκηση η κάποιο άθλημα κάθε μέρα (6 - 7 μέρες την εβδομάδα) |
| **1.9** | **Αθλητική** | Πολύ σκληρή άσκηση κάθε μέρα και δουλεία που απαιτεί φυσική δραστηριότητα |

Οι παραπάνω παράγοντες αν και θεωρούνται από κάποιους γενικοί και πρόχειροι παράγουν αν πολλαπλασιαστούν με τον δείκτη BMR μια αρκετά ρεαλιστική προσέγγιση για τον αριθμό των ημερήσιων θερμίδων που χρειάζεται κάποιος.

Οι πιο δημοφιλείς εξισώσεις για τον υπολογισμό του δείκτη BMR είναι 2. Η πρώτη και η παλαιότερη είναι του Harris-Benedict η οποία είναι φαίνεται παρακάτω:

**Άνδρες:**

BMR = 66 + [13.7 x βάρος (κιλά)] + [5 x ύψος (εκατοστά)] - [6.76 x ηλικία (χρόνια)]

**Γυναίκες:**

BMR = 655 + [9.6 x βάρος (κιλά)] + [1.8 x ύψος (εκατοστά)] - [4.7 x ηλικία (χρόνια)]

Η μέθοδος αυτή λόγο του ότι είναι η παλαιότερη είναι και η πιο διαδομένη και δημοφιλής μέθοδος.

Η δεύτερη μέθοδος θεωρείται πιο σύγχρονη και πιο ρεαλιστική. Δημιουργήθηκε από τον Mifflin-St Jeor το 1990. Την μέθοδο αυτή χρησιμοποιεί και η εφαρμογή θερμιδομετρητής. Η εξίσωση φαίνεται παρακάτω:

**Άνδρες**:

BMR = [9.99 x βάρος (κιλά)] + [6.25 x ύψος (εκατοστά)] - [4.92 x ηλικία (χρόνια)] + 5

**Γυναίκες**:

BMR = [9.99 x βάρος (κιλά)] + [6.25 x ύψος (εκατοστά)] - [4.92 x ηλικία (χρόνια)] -161

Και οι 2 μέθοδοι έχουν αρκετά μειονεκτήματα και δεν πρέπει να ξεχνάμε ότι κάνουν απλά προβλέψεις και πως δεν πέφτουν πάντα μέσα. Ένα από τα βασικότερα μειονεκτήματα των μεθόδων αυτών είναι ότι δεν υπολογίζουν την ποσότητα λίπους και μυϊκής μάζας που έχει κάποιος. Είναι γνωστό ότι η μυϊκή μάζα αυξάνει τον μεταβολισμό και καίει περισσότερο από ότι το λίπος. Ένας άνθρωπος που έχει παραπάνω από τον φυσιολογικό μυϊκή μάζα θα έχει και μεγαλύτερο BMR από κάποιον με λιγότερη. Οπότε είναι σημαντικό να γίνει κατανοητό ότι οι παραπάνω δείκτες πρέπει να χρησιμοποιούνται καθαρά για λόγους αναφοράς.

### 5.3.2 Ο δείκτης BMI

Ο Δείκτης Μάζας Σώματος BMI (Body Mass Index) δημιουργήθηκε από τον Βέλγο στατιστικολόγο Dolphe Quetelet το 1835 και αποτελεί έναν από τους πιο διαδεδομένους τρόπους για τη μέτρηση της παχυσαρκίας ανδρών και γυναικών. Με αυτόν τον δείκτη μπορούμε να υπολογίσουμε σε τι επίπεδα βρίσκεται το σωματικό μας βάρος (για παράδειγμα φυσιολογικός, παχύσαρκος κλπ). Ο υπολογισμός του δείκτη αυτού γίνεται διαιρώντας το σωματικό βάρος με το τετράγωνο του ύψους. Βασικό μειονέκτημα αυτού του δείκτη είναι ότι δεν υπολογίζει το μυϊκό και σκελετικό σύστημα παρόλα αυτά θεωρείτε ο πιο έγκυρος και αξιόπιστος δείκτης διεθνώς για τον υπολογισμό της μάζας σώματος.

Με βάση το αποτέλεσμα που θα δώσει ο δείκτης υπάρχουν κάποιες κλίμακες για να βρεθεί το αποτέλεσμα. Οι βαθμίδες αυτές φαίνονται παρακάτω.

| BMI | Κλίμακα |
|---|---|
| **< 18.5** | Ελλειποβαρές άτομο |
| **18.5-25** | Φυσιολογικό άτομο |
| **25-30** | Υπέρβαρο άτομο |
| **30-35** | Παχύσαρκο άτομο (Α' βαθμός παχυσαρκίας) |
| **35-40** | Παχύσαρκο άτομο (Β' βαθμός παχυσαρκίας) |
| **>40** | Παχύσαρκο άτομο (Γ' βαθμός παχυσαρκίας) |

Έτσι για παράδειγμα αν κάποιος έχει σωματικό βάρος σε κιλά 75 και ύψος 1.80 τότε ο δείκτης υπολογίζεται έτσι:

75/1.80^2= 23.15 (φυσιολογικός)

# 6. Συμπεράσματα

## 6.1 Μελλοντικές επεκτάσεις / βελτιώσεις

Η εφαρμογή λογισμικού που παρουσιάστηκε βρίσκεται στην πρώτη έκδοση (Θερμιδομετρητής v.1.0). Για τον λόγο αυτό υπάρχουν ακόμα αρκετές βελτιώσεις - επεκτάσεις που μπορούν να γίνουν σε επόμενες εκδόσεις, έτσι ώστε η εφαρμογή να είναι πιο φιλική και πιο λειτουργική για τους χρήστες.

Η πρώτη λειτουργία που πρέπει να μπει στις επόμενες εκδόσεις είναι η δυνατότητα να μπορεί ο χρήστης να προσθέτει τα δικά του φαγητά. Παρόλο που υπάρχουν διαθέσιμα στην βάση αρκετά δημοφιλή φαγητά είναι κατανοητό ότι δεν είναι εφικτό να καλύψει όλους τους χρήστες. Για τον λόγο αυτό ο χρήστης θα μπορεί στο μέλλον να καταχωρεί σε μια φόρμα τα δικά του φαγητά. Επίσης παρόλο που χρησιμοποιήθηκαν κάποια δημοφιλή μοντέλα για την σωστό υπολογισμό των ημερήσιων θερμίδων καταλαβαίνουμε ότι τα αποτελέσματα σε τέτοιου είδους μετρήσεις είναι κατά προσέγγιση και δεν μπορούν να καλύψουν όλες τις ομάδες χρηστών. Για τον λόγο αυτό πρέπει να μπει σε νεότερη έκδοση της εφαρμογής η επιλογή να προσθέτει ο χρήστης χειροκίνητα τις θερμίδες που πρέπει να καταναλώνει μέσα στην ημέρα.

Μια επίσης ιδέα για μελλοντική ενημέρωση, είναι να μπορεί ο χρήστης να προσθέτει τι τύπου σωματική άσκηση έκανε και να αφαιρούνται οι θερμίδες που έκαψε από το σύνολο των ημερήσιων θερμίδων. Γενικότερα θα ήταν καλό να δίνεται η δυνατότητα στον χρήστη να περνάει και άλλα βασικά στοιχεία που προσδιορίζουν μια ισορροπημένη διατροφή (όπως πόσα ποτήρια νερού ανά μέρα κατανάλωσε ο χρήστης).

Η εφαρμογή θα πρέπει επίσης να βρει κάποιο τρόπο να κάνει τον χρήστη να την χρησιμοποιεί πιο συχνά. Αυτό θα μπορούσε να γίνει με την μέθοδο των ειδοποιήσεων (notifications) που είναι διαθέσιμη σε όλα τα κινητά. Έτσι όταν ο χρήστης ξεχάσει να συμπληρώσει το μεσημεριανό του θα βγαίνει ειδοποίηση στο κινητό για να του το υπενθυμίζει να το συμπληρώσει. Αντίστοιχα θα μπορούσαν να δημιουργηθούν διάφορα άλλα hints που θα ενισχύσουν την χρησιμότητα της εφαρμογής. Κάποια παραδείγματα είναι καταμέτρηση ποτηριών νερού (και ειδοποίηση στο τέλος της ημέρας αν δεν έχει πιει το προβλεπόμενο αριθμό), καταγραφή δραστηριοτήτων - γυμναστικής και τύπου άσκησης (και προσθήκη των θερμίδων που έχουν καταναλωθεί από την άσκηση στον ημερήσιο αριθμό της εφαρμογής).

Ένα ακόμα χρήσιμο στοιχείο που θα μπορούσε να προστεθεί μελλοντικά, είναι η αυτόματη ειδοποίηση για ενημέρωση του βάρους. Έτσι στην περίπτωση που κάποιος έχει καταναλώσει λιγότερες θερμίδες από αυτές που χρειάζεται να βγαίνει ένα μήνυμα που θα τον ρωτάει αν έχει χάσει x κιλά και αν θέλει να ενημερώσει το σωματικό του βάρος. Για παράδειγμα αν κάποιος έχει καταναλώσει 3500 θερμίδες λιγότερες από αυτές που είχαν υπολογιστεί (με την βοήθεια του δείκτη BMR) που ισοδυναμούν με μείον ένα κιλό, να βγαίνει μήνυμα που θα ζητάει από τον χρήστη να ζυγιστεί και σε περίπτωση που έχει χάσει να ενημερώσει το βάρος του. Αυτόματα μετά η εφαρμογή θα υπολογίζει τον νέο δείκτη BMR.

Επειδή η εφαρμογή είναι διαθέσιμη μόνο στην Ελληνική γλώσσα μια ακόμα επέκταση της εφαρμογής θα μπορούσε να είναι η υποστήριξη ενός πολυγλωσσικού περιβάλλοντος. Τέλος η επέκταση της εφαρμογής και σε άλλες πλατφόρμες με διαφορετικό λειτουργικό σύστημα (Windows, iOS) και η ασφάλεια της εφαρμογής είναι μερικοί από τους μελλοντικούς στόχους.

## 6.2 Σύνοψη και συμπεράσματα

Η κινητή τηλεφωνία έχει δυνατότητες να προσφέρει πολλά στον κλάδο της υγείας και της διατροφής ιδιαίτερα στις αναπτυσσόμενες χώρες. Ο σύγχρονος τρόπος ζωής, η καθιστική ζωή και οι γρήγοροι ρυθμοί μας έχουν κάνει να καταναλώνουμε τροφές που είναι γρήγορες στην παρασκευή (fast foods) αλλά εξαιρετικά επικίνδυνες για την υγεία. Η καταμέτρηση των θερμίδων με την βοήθεια του κινητού, μια συσκευή που σήμερα ο καθένας έχει πάνω του ανά πάσα στιγμή, είναι απαραίτητη για την βελτίωση της ποιότητας της διατροφής και για έλεγχο της ημερήσιας κατανάλωσης θερμίδων

Σήμερα υπάρχει μια περίσσεια εφαρμογών για θερμίδες τόσο σε Ελλάδα όσο και στο εξωτερικό, παρόλα αυτά η ποιότητα των εφαρμογών στην Ελλάδα είναι ακόμα ελλιπής και χωρίς κίνητρο. Η εφαρμογή θερμιδομετρητής προσπαθεί να καλύψει αυτό το κενό που υπάρχει στον χώρο της διατροφής.

# Βιβλιογραφία

CaloriesPerHour.com. *Diet and Weight Loss Tutorial.*

Heeks, R., A Jagun. *Mobile phones and Commonwealth development.* London, 2008.

Inc, Google. *SDK Tools | Android Developers.* http://developer.android.com.

Müller B, Merk S, Bürgi U, Diem P. «Calculating the basal metabolic rate and severe and morbid obesity.» *Praxis .*

Patrick K., Griswold WG., Raab F., Intille SS. «Health and the mobile phone.» *American Journal of Preventive Medicine*, 2008.

Roza, Allan M, και Harry M Shizgal. «The Harris Benedict equation reevaluated: resting energy requirements and the body cell mass.» *The American Journal of Clinical Nutrition*, 1984.

Venuto, Tom. *Burn the Fat, Feed the Muscle.* 2003.

wikipedia.org. *Basal metabolic rate.*

Αγγελίδης, Π. *Ιατρική Πληροφορική.* 2011.

Michael J. Ackerman, Rosemarie Filart, Lawrence P. Burgess, Insup Lee, and Ronald K. Poropatich. "Developing next-generation telehealth tools and technologies: patients, systems, and data perspectives." *Telemedicine Journal and e-Health.*

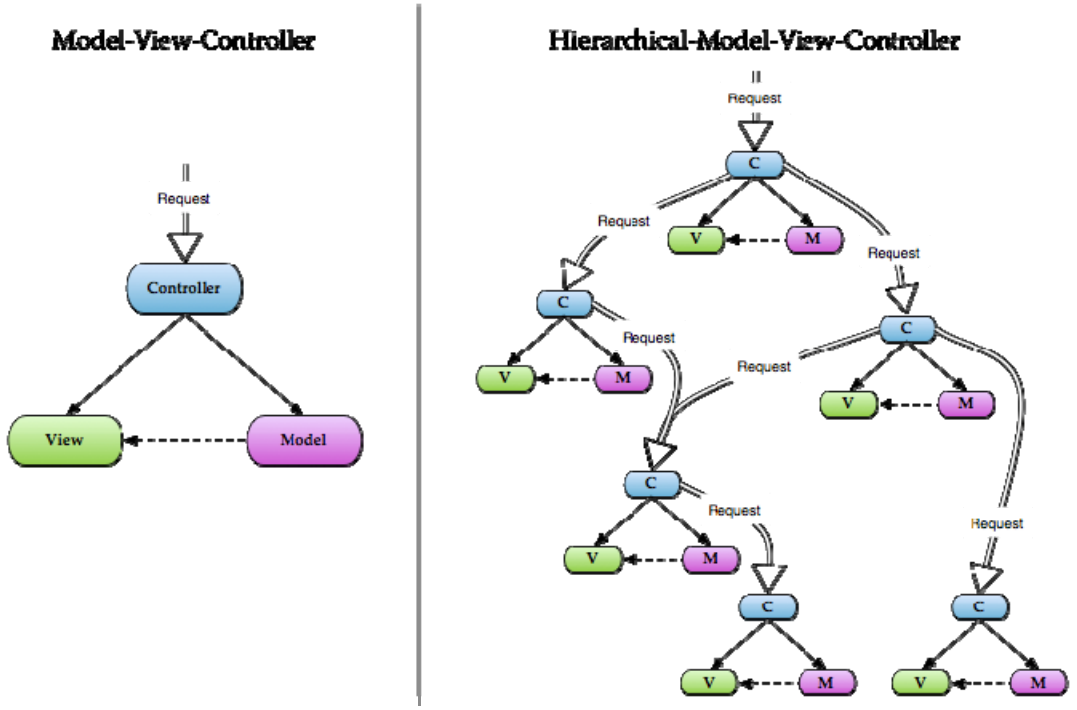
## Παράρτημα Α – Πίνακας Ακρωνυμίων/Συντομογραφιών

| Συντομογραφία | Εξήγηση |
|---|---|
| **HMVC** | Hierarchical model–view–controller |
| **MVC** | Model View Controller |
| **SDK** | Software development kit |
| **IDE** | Integrated development environment |
| **BMR** | Basal Metabolic Rate |
| **BMI** | Body mass index |
| **API** | Application programming interface |

# Παράρτημα Β

## Τμήματα κώδικα απομακρυσμένης εφαρμογής ( PHP 5.4)

Η απομακρυσμένη εφαρμογή ακολουθεί την λογική του **Hierarchical model–view–controller (HMVC)** η οποία είναι μια επέκταση του κλασικού σχεδιασμού MVC**.** Με λίγα λόγια αυτό σημαίνει ότι η εφαρμογή χωρίζεται σε ακόμα περισσότερα κομμάτια (blocks) έτσι ώστε να υπάρχει μεγαλύτερη επεκτασιμότητα. Στην παρακάτω εικόνα φαίνεται η διαφορά ανάμεσα στα 2 μοντέλα.




**Εικόνα 41 - MVC- HMVC**

Παρακάτω θα δείξω κάποια από τα βασικά κομμάτια της εφαρμογής που ακολουθούν την αυτή την λογική. Όλη η εφαρμογή έχει χωριστεί σε modules τα οποία είναι στον φάκελο service/admin/modules. Κάθε ένα από τα modules αυτά έχει απαραίτητα έναν controller και ένα view (model έχουν μόνο οι βασικές ενότητες). Τα modules στο κομμάτι του διαχειριστή είναι:

1) Module Auth. Αυτό το module είναι υπεύθυνο για το authentication του χρήστη όπως επίσης και για τον έλεγχο πρόσβασης στις ενότητες βάση ρόλου.

2) Module Home. Η αρχική σελίδα της εφαρμογής και οθόνη που εμφανίζεται αμέσως μετά την σύνδεση.

3) Module Thermides. Το module αυτό χειρίζεται όλο το τμήμα της διαχείρισης θερμίδων. Εμφάνιση λίστας - δημιουργία - επεξεργασία - διαγραφή φαγητού.

4) Module Categories. Αντίστοιχα αυτό το κομμάτι είναι υπεύθυνο για τις κατηγορίες.

5) Module Settings. Το τμήμα αυτό είναι υπεύθυνο για τις ρυθμίσεις του συστήματος που μπορεί να κάνει ο διαχειριστής.

Στην εικόνα πιο κάτω βλέπουμε την κλάση Home η οποία είναι ο controller για την αρχική σελίδα (μετά την σύνδεση).

```php
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class Home extends CI_Controller {


    function __construct()
    {
        parent::__construct();

        // check if user logged in
        if (!$this->ion_auth->logged_in())
        {
            redirect('auth/login');
        }
        $this->load->library('form_validation');
        //$this->security->csrf_verify();
        $this->load->model('home_model');

    }

    function index()
    {

        $data['success_message'] = $this->session->flashdata('success_message');

        $meta['page_title'] = $this->lang->line("welcome")." ".SITE_NAME."!";
        $data['page_title'] = $this->lang->line("welcome")." ".SITE_NAME."!";
        $this->load->view('commons/header', $meta);
        $this->load->view('content', $data);
        $this->load->view('commons/footer');

    }

}
```

**Εικόνα 42- Controller Home**

Όπως φαίνεται παραπάνω γίνεται extend του βασικού controller του code igniter έτσι ώστε να μπορούν να εκτελεστούν οι βασικές μέθοδοι του. Στον constructor το πρώτο πράγμα που γίνεται (σε κάθε controller) είναι ο έλεγχος αν ο χρήστης έχει συνδεθεί, αν όχι τον κάνει απευθείας redirect στη οθόνη σύνδεσης. Μετά φορτώνεται το form validator και το μοντέλο.

Η **index** είναι η βασική function που τρέχει αμέσως μετά τον constructor. Εδώ δημιουργούνται πίνακες με δεδομένα ($meta, $data) τα οποία περνιούνται αμέσως μετά στα views με την κλήση $this->load->view ( 'template', data).

Στην ακόλουθη εικόνα φαίνεται πως αυτά τα δεδομένα μπορούν να χρησιμοποιηθούν στο view.

```php
<?php
if($success_message) {
 echo "<div class=\"alert alert-success\"><button type=\"button\" class=\"close\" data-dismiss=\"alert\">&times;</button>" ;
 echo $success_message . "</div>";
 } ?>

<h3 class="title"><?php echo $page_title; ?></h3>
```

**Εικόνα 43 - Content view**

Εδώ βλέπουμε πως αν έχει οριστεί η μεταβλητή success_message (στην περίπτωση που ερχόμαστε από άλλη σελίδα και θέλουμε να δείξουμε ένα μήνυμα επιτυχίας) τότε εκτυπώνει στην σελίδα το μήνυμα. Επίσης η μεταβλητή $page_title είναι το string που περάστηκε από τον controller στο view και στην εδώ αντιπροσωπεύει την ονομασία της σελίδας που βρισκόμαστε.

## Ρόλοι χρηστών:

Σε κάθε view γίνεται έλεγχος για το αν ο συνδεδεμένος χρήστης έχει δυνατότητα να δει κάποια κομμάτια της σελίδας.

```
<?php if ($this->ion_auth->in_group('owner')) { ?>
<li>
    <a class="tip" href="index.php?module=settings&view=system_setting" title="<?php echo $this->lang->line("settings"); ?>">
        <i><img src="<?php echo $this->config->base_url(); ?>assets/img/icons/settings.png" alt="" /></i>
        <span><span><?php echo $this->lang->line("settings"); ?></span></span>
    </a>
</li>
<li>
    <a class="tip" href="index.php?module=auth&view=users" title="<?php echo $this->lang->line("users"); ?>">
        <i><img src="<?php echo $this->config->base_url(); ?>assets/img/icons/users.png" alt="" /></i>
        <span><span><?php echo $this->lang->line("users"); ?></span></span>
    </a>
</li>
<li>
    <a class="tip" href="index.php?module=auth&view=create_user" title="<?php echo $this->lang->line("new_user"); ?>">
        <i><img src="<?php echo $this->config->base_url(); ?>assets/img/icons/user_add.png" alt="" /></i>
        <span><span><?php echo $this->lang->line("new_user"); ?></span></span>
    </a>
</li>
<?php } ?>
<li>
    <a class="tip" href="index.php?module=auth&view=change_password" title="<?php echo $this->lang->line("change_password"); ?>">
        <i><img src="<?php echo $this->config->base_url(); ?>assets/img/icons/user_edit.png" alt="" /></i>
        <span><span><?php echo $this->lang->line("change_password"); ?></span></span>
    </a>
</li>
```

**Εικόνα 44 - Έλεγχος ρόλου χρήστη**

Για να ελέγξουμε τον ρόλο του χρήστη χρησιμοποιούμε το library ion auth. Εδώ φαίνεται ο έλεγχος που γίνεται στο dashboard της αρχικής σελίδας. Το παραπάνω τμήμα κώδικα με λίγα λόγια μας λέει ότι μόνο αν ο χρήστης είναι ο ιδιοκτήτης εκτύπωσε τις παρακάτω επιλογές: Ρυθμίσεις, λίστα των χρηστών και δημιουργία. Η τελευταία ενότητα είναι έξω από τον έλεγχο οπότε είναι διαθέσιμη για όλους τους χρήστες.

Ο έλεγχος αυτός θα πρέπει να γίνεται και μέσα στους controllers, έτσι ώστε να αποφευχθεί η περίπτωση να έχει πρόσβαση κάποιος που γνωρίζει τον σύνδεσμο. Παρακάτω φαίνεται ο έλεγχος που γίνεται στον controller της σελίδας των χρηστών :

```
function users() {
    $groups = array('admin');
    if ($this->ion_auth->in_group($groups))
    {
        $this->session->set_flashdata('message', $this->lang->line("access_denied"));
        $data['message'] = (validation_errors() ? validation_errors() : $this->session->flashdata('message'));
        redirect('module=home', 'refresh');
    }

    $this->user_check();

    $data['message'] = (validation_errors()) ? validation_errors() : $this->session->flashdata('message');
    $data['success_message'] = $this->session->flashdata('success_message');

        //list the users
        $data['users'] = $this->ion_auth->users()->result();
        foreach ($data['users'] as $k => $user)
        {
            $data['users'][$k]->groups = $this->ion_auth->get_users_groups($user->id)->result();
        }

        $meta['page_title'] = 'Users';
        $this->load->view('commons/header', $meta);

        $this->load->view('index', $data);

        $this->load->view('commons/footer');

}
```

**Εικόνα 45 - Έλεγχος ρόλου στον controller**

Στην αρχή του controller users βλέπουμε ότι δημιουργείται ένα array με τα group που δεν επιτρέπεται να έχουν πρόσβαση (στην συγκεκριμένη περίπτωση μόνο ο admin). Χρησιμοποιώντας την βιβλιοθήκη ion_auth ελέγχουμε αν ο χρήστης είναι σε αυτό το group. Στην περίπτωση που είναι στην ομάδα που δεν επιτρέπεται η πρόσβαση γίνεται redirect στην αρχική σελίδα και περνιέται το μήνυμα ότι απαγορεύεται η πρόσβαση. Αν ο χρήστης είναι στην ομάδα των χρηστών που επιτρέπετε η πρόσβαση συνεχίζεται η εκτέλεση του κώδικα.

**Validation Φορμών:**

Πριν περαστούν τα δεδομένα στην βάση δεδομένων της εφαρμογής πρέπει να τηρούνται κάποια κριτήρια. Για τον λόγο αυτό έχει χρησιμοποιηθεί η βιβλιοθήκη form validation όπου μας βοηθάει στο server side validation. Παρακάτω φαίνεται ο έλεγχος που γίνεται στην προσθήκη χρήστη (αντίστοιχος με αυτόν που γίνεται στην επεξεργασία).

```
$this->form_validation->set_message('is_natural_no_zero', 'The %s field is required.');
//validate form input
$this->form_validation->set_rules('first_name', $this->lang->line("first_name"), 'required|xss_clean');
$this->form_validation->set_rules('last_name', $this->lang->line("last_name"), 'required|xss_clean');
$this->form_validation->set_rules('role', $this->lang->line("user_role"), 'is_natural_no_zero|required|xss_clean');
$this->form_validation->set_rules('email', $this->lang->line("email_address"), 'required|valid_email');
$this->form_validation->set_rules('phone', $this->lang->line("phone"), 'required|xss_clean|min_length[9]|max_length[16]');
$this->form_validation->set_rules('company', $this->lang->line("company"), 'required|xss_clean');
$this->form_validation->set_rules('password', $this->lang->line("pw"), 'required|min_length[' . $this->config->item('min_password_length', 'ion_auth') . ']|max_length[' .
$this->config->item('max_password_length', 'ion_auth') . ']|matches[password_confirm]');
$this->form_validation->set_rules('password_confirm', $this->lang->line("confirm_pw"), 'required');

if ($this->form_validation->run() == true)
{

    $username = strtolower($this->input->post('first_name')) . ' ' . strtolower($this->input->post('last_name'));
    $email = $this->input->post('email');
    $password = $this->input->post('password');

    $additional_data = array('first_name' => $this->input->post('first_name'),
        'last_name' => $this->input->post('last_name'),
        'company' => $this->input->post('company'),
        'phone' => $this->input->post('phone'),
    );

    $group = array($this->input->post('role'));

}


if ($this->form_validation->run() == true && $this->ion_auth->register($username, $password, $email, $additional_data, $group))
{ //check to see if we are creating the user
    //redirect them back to the admin page
    $this->session->set_flashdata('success_message', $this->lang->line("user_added"));
    redirect("module=auth&view=users", 'refresh');
}
```

**Εικόνα 46- Validation Χρήστη**

Για να ορίσουμε τους κανόνες χρησιμοποιούμε την μέθοδο set_rules η οποία δέχεται σαν ορίσματα το όνομα του πεδίου, τη περιγραφή του πεδίου (η οποία μπαίνει στο μήνυμα λάθους) και σαν τρίτο όρισμα τους κανόνες. Τέλος αν ο validator μας γυρίσει true και γίνει επιτυχώς το register (που παρέχετε από το library ion auth) του χρήστη γίνεται ανακατεύθυνση στην λίστα χρηστών και περνιέται ένα μήνυμα επιτυχίας.

Παρακάτω βλέπουμε το view της φόρμας προσθήκης χρήστη (create_user.php).

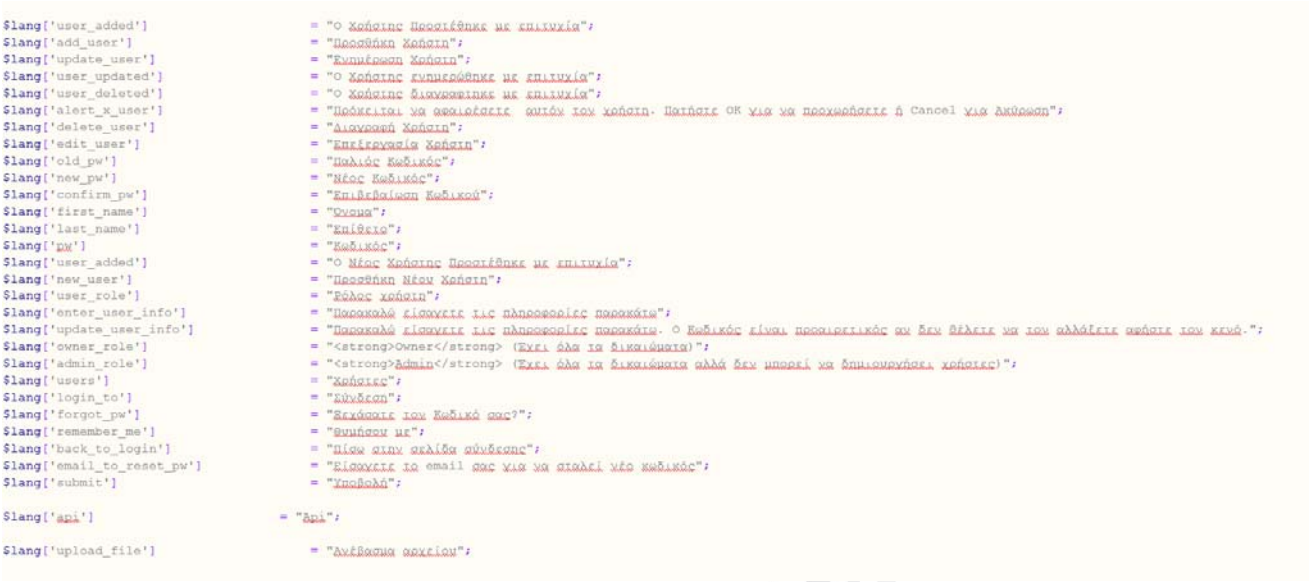
```
<?php if($message) { echo "<div class=\"alert alert-error\"><button type=\"button\" class=\"close\" data-dismiss=\"alert\">&times;</button>" . $message . "</div>"; } ?>

<h3 class="title"><?php echo $this->lang->line("create_user"); ?></h3>
<p><?php echo $this->lang->line("enter_user_info"); ?></p>
<?php $attrib = array('class' => 'form-horizontal'); echo form_open("module=auth&view=create_user", $attrib);?>
<div class="control-group">
  <label class="control-label" for="first_name"><?php echo $this->lang->line("first_name"); ?></label>
  <div class="controls"> <?php echo form_input($first_name, '', 'class="span4" pattern=".{2,55}" required="required" data-error="'.$this->lang->line("first_name").' '.$this->lang->line("is_required").'"');?> </div>
</div>
<div class="control-group">
  <label class="control-label" for="last_name"><?php echo $this->lang->line("last_name"); ?></label>
  <div class="controls"> <?php echo form_input($last_name, '', 'class="span4" pattern=".{2,55}" required="required" data-error="'.$this->lang->line("last_name").' '.$this->lang->line("is_required").'"');?> </div>
</div>
<div class="control-group">
  <label class="control-label" for="company"><?php echo $this->lang->line("company"); ?></label>
  <div class="controls"> <?php echo form_input($company, '', 'class="span4" required="required" data-error="'.$this->lang->line("company").' '.$this->lang->line("is_required").'"');?> </div>
</div>
<div class="control-group">
  <label class="control-label" for="phone"><?php echo $this->lang->line("phone"); ?></label>
  <div class="controls"> <?php /* echo form_input($phone, '', 'class="span4" required="required" data-error="'.$this->lang->line("phone").' '.$this->lang->line("is_required").'"'); */?>
  <input type="tel" name="phone" class="span4" pattern="[0-9]{7,15}" required="required" data-error="<?php echo $this->lang->line("phone").' '.$this->lang->line("is_required"); ?>" /></div>
</div>
<div class="control-group">
  <label class="control-label" for="email"><?php echo $this->lang->line("email_address"); ?></label>
  <div class="controls"> <?php /* echo form_input($email, '', 'class="span4" required="required" data-error="'.$this->lang->line("date").' '.$this->lang->line("is_required").'"'); */?>
  <input type="email" name="email" class="span4" required="required" data-error="<?php echo $this->lang->line("email_address").' '.$this->lang->line("is_required"); ?>" /></div>
</div>
<div class="control-group">
  <label class="control-label" for="role"><?php echo $this->lang->line("user_role"); ?></label>
  <div class="controls">
    <div class="btn-group user-role" data-toggle="buttons-radio">
   <button type="button" value="1" class="btn"><?php echo $this->lang->line("owner"); ?></button>
   <button type="button" value="2" class="btn"><?php echo $this->lang->line("admin"); ?></button>
   </div>
   <input type="hidden" name="role" id="role" value="<?php /* echo $group->group_id; */ ?>">
  </div>
</div>
<div class="control-group">
  <label class="control-label" for="password"><?php echo $this->lang->line("pw"); ?></label>
  <div class="controls"> <?php echo form_input($password , '', 'class="password span4" id="password" pattern=".{8,55}" required="required" data-error="'.$this->lang->line("pw").' '.$this->lang->line("is_required").'"');?> </div>
</div>
<div class="control-group">
  <label class="control-label" for="confirm_pw"><?php echo $this->lang->line("confirm_pw"); ?></label>
  <div class="controls"> <?php echo form_input($password_confirm , '', 'class="password span4" id="confirm_pw" pattern=".{8,55}" required="required" data-error="'.$this->lang->line("confirm_pw").' '.$this->lang->line("is_required").'"');?> </div>
</div>
<div class="control-group">
  <div class="controls"> <?php echo form_submit('submit', $this->lang->line("add_user"), 'class="btn btn-primary"');?> </div>
</div>
<?php echo form_close();?>
```

**Εικόνα 47 - View create_user.php**

Στο view αυτό εμφανίζεται η φόρμα. Αξίζει να προσέξουμε ότι η φόρμα για ασφάλεια ανοίγει και κλείνει με τις συναρτήσεις form_open και form_close αντίστοιχα. Κάθε πεδίο έχει ένα ξεχωριστό name (το οποίο χρειάζεται στο validation που κάνει ο controller) . Τέλος το χαρακτηριστικό pattern (που το βρίσκουμε σε κάποια πεδία) είναι το εύρος των ψηφίων όταν είναι σε αγκύλες ({..}) και οι χαρακτήρες που μπορεί να δεχτεί το πεδίο όταν είναι σε παρένθεση ([..]). Για παράδειγμα το pattern="[0-9]{7,15}" που βλέπουμε στο πεδίο phone σημαίνει ότι τα ψηφία πρέπει να είναι αριθμός μόνο και το εύρος πρέπει να είναι από 7 εώς 15 ψηφία το μέγιστο.

**Πολυγλωσσία:**

Για να είναι εφικτό στο μέλλον να προστεθούν και άλλες γλώσσες όλες οι ετικέτες των views και τα μηνύματα βρίσκονται σε ξεχωριστό αρχείο (language/greek/ service_lang.php). Ο πίνακας $lang κρατάει όλα τα κλειδιά που χρησιμοποιούνται. Έτσι αν στο μέλλον χρειαστεί να μεταφραστεί η σελίδα, για παράδειγμα στα αγγλικά, αρκεί να δημιουργήσουμε έναν φάκελο english μέσα και να βάλουμε το service_lang.php με αγγλικές τίμες.



```
$lang['user_added']                 = "Ο Χρήστης Προστέθηκε με επιτυχία";
$lang['add_user']                   = "Προσθήκη Χρήστη";
$lang['update_user']                = "Ενημέρωση Χρήστη";
$lang['user_updated']               = "Ο Χρήστης ενημερώθηκε με επιτυχία";
$lang['user_deleted']               = "Ο Χρήστης διαγραφτηκε με επιτυχία";
$lang['alert_x_user']               = "Πρόκειται να αφαιρέσετε  αυτόν τον χρήστη. Πατήστε OK για να προχωρήσετε ή Cancel για Ακύρωση";
$lang['delete_user']                = "Διαγραφή Χρήστη";
$lang['edit_user']                  = "Επεξεργασία Χρήστη";
$lang['old_pw']                     = "Παλιός Κωδικός";
$lang['new_pw']                     = "Νέος Κωδικός";
$lang['confirm_pw']                 = "Επιβεβαίωση Κωδικού";
$lang['first_name']                 = "Όνομα";
$lang['last_name']                  = "Επίθετο";
$lang['pw']                         = "Κωδικός";
$lang['user_added']                 = "Ο Νέος Χρήστης Προστέθηκε με επιτυχία";
$lang['new_user']                   = "Προσθήκη Νέου Χρήστη";
$lang['user_role']                  = "Ρόλος χρήστη";
$lang['enter_user_info']            = "Παρακαλώ είσαγετε τις πληροφορίες παρακάτω";
$lang['update_user_info']           = "Παρακαλώ είσαγετε τις πληροφορίες παρακάτω. Ο Κωδικός είναι προαιρετικός αν δεν θέλετε να τον αλλάξετε αφήστε τον κενό.";
$lang['owner_role']                 = "<strong>Owner</strong> (Έχει όλα τα δικαιώματα)";
$lang['admin_role']                 = "<strong>Admin</strong> (Έχει όλα τα δικαιώματα αλλά δεν μπορεί να δημιουργήσει χρήστες)";
$lang['users']                      = "Χρήστες";
$lang['login_to']                   = "Σύνδεση";
$lang['forgot_pw']                  = "Ξεχάσατε τον Κωδικό σας?";
$lang['remember_me']                = "Θυμήσου με";
$lang['back_to_login']              = "Πίσω στην σελίδα σύνδεσης";
$lang['email_to_reset_pw']          = "Είσαγετε το email σας για να σταλεί νέο κωδικός";
$lang['submit']                     = "Υποβολή";

$lang['api']                    = "Api";

$lang['upload_file']                = "Ανέβασμα αρχείου";
```

**Εικόνα 48 - Αρχείο γλώσσας**

### Ανέβασμα φωτογραφίας στον server

Οι διαχειριστές μπορούν να ανεβάσουν φωτογραφίες ανά φαγητό και ανά κατηγορία. Παρακάτω δίνεται η διαδικασία που γίνεται για να ανέβει η φωτογραφία στον εξυπηρετητή. Για την χρήση αυτής της λειτουργίας χρησιμοποιήθηκε το library upload_photo.

Στην αρχή γίνεται έλεγχος αν υπάρχει αρχείο (if($_FILES['userfile']['size'] > 0)) αν δεν υπάρχει περνιέται κενή φωτογραφία στην βάση. Όταν υπάρχει διαθέσιμο αρχείο γίνεται φόρτωση της βιβλιοθήκης και δίνονται κάποιες βασικές τιμές:

- $config['upload_path'] = 'assets/uploads/';  : Το μονοπάτι που θα αποθηκευτεί η φωτογραφία.
- $config['allowed_types'] = 'gif|jpg|png'; Οι τύποι αρχείων που επιτρέπονται.
- $config['max_size'] = '1000';  Το μέγιστο μέγεθος (σε kilobytes).
- $config['max_width'] = '1000'; Το μέγιστο πλάτος.
- $config['max_height'] = '1000'; Το μέγιστο ύψος.
- $config['overwrite'] = FALSE; Αν υπάρχει το αρχείο να γίνει overwite (true/ false).

Αφού οριστούν οι παραπάνω παράμετροι γίνεται αρχικοποίηση της βιβλιοθήκης και ξεκινάει η διαδικασία ανεβάσματος. Στην μεταβλητή $photo κρατιέται το όνομα του αρχείου για να χρησιμοποιηθεί πιο κάτω για εισαγωγή στην βάση δεδομένων.

```
if($_FILES['userfile']['size'] > 0){

$this->load->library('upload_photo');

$config['upload_path'] = 'assets/uploads/';
$config['allowed_types'] = 'gif|jpg|png';
$config['max_size'] = '1000';
$config['max_width'] = '1000';
$config['max_height'] = '1000';
$config['overwrite'] = FALSE;

    $this->upload_photo->initialize($config);

    if( ! $this->upload_photo->do_upload()){

        $error = $this->upload_photo->display_errors();
        $this->session->set_flashdata('message', $error);
        redirect("module=thermides&view=add", 'refresh');
    }

$photo = $this->upload_photo->file_name;

$this->load->helper('file');
$this->load->library('image_lib');
$config['image_library'] = 'gd2';
$config['source_image'] = 'assets/uploads/'.$photo;
$config['new_image'] = 'assets/uploads/thumbs/'.$photo;
$config['maintain_ratio'] = TRUE;
$config['width'] = 76;
$config['height'] = 76;

$this->image_lib->clear();
$this->image_lib->initialize($config);

if ( ! $this->image_lib->resize())
{
    echo $this->image_lib->display_errors();

}

} else {
    $photo = NULL;
}
```

**Εισαγωγή στην βάση:**

Η εισαγωγή στην βάση όλων των στοιχειών των φορμών γίνεται με κλήση στο μοντέλο από τον controller. Όπως φαίνεται παρακάτω, για την εισαγωγή φαγητού (και αφού έχουμε περάσει όλα τα validators) ο controller κάνει κλήση στην συνάρτηση addThermida().

```
if ( $this->form_validation->run() == true && $this->thermides_model->addThermida( $name, $photo, $data))
{
    $this->session->set_flashdata('success_message', $this->lang->line("thermida_added"));
    redirect('module=thermides', 'refresh');
}
```

**Εικόνα 49- Κλήση controller στο model για εισαγωγή φαγητού**

Αν δεν υπάρξει κάποιο σφάλμα και γυρίσει true τότε γίνεται ανακατεύθυνση στην σελίδα με την λίστα των θερμίδων (περνώντας μήνυμα επιτυχίας) . Το μοντέλο addThermida φαίνεται παρακάτω και βρίσκεται στο module thermides και στον φάκελο views/thermides_model.php :

```
public function addThermida( $name,  $photo, $data = array())
{

    if($photo == NULL) {
        // Φαγητό χωρίς εικόνα
        $thermidesData = array(
            'name'                  => $data['name'],
            'category_id'           => $data['category_id'],
            'quantity'              => $data['quantity'],
            'thermides_value'       => $data['thermides_value'],
            'details'               => $data['details']
        );
    } else {
        // Φαγητό με εικόνα
        $thermidesData = array(
            'name'                  => $data['name'],
            'category_id'           => $data['category_id'],
            'quantity'              => $data['quantity'],
            'thermides_value'       => $data['thermides_value'],
            'details'               => $data['details'],
            'image'                 => $photo
        );
    }

    if($this->db->insert('thermides', $thermidesData)) {
        return true;
    } else {
        return false;
    }
}
```

**Εικόνα 50- Μοντέλο για εισαγωγή φαγητού**

Όπως ξεκάθαρα φαίνεται αν υπάρχει εικόνα περνιέται ένα array με το όνομα της εικόνας ενώ αλλιώς περνιούνται μόνο τα υπόλοιπα στοιχεία του φαγητού. Η εισαγωγή γίνεται με την κλήση $this->db->insert('thermides', $thermidesData) με ορίσματα το όνομα του πίνακα και ένα array με μορφή κλειδί->τιμή. Το κλειδί πρέπει να ταυτίζεται με το όνομα του πεδίου που θέλουμε να γίνει insert.

### Επεξεργασία βάσης δεδομένων

Η επεξεργασία της βάσης δεδομένων έχει ακριβώς την ίδια λογική, μόνο που εδώ γίνεται update αντί για insert οπότε μας χρειάζεται μια επιπλέον πληροφορία (το id ). Βλέπουμε παρακάτω ένα παράδειγμα για ενημέρωση του φαγητού.

```
if ( $this->form_validation->run() == true && $this->thermides_model->updateThermida($id, $photo, $data))
{
    $this->session->set_flashdata('success_message', $this->lang->line("thermides_updated"));
    redirect("module=thermides", 'refresh');
}
```

**Εικόνα 51 - Controller ενημέρωση φαγητού**

```
public function updateThermida($id, $photo, $data = array())
{


    if($photo == NULL) {
        // Φαγητό χωρίς εικόνα
        $thermidesData = array(
            'name'                  => $data['name'],
            'category_id'           => $data['category_id'],
            'quantity'              => $data['quantity'],
            'thermides_value'       => $data['thermides_value'],
            'details'               => $data['details']
        );
    } else {
        // Φαγητό με εικόνα
        $thermidesData = array(
            'name'                  => $data['name'],
            'category_id'           => $data['category_id'],
            'quantity'              => $data['quantity'],
            'thermides_value'       => $data['thermides_value'],
            'details'               => $data['details'],
            'image'                 => $photo
        );
    }


    $this->db->where('id', $id);
    if($this->db->update('thermides', $thermidesData)) {
        return true;
    } else {
        return false;
    }
}
```

**Εικόνα 52 - Μοντέλο για ενημέρωση φαγητού**

### Διαγραφή βάσης δεδομένων

Και στην διαγραφή μιας εγγραφής, όπως είναι λογικό, μας χρειάζεται η επιπλέον πληροφορία του id. Παρακάτω φαίνεται η διαδικασία διαγραφής ενός φαγητού.

```
if ( $this->thermides_model->deleteThermida($id) )
{
    $this->session->set_flashdata('success_message', $this->lang->line("thermida_deleted"));
    redirect('module=thermides', 'refresh');
}
```

**Εικόνα 53 - Controller διαγραφή φαγητού**

```
public function deleteThermida($id)
{
    if($this->db->delete('thermides', array('id' => $id)) ) {
        return true;
    }
return FALSE;
}
```

**Εικόνα 54 - Μοντέλο για διαγραφή φαγητού**

# Παράρτημα Γ

**Τμήματα κώδικα mobile εφαρμογής ( JAVA, XML)**

Παρακάτω θα αναλύσουμε κάποια βασικά σημεία του κώδικα της mobile εφαρμογής. Το περιβάλλον (IDE) που έγινε η ανάπτυξη είναι το eclipse.

**Android Manifest**

Σε κάθε android εφαρμογή απαιτείται να υπάρχει ένα αρχείο με το όνομα AndroidManifest.xml στον κορμό του φάκελου με το project. Αυτό το αρχείο είναι το πιο βασικό αρχείο που χρειάζεται κάθε εφαρμογή για να ξεκινήσει καθώς ονομάζει και αναγνωρίζει τα java πακέτα τις εφαρμογής μέσα στα οποία βρίσκονται οι java κλάσεις. Το αρχείο αυτό, όπως δηλώνει και η επέκταση του, είναι γραμμένο στην γλώσσα XML και περιέχει όλα τα απαραίτητα δικαιώματα που χρειάζεται η εφαρμογή (πχ ιντερνέτ, πρόσβαση στην κάμερα κλπ), δηλώνει το ελάχιστο (και το μέγιστο) Android API που χρειάζεται η εφαρμογή για να τρέξει και τέλος δηλώνει όλα τα activities, services, broadcast receivers που η εφαρμογή έχει. Παρακάτω φαίνεται το android manifest της εφαρμογής θερμιδομετρητής.

Τα βασικά σημεία που πρέπει να προσέξουμε σε αυτό το αρχείο είναι τα παρακάτω

- ✓ Στην γλώσσα XML όλα τα tags πρέπει να κλείνουν. Σε αυτό το αρχείο είναι υποχρεωτικά τα tags <manifest>,<application> και <uses-permission>.
- ✓ Στο <manifest> tag πρέπει να δηλωθούν υποχρεωτικά το πακέτο της εφαρμογής (συνήθως ένα ανεστραμμένο url της εταιρίας που φτιάχνει την εφαρμογή το οποίο δηλώνει και την ιεραρχία των φακέλων που βρίσκεται ο κώδικας JAVA) και το android version code και version name (η τρέχουσα έκδοση της εφαρμογής).
- ✓ Πρέπει να δηλωθεί με το tag <uses-sdk> η ελάχιστη έκδοση SDK που πρέπει να τρέξει η εφαρμογή καθώς και η έκδοση που στοχεύουμε. Όπως φαίνεται στο manifest ο θερμιδομετρητής στοχεύει στην έκδοση 21 ενώ η ελάχιστη έκδοση που μπορεί να τρέξει είναι η 8 (Android 2.2).
- ✓ Αν η εφαρμογή κάνει χρήση κάποιας λειτουργίας του λειτουργικού πρέπει να δηλωθεί το αντίστοιχο permission (με το tag <uses-permission>.
- ✓ Στο application tag δηλώνονται μεταξύ άλλων, το εικονίδιο της εφαρμογής, το θέμα, το όνομα της εφαρμογής. Μέσα εδώ πρέπει να δηλωθούν όλα τα activities, services, broadcast receivers και content providers πρέπει να δηλωθούν εδώ, αλλιώς το λειτουργικό δεν τα ξέρει και δεν τα τρέχει.

Για να δηλώσουμε ποια οθόνη θα ξεκινάει με το πάτημα του εικονιδίου της εφαρμογής καθώς και ποιες άλλες οθόνες θέλουμε να βρίσκονται στον launcher μας (εικονίδια με όλες τις εφαρμογές στο κινητό) πρέπει να δηλώσουμε intent filter tag. Μέσα σε αυτό το tag δηλώνουμε action και category. Όπως φαίνεται παρακάτω στο activity splash έχει δηλωθεί:

```
<intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

Αυτό σημαίνει πως η top level οθόνη θα είναι η splash κάθε φορά που ανοίγει η εφαρμογή.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.maroulis.thermidometritis"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="21" />
    <!-- Internet permission -->
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:windowSoftInputMode="adjustPan"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
        </activity>
        <activity
            android:name=".ThermidesList"
            android:label="@string/app_name" >
        </activity>
        <activity
            android:name=".GoogleCardsActivity"
            android:label="@string/app_name" >
        </activity>
        <activity
            android:name=".Thermida_details"
            android:label="@string/title_activity_thermida_details"
            android:theme="@style/CustomDialog" >
        </activity>
        <activity
            android:name=".GetCategoriesActivity"
            android:label="@string/title_activity_thermida_details" >
        </activity>
        <activity
            android:name=".ThermidesByCatIdActivity"
            android:label="@string/app_name" >
        </activity>
        <activity
            android:name=".WizardFragment"
            android:label="@string/wizard" >
        </activity>
        <activity
            android:name=".AfterWizardPage"
            android:label="@string/title_activity_after_wizard_page" >
        </activity>
        <activity
            android:name=".DiaryActivity"
            android:label="@string/title_activity_after_wizard_page" >
        </activity>
        <activity
            android:name=".SettingsActivity"
            android:label="@string/app_name"
            android:theme="@style/AppBaseTheme" >
        </activity>
        <activity
            android:name=".SplashActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".NewsActivity"
            android:label="@string/title_activity_news" >
        </activity>
    </application>

</manifest>
```

**Εικόνα 55- Android Manifest**

### Splash activity

Κάθε φορά που ανοίγει η εφαρμογή γίνεται έλεγχος για το αν έχει δημιουργηθεί προφίλ. Αν δεν έχει δημιουργηθεί ανοίγει η οθόνη δημιουργίας προφίλ, αλλιώς ανοίγει η οθόνη - dashboard με όλες τις επιλογές. Ο κώδικας αυτός φαίνεται στην παρακάτω εικόνα.

```
final int ISprofilCreated;
SharedPreferences prefs = getSharedPreferences("Wizard", MODE_PRIVATE);
ISprofilCreated =  prefs.getInt("ISprofilCreated", 0);
Log.e("ISprofilCreated", String.valueOf(ISprofilCreated));


new Handler().postDelayed(new Runnable() {

    /*
     * Showing splash screen with a timer.
     */

    @Override
    public void run() {

        if (ISprofilCreated!=0){
            Intent i = new Intent(SplashActivity.this, MainActivity.class);
            startActivity(i);
            finish();
        }else{
            Intent i = new Intent(SplashActivity.this, WizardFragment.class);
            startActivity(i);
            finish();
        }
    }
}, SPLASH_TIME_OUT);
```

**Εικόνα 56 - Splash Έλεγχος για προφίλ**

Χρησιμοποιώντας την κλάση SharedPreferences η οποία μπορεί να αποθηκεύσει τις επιλογές του χρήστη είτε κάποιες μεταβλητές για χρήση από τον προγραμματιστή, χωρίς την χρήση βάσης δεδομένων, γίνεται έλεγχος αν έχει δημιουργηθεί. Με την κλήση της prefs.getInt("ISprofilCreated", 0); κρατάμε στην ακέραια μεταβλητή ISprofilCreated την τιμή (τα ορίσματα εδώ είναι ένα string που αποτελεί το κλειδί και η τιμή του ακέραιος). Αν αυτή η τιμή είναι 0 (δηλαδή η προεπιλεγμένη) τότε σημαίνει ότι δεν έχει δημιουργηθεί το προφίλ οπότε ανοίγει ο οδηγός δημιουργίας προφίλ σε κάθε άλλη περίπτωση ανοίγει η κεντρική οθόνη. Στο activity AfterWizardPage, όπου είναι η οθόνη που ο χρήστης είναι στο τελευταίο βήμα, με το πάτημα του κουμπιού γίνεται εισαγωγή της επιλογής ISprofilCreated με τιμή 1 έτσι ώστε να μην ξανατρέξει ο οδηγός. Παρακάτω φαίνεται αυτό το σημείο.

```
Button btnOK= (Button)findViewById(R.id.btn_afterWizardOk);
btnOK.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View arg0) {
        db.addProfil(new Profil( name, age, weight, height, String.valueOf(BMI)+" ("+ResultBMI(BMI)[0]+")", String.valueOf(BMR)));
        WizardFragment.WizardActivity.finish();
         SharedPreferences prefs = getSharedPreferences("Wizard", MODE_PRIVATE);
         SharedPreferences.Editor editor = prefs.edit();
         editor.putInt("ISprofilCreated", 1);
         editor.commit();
        finish();
        Intent i =new Intent(AfterWizardPage.this, MainActivity.class);
        startActivity(i);
    }
});
```

**Εικόνα 57 - Αποθήκευση SharePreference**

Για την εγγραφή ενός shared preference είναι απαραίτητη η χρήση του Editor και με την συνάρτηση putInt ορίζουμε τιμή για το κλειδί που θέλουμε. Τέλος κάνουμε κλήση της commit() για να αποθηκευτούν οι αλλαγές.

**Sliding Menu ( Αριστερή στήλη)**

Για την δημιουργία μενού, το οποίο παραμένει ίδιο σε κάθε activity (νέα οθόνη), χρησιμοποίησα την βιβλιοθήκη SlidingMenu η οποία κάνει εφικτό την σύγχρονη χρήση αυτού του τύπου μενού σε παλαιότερης έκδοσης Android. Παρακάτω φαίνεται η αρχικοποίηση του μενού αυτού.

```
private SlidingMenu slidingMenu;
 DatabaseHandler db ;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    ViewGroup vg = (ViewGroup)findViewById(R.id.main_root);

    slidingMenu = new SlidingMenu(this);
    slidingMenu.setTouchModeAbove(SlidingMenu.TOUCHMODE_FULLSCREEN);
    slidingMenu.setShadowWidthRes(R.dimen.shadow_width);
    slidingMenu.setShadowDrawable(R.drawable.shadow);
    slidingMenu.setBehindOffsetRes(R.dimen.slidingmenu_offset);
    slidingMenu.setFadeDegree(0.35f);
    slidingMenu.attachToActivity(this, SlidingMenu.SLIDING_CONTENT);
    slidingMenu.setMenu(R.layout.slidingmenu);
```

**Εικόνα 58 - Sliding Menu**

Οι επιλογές του μενού ορίζονται στο αρχείο SlidingMenuFragment.java και η δημιουργία γίνεται με την κλήση addSectionItem(θέση-id, όνομα, όνομα εικόνας).

```
private List<SlideMenuSection> createMenu() {
    List<SlideMenuSection> sectionList = new ArrayList<SlideMenuSection>();

    SlideMenuSection generalSection = new SlideMenuSection("Γενικά");
    generalSection.addSectionItem(101,"Κατάλογος θερμίδων", "ico_heart");
    generalSection.addSectionItem(102, "Αλλαγή προφίλ", "ico_person");
    generalSection.addSectionItem(103, "Αγαπημένα", "ico_like");
    generalSection.addSectionItem(104, "Τι έφαγα σήμερα?", "ico_check");
    generalSection.addSectionItem(105, "Νέα", "ico_compose");
    generalSection.addSectionItem(106, "Σχετικά", "ico_comment");


    SlideMenuSection epilogesSection = new SlideMenuSection("Επιλογές");
    epilogesSection.addSectionItem(201, "Ρυθμίσεις", "ico_settings");
    epilogesSection.addSectionItem(204, "Έξοδος", "ico_close");

    sectionList.add(generalSection);
    sectionList.add(epilogesSection);
    return sectionList;
}
```

**Εικόνα 59 - SlidingMenuFragment προσθήκη επιλογών**

Τέλος παρακάτω βλέπουμε πώς με βάση το id - θέση του μενού ορίζουμε το τί θα γίνει κάνοντας override μια superclass της βιβλιοθήκης.

```
@Override
public boolean onChildClick(ExpandableListView parent, View v,
        int groupPosition, int childPosition, long id) {

    switch ((int)id) {
    case 101:
        Intent i= new Intent(getActivity(), GetCategoriesActivity.class);
        startActivity(i);
        break;
    case 102:
         Intent intent = new Intent(getActivity(), WizardFragment.class);
         startActivity(intent);
        break;
    case 103:
         Intent intent3 = new Intent(getActivity(), ThermidesByCatIdActivity.class);
         intent3.putExtra("favorites", true);
         startActivity(intent3);
        break;
    case 104:
         Intent intent4 = new Intent(getActivity(), DiaryActivity.class);
         startActivity(intent4);
        break;
    case 105:
         Intent intent5 = new Intent(getActivity(), NewsActivity.class);
         startActivity(intent5);
        break;
    case 201:
         Intent intent6 = new Intent(getActivity(), SettingsActivity.class);
         startActivity(intent6);
        break;
    case 106:
       // Intent i= new Intent(SlidingMenuFragment.this,)
        break;
    case 204:
        android.os.Process.killProcess(android.os.Process.myPid());
        break;
    }

    return false;
}
```

**Εικόνα 60 - Κλικ επιλογών Sliding Menu**

### Ασύγχρονος τρόπος ενημέρωσης λίστας

Στις κατηγορίες, τα φαγητά και στα νέα χρησιμοποιείται ασύγχρονος τρόπος ενημέρωσης (στο παρασκήνιο) της λίστας κάνοντας extent την AsyncTask από το Android API. Όπως φαίνεται παρακάτω προτού ξεκινήσει η διαδικασία του παρασκηνίου κάνουμε στην onPreExecute γίνεται αρχικοποίηση του παραθύρου (progress dialog) που ενημερώνει τον χρήστη να περιμένει. Στο παρασκήνιο, πριν από οτιδήποτε διαγράφουμε από την τοπική βάση τις παλιές εγγραφές (αν υπάρχουν) και μετά κάνουμε μια κλήση στον σύνδεσμο που είναι το JSON με τις κατηγορίες και το κρατάμε σε ένα object (JSONObject c). Ύστερα κάνουμε insert με την βοήθεια του μοντέλου Categories.java. Και τέλος περνάμε στην λίστα με τις κατηγορίες, το οποίο είναι τύπου ArrayList<HashMap<String, String>>, τα νέα δεδομένα. Στο onPostExecute αφαιρούμε το progress dialog και περνάμε στον adapter την νέα λίστα. Κρατάμε και σε μια επιπλέον λίστα τα δεδομένα (InitialCategoriesList) γιατί θα μας χρειαστεί στο φιλτράρισμα.

```
 /**
 * Async task class to get json by making HTTP call
 * */
public class GetCategoriesFromService extends AsyncTask<Void, Void, Void> {

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        // Showing progress dialog
        pDialog = new ProgressDialog(GetCategoriesActivity.this);
        pDialog.setMessage("Please wait...");
        pDialog.setCancelable(false);
        pDialog.show();

    }
    @Override
    protected Void doInBackground(Void... arg0) {
        // Creating service handler class instance
        ServiceHandler sh = new ServiceHandler();
        // Making a request to url and getting response
        String jsonStr = sh.makeServiceCall(url, ServiceHandler.GET);
        db.deleteCategoryTable();
        CategoriesList.clear();
        if (jsonStr != null) {
            try {

                categoriesArray = new JSONArray(jsonStr);;

                // looping through All Contacts
                for (int i = 0; i < categoriesArray.length(); i++) {
                    JSONObject c = categoriesArray.getJSONObject(i);

                    String name = c.getString(TAG_NAME);
                    String id = c.getString(TAG_ID);
                    String total = c.getString(TAG_TOTAL);
                    String image = c.getString(TAG_IMAGE);

                    Bitmap bm = Utils.downloadBitmap(image);
                    String[] imageNamearray=image.split("/");
                    String imageName=imageNamearray[imageNamearray.length-1];
                    Utils.saveFile(getApplicationContext(), bm, imageName);

                    db.addCategory(new Category(id, name, total, imageName));

                    HashMap<String, String> category = new HashMap<String, String>();
                    category.put(TAG_NAME, name);
                    category.put(TAG_ID, id);
                    category.put(TAG_TOTAL, total);
                    category.put(TAG_IMAGE, imageName);
                    CategoriesList.add(category);
                }
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
        return null;
    }
  @Override
     protected void onPostExecute(Void result) {
         super.onPostExecute(result);
         // Dismiss the progress dialog
         if (pDialog.isShowing())
             pDialog.dismiss();
         mCustomAdapter.clear();
         mCustomAdapter.addAll(getItems(CategoriesList));
         InitialCategoriesList=CategoriesList;

         swipeRefreshLayout.setRefreshing(false);
     }

 }
```

**Εικόνα 61 - Ασύγχρονη κλήση σε JSON για ενημέρωση λίστας κατηγοριών**

### Μοντέλα (Models)

Κάθε εγγραφή στους πίνακες γίνεται με την βοήθεια των μοντέλων. Κάθε μοντέλο περιγράφει τον πίνακα και όλες τις στήλες του. Παρακάτω φαίνεται το μοντέλο των αγαπημένων (ο πίνακας που κρατάει τα αγαπημένα φαγητά του χρήστη).

```
package com.maroulis.calorieswizard.model;

public class Favorites {

    //private variables
    int _id;
    int _fagito_id;


    // Empty constructor
    public Favorites(){

    }
    // constructor
    public Favorites(String id, String fagito_id){
        this._id = Integer.valueOf(id);
        this._fagito_id = Integer.valueOf(fagito_id);

    }
    // constructor
    public Favorites(int id, int fagito_id){
        this._id = id;
        this._fagito_id = fagito_id;

    }

    // getting ID
    public int getID(){
        return this._id;
    }

    // setting id
    public void setID(int id){
        this._id = id;
    }

    // getting fagito_id
    public int getFagito_id(){
        return this._fagito_id;
    }

    // setting fagito_id
    public void setFagito_id(int fagito_id){
        this._fagito_id = fagito_id;
    }


}
```

**Εικόνα 62 - Μοντέλο Αγαπημένων**

Όπως φαίνεται χρησιμοποιούμε 3 τύπους constructors, έναν άδειο έναν που δέχεται 2 ακεραίους και έναν που δέχεται ακέραιο και συμβολοσειρά. Ο πίνακας των αγαπημένων είναι απλός, αφού περιέχει μόνο το id του αγαπημένου και το id του φαγητού. Τέλος έχουν φτιαχτεί οι κατάλληλες συναρτήσεις (get - set) για τα πεδία αυτά.

### Database Handler

Η κλάση DatabaseHandler κάνει extend την SQLiteOpenHelper και είναι υπεύθυνη για την δημιουργία και επεξεργασία όλων των πινάκων της εφαρμογής. Παρακάτω βλέπουμε πως δημιουργούνται οι πίνακες αν δεν υπάρχουν (στην συνάρτηση onCreate).

```
// Creating Tables
@Override
public void onCreate(SQLiteDatabase db) {


    String CREATE_Categories_TABLE = "CREATE TABLE IF NOT EXISTS " + TABLE_CATEGORIES + "("
            + CATEGORY_KEY_ID + " INTEGER PRIMARY KEY," + CATEGORY_KEY_NAME + " TEXT,"  + CATEGORY_KEY_TOTAL + " TEXT,"
            + CATEGORY_KEY_IMAGE + " TEXT" + ")";
    String CREATE_Thermides_TABLE = "CREATE TABLE IF NOT EXISTS " + TABLE_THERMIDES + "("
            + FAGITO_KEY_ID + " INTEGER PRIMARY KEY," + FAGITO_KEY_CATEGORY_ID + " INTEGER," + FAGITO_KEY_NAME + " TEXT,"
            + FAGITO_KEY_QUANTITY + " TEXT," + FAGITO_KEY_VALUE + " TEXT," + FAGITO_KEY_CARBS + " TEXT," + FAGITO_KEY_FAT + " TEXT," + FAGITO_KEY_PROTEIN + " TEXT,
            + FAGITO_KEY_IMAGE + " TEXT" + ")";
    String CREATE_Profil_TABLE = "CREATE TABLE IF NOT EXISTS " + TABLE_PROFIL + "("
            + PROFIL_KEY_ID + " INTEGER PRIMARY KEY," + PROFIL_KEY_NAME + " TEXT,"
             + PROFIL_KEY_AGE + " TEXT,"  + PROFIL_KEY_WEIGHT + " TEXT,"  + PROFIL_KEY_HEIGHT + " TEXT,"
              + PROFIL_KEY_BMI + " TEXT,"  + PROFIL_KEY_BMR + " TEXT"
             + ")";

    String CREATE_Favorites_TABLE = "CREATE TABLE IF NOT EXISTS " + TABLE_FAVORITES + "("
            + FAVORITES_KEY_ID + " INTEGER PRIMARY KEY," + FAVORITES_KEY_FAGITO_ID + " INTEGER" + ")";

    String CREATE_Diary_TABLE = "CREATE TABLE IF NOT EXISTS " + TABLE_DIARY + "("
            + DIARY_KEY_ID + " INTEGER PRIMARY KEY," + DIARY_KEY_CATEGORY_ID + " INTEGER," + DIARY_KEY_NAME + " TEXT,"
            + DIARY_KEY_QUANTITY + " TEXT," + DIARY_KEY_VALUE + " TEXT,"
            + DIARY_KEY_DATE + " TEXT" + ")";

    db.execSQL(CREATE_Profil_TABLE);
    db.execSQL(CREATE_Categories_TABLE);
    db.execSQL(CREATE_Thermides_TABLE);
    db.execSQL(CREATE_Favorites_TABLE);
    db.execSQL(CREATE_Diary_TABLE);
}
```

**Εικόνα 63 - DatabaseHandler δημιουργία πινάκων**

Μέσα στην DatabaseHandler έχουν δημιουργηθεί μέθοδοι για την δημιουργία-επεξεργασία και διαγραφή των πινάκων αυτών. Παρακάτω βλέπουμε ένα παράδειγμα για τον πίνακα που κρατάει τις κατηγορίες των φαγητών.

```
// Adding new Category
void addCategory(Category Category) {
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(CATEGORY_KEY_ID, Category.getID());
    values.put(CATEGORY_KEY_NAME, Category.getName()); // Category Name
    values.put(CATEGORY_KEY_TOTAL, Category.getTotal()); // Category Total
    values.put(CATEGORY_KEY_IMAGE, Category.getImage()); // Category Image


    // Inserting Row
    db.replace(TABLE_CATEGORIES, null, values);
    db.close(); // Closing database connection
}
```

**Εικόνα 64 - DatabaseHandler εισαγωγή κατηγορίας**

Το getWritableDatabase ανοίγει την βάση μας (και ελέγχει τους πίνακες αν έχουν δημιουργηθεί). Κρατάμε τις μεταβλητές σε ένα map και τις περνάμε με την βοήθεια της replace που δέχεται σαν ορίσματα: το όνομα του πίνακα, τις τιμές αν είναι κενές οι μεταβλητές και το array map με τις μεταβλητές μας.

Ο μέθοδος διαγραφής μιας κατηγορίας φαίνεται παρακάτω.

```
// Deleting single Category
public void deleteCategory(Category Category) {
    SQLiteDatabase db = this.getWritableDatabase();
    db.delete(TABLE_CATEGORIES, CATEGORY_KEY_ID + " = ?",
            new String[] { String.valueOf(Category.getID()) });
    db.close();
}
```

**Εικόνα 65 - DatabaseHandler διαγραφή κατηγορίας**

Η Μέθοδος delete δέχεται σαν όρισμα ένα category και μας βοηθάει να διαγράψουμε μια κατηγορία με βάση το ID (που το παίρνουμε με την βοήθεια του μοντέλου μας).

Για να φέρουμε τις τιμές ενός πίνακα έχουν φτιαχτεί ανά πίνακα οι αντίστοιχες συναρτήσεις. Παρακάτω φαίνεται πως παίρνουμε μια κατηγορία.

```
// Getting single Category
Category getCategory(int id) {
    SQLiteDatabase db = this.getReadableDatabase();

    Cursor cursor = db.query(TABLE_CATEGORIES, new String[] { CATEGORY_KEY_ID,
            CATEGORY_KEY_NAME, CATEGORY_KEY_TOTAL, CATEGORY_KEY_IMAGE }, CATEGORY_KEY_ID + "=?",
            new String[] { String.valueOf(id) }, null, null, null, null);
    if (cursor != null)
        cursor.moveToFirst();

    Category Category = new Category(cursor.getString(0),
            cursor.getString(1), cursor.getString(2), cursor.getString(3));
    // return Category
    return Category;
}
```

**Εικόνα 66 - DatabaseHandler επιλογή κατηγορίας με βάση το id**

Όπως φαίνεται και στην πιο πάνω εικόνα κάνουμε χρήση του Cursor για σειριακή προσπέλαση των δεδομένων (με βάση το categoryID) και δημιουργούμε ένα νέο Category το οποίο και γυρνάμε.

Για να πάρουμε όλα τα δεδομένα από έναν πίνακα έχουν φτιαχτεί μέθοδοι τύπου λίστας που κάνουν SELECT όλα τα στοιχεία του πίνακα. Πιο κάτω το παράδειγμα λίστας κατηγορίας.

```
// Getting All Categories
public List<Category> getAllCategories() {
    List<Category> CategoryList = new ArrayList<Category>();
    // Select All Query
    String selectQuery = "SELECT  * FROM " + TABLE_CATEGORIES;

    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery(selectQuery, null);

    // looping through all rows and adding to list
    if (cursor.moveToFirst()) {
        do {
            Category Category = new Category();
            Category.setID(Integer.parseInt(cursor.getString(0)));
            Category.setName(cursor.getString(1));
            Category.setTotal(cursor.getString(2));
            Category.setImage(cursor.getString(3));
            // Adding Category to list
            CategoryList.add(Category);
        } while (cursor.moveToNext());
    }

    // return Category list
    return CategoryList;
}
```

**Εικόνα 67 - Επιλογή όλων των κατηγοριών**

Εδώ βλέπουμε πώς γίνεται προσπέλαση όλων των δεδομένων με την βοήθεια της while. Σε κάθε επανάληψη δημιουργείται ένα νέο Category και αφού μπουν οι τιμές προστίθεται σε μια λίστα την οποία στο τέλος την επιστρέφουμε.

### Υπολογισμός BMI

Για τον υπολογισμό του BMI έχει φτιαχτεί η μέθοδος calculateBMI η οποία δέχεται σαν ορίσματα το ύψος (συμβολοσειρά) και το βάρος (συμβολοσειρά).

```
public static double calculateBMI(String height, String weight){

    double h= Double.valueOf(height)*0.01;// metatroph se metra
    double w= Double.valueOf(weight);
    double BMI = 0.0;
    BMI = ((w )/(h * h));


    return round(BMI,2);
}
public static double round(double value, int places) {
    if (places < 0) throw new IllegalArgumentException();

    BigDecimal bd = new BigDecimal(value);
    bd = bd.setScale(places, RoundingMode.HALF_UP);
    return bd.doubleValue();
}
```

**Εικόνα 68 - Υπολογισμός δείκτη BMI**

Το πρώτο που κάνουμε είναι η μετατροπή του ύψους και του βάρους σε double και μετά σε μέτρα. Ο δείκτης BMI είναι το βάρος διά το τετράγωνο του ύψους. Με την χρήση της συνάρτησης round κάνουμε στρογγυλοποίηση και γυρνάμε το αποτέλεσμα.

### Υπολογισμός BMR

Για τον υπολογισμό του δείκτη BMR έχει φτιαχτεί η παρακάτω μέθοδος.

```
public static double calculateBMR(boolean isMale, String age_string, String height_string, String weight_string, Double activity){

  double BMR;

  int age= Integer.valueOf(age_string);
  int height=Integer.valueOf(height_string);
  int weight=Integer.valueOf(weight_string);

  // Basismeno sthn formoula tou Mifflin-St
  if (isMale){
       BMR = (int)((9.99 * weight) + (6.25 * height) - (4.92 * age) + 5);
  }
  else{
      BMR = (int)( (9.99 * weight) + (6.25 * height) - (4.92 * age) - 161);

  }

  BMR*=activity;

  return BMR;

}
```

**Εικόνα 69 - Υπολογισμός δείκτη BMR**

Η μέθοδος αυτή δέχεται σαν ορίσματα: το φύλο (άνδρας - γυναίκα), την ηλικία, το ύψος, το βάρος και τον δείκτη δραστηριότητας (activity). Η μέθοδος υπολογισμού είναι βασισμένη στον τύπο του Mifflin - St Jeor και είναι η πιο ακριβής μέθοδος μέχρι σήμερα. Αφού γίνει μετατροπή των τιμών του ύψους, του βάρους και της ηλικίας σε ακέραιους γίνεται διαχωρισμός με βάση το φύλο και υπολογίζεται το BMR. Η τιμή αυτή για να είναι ακριβής πρέπει μετά να πολλαπλασιαστεί με το activity level (δραστηριότητα). Ο τιμές του activity level φαίνονται παρακάτω.

```
public static double GetBMRvariable(String activity){

    double var;
    switch (activity) {
    case "Πολύ Μικρή":
        var= 1.2;
        break;
    case "Μέτρια":
        var= 1.375;
        break;
    case "Έντονη":
        var= 1.55;
        break;
    case "Καθημερινή":
        var= 1.725;
        break;
    case "Αθλητική":
        var= 1.9;
        break;

    default:
        var= 1.2;
    }
    return var;

}
```

**Εικόνα 70 - Υπολογισμός activity**

Η συνάρτηση αυτή παίρνει σαν όρισμα την επιλογή του χρήστη από την φόρμα που υπάρχει στον οδηγό δημιουργίας προφίλ. Με βάση αυτή την επιλογή επιστρέφει την κατάλληλη τιμή.

Με βάση τον στόχο που έχει επιλέξει ο χρήστης αφαιρούνται ή προσθέτουμε στο συνολικό δείκτη BMR τις αντίστοιχες θερμίδες. Για παράδειγμα ένα κιλό ισούται με 3500 θερμίδες. Αν ο χρήστης θέλει να χάνει 1 κιλό την εβδομάδα θα πρέπει να μειώσει τις ημερήσιες θερμίδες του κατά 500 (3500/7) περίπου. Παρακάτω παρουσιάζεται η συνάρτηση που γυρνάει το αριθμό των θερμίδων που πρέπει να αφαιρείται. Σαν όρισμα παίρνει τον στόχο που έχει επιλέξει ο χρήστης στον οδηγό δημιουργίας προφίλ.

```
public static int GetTarget(String target){

        int var;
        switch (target) {
        case "Να χάνω 1 κιλό την βδομάδα":
            var= -500;
            break;
        case "Να χάνω 1 κιλό τον μήνα":
            var= -118;
            break;
        case "Να χάνω 3 κιλά τον μήνα":
            var= -350;
            break;
        case "Να διατηρήσω το βάρος μου":
            var= 0;
            break;
        case "Να πάρω 1 κιλό την βδομάδα":
            var= 500;
            break;
        case "Να πάρω 1 κιλό τον μήνα":
            var= 118;
            break;
        case "Να πάρω 3 κιλά τον μήνα":
            var= 350;
            break;

        default:
            var= 0;
        }
        return var;

    }
```

**Εικόνα 71 - Συνάρτηση υπολογισμού στόχου**

### Έλεγχος για σύνδεση στο internet

Παρόλο που η εφαρμογή διαθέτει τοπική βάση για να αποθηκεύει και να διαχειρίζεται τα δεδομένα, την πρώτη φορά που ανοίγει αλλά και σε περίπτωση που ο χρήστης θελήσει να κάνει ανανέωση πρέπει να ελέγξουμε αν υπάρχει διαθέσιμη σύνδεση. Ο έλεγχος αυτός επειδή χρησιμοποιείται σε αρκετά σημεία της εφαρμογής βρίσκεται στο αρχείο Utils.java. Παρακάτω φαίνονται η συναρτήσεις για έλεγχο διαθέσιμης σύνδεσης και προβολή μηνύματος στο χρήστη σε περίπτωση που δεν είναι συνδεμένος (haveNetworkConnection και showNoInternetDialog αντίστοιχα).

```
public static boolean haveNetworkConnection(Context c) {
    boolean haveConnectedWifi = false;
    boolean haveConnectedMobile = false;

    ConnectivityManager cm = (ConnectivityManager) c.getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo[] netInfo = cm.getAllNetworkInfo();
    for (NetworkInfo ni : netInfo) {
        if (ni.getTypeName().equalsIgnoreCase("WIFI"))
            if (ni.isConnected())
                haveConnectedWifi = true;
        if (ni.getTypeName().equalsIgnoreCase("MOBILE"))
            if (ni.isConnected())
                haveConnectedMobile = true;
    }
    return haveConnectedWifi || haveConnectedMobile;
}


@SuppressWarnings("deprecation")
public static void showNoInternetDialog(Context c) {
 try {
        final AlertDialog alertDialog = new AlertDialog.Builder(c).create();

        alertDialog.setTitle("Info");
        alertDialog.setMessage("Δεν υπάρχει διαθέσιμη σύνδεση στο ιντερνετ. " +
                "Ενεργοποιήστε την σύνδεση και ξαναδοκιμάστε.");
        alertDialog.setButton("OK", new DialogInterface.OnClickListener() {
           public void onClick(DialogInterface dialog, int which) {
             alertDialog.dismiss();
           }
        });

        alertDialog.show();
        }
        catch(Exception e)
        {
            Log.d("ERROR IN DIALOG","Show Dialog: "+e.getMessage());
        }
   }
```

**Εικόνα 72 - Έλεγχος για σύνδεση στο ιντερνέτ**

Η συνάρτηση haveNetworkConnection παίρνει σαν όρισμα το context και κάνει αρχικοποίηση του NetworkInfo (το οποίο μας βοηθάει να βρούμε τις διαθέσιμες συνδέσεις στην συσκευή). Στην συνέχεια αν ο χρήστης έχει διαθέσιμη σύνδεση (data ή wi-fi) επιστέφει true αλλιώς επιστρέφει false.

Η showNoInternetDialog χρησιμοποιώντας το alertDialog του android δείχνει στον χρήστη ένα παράθυρο που τον προτρέπει να ανοίξει το ιντερνέτ.

### Κατέβασμα εικόνας από το ιντερνέτ (απομακρυσμένο service).

Η εφαρμογή στο κινητό όταν διαβάσει από το απομακρυσμένο site (JSON) τα δεδομένα θα πάρει και ένα link με την εικόνα (αν έχει βάλει ο διαχειριστής). Αυτό το link η εφαρμογή το χρησιμοποιεί για να πάρει την εικόνα και να την αποθηκεύσει τοπικά. Παρακάτω δείχνω αυτές τις 2 συναρτήσεις.

```
public static Bitmap downloadBitmap(String url) {
    // initilize the default HTTP client object
    final DefaultHttpClient client = new DefaultHttpClient();

    //forming a HttoGet request
    final HttpGet getRequest = new HttpGet(url);
    try {

        HttpResponse response = client.execute(getRequest);

        //check 200 OK for success
        final int statusCode = response.getStatusLine().getStatusCode();

        if (statusCode != HttpStatus.SC_OK) {
            Log.w("ImageDownloader", "Error " + statusCode +
                    " while retrieving bitmap from " + url);
            return null;

        }

        final HttpEntity entity = response.getEntity();
        if (entity != null) {
            InputStream inputStream = null;
            try {
                // getting contents from the stream
                inputStream = entity.getContent();

                // decoding stream data back into image Bitmap that android understands
                final Bitmap bitmap = BitmapFactory.decodeStream(inputStream);

                return bitmap;
            } finally {
                if (inputStream != null) {
                    inputStream.close();
                }
                entity.consumeContent();
            }
        }
    } catch (Exception e) {
        // You Could provide a more explicit error message for IOException
        getRequest.abort();
        Log.e("ImageDownloader", "Something went wrong while" +
                " retrieving bitmap from " + url + e.toString());
    }

    return null;
}
```

**Εικόνα 73 - Κατέβασμα εικόνας από το ιντερνέτ**

```
public static void saveFile(Context context, Bitmap b, String picName){
    FileOutputStream fos;
    try {
        fos = context.openFileOutput(picName, Context.MODE_PRIVATE);
        b.compress(Bitmap.CompressFormat.PNG, 100, fos);
        fos.close();
    }
    catch (FileNotFoundException e) {
        Log.d("not found", "file not found");
        e.printStackTrace();
    }
    catch (IOException e) {
        Log.d("io", "io exception");
        e.printStackTrace();
    }

}
```

**Εικόνα 74 - Αποθήκευση εικόνας τοπικά**

Με το που κατέβει η εικόνα την κρατάμε σε μια μεταβλητή τύπου bitmap και ύστερα την αποθηκεύουμε στην συσκευή. Ένα παράδειγμα χρήσης αυτών των 2 συναρτήσεων φαίνεται όταν κατεβάζουμε όλες τις κατηγορίες (στο παρασκήνιο).

```
String image = c.getString(TAG_IMAGE);

Bitmap bm = Utils.downloadBitmap(image);
String[] imageNamearray=image.split("/");
String imageName=imageNamearray[imageNamearray.length-1];
Utils.saveFile(getApplicationContext(), bm, imageName);
```

**Εικόνα 75 - Αποθήκευση εικόνας κατηγορίας**

Όπως φαίνεται κρατάμε το bitmap που επιστρέφει η downloadBitmap και αφού πάρουμε το όνομα του αρχείου καλούμε την savefile η οποία πάει και αποθηκεύει την εικόνα μας στο κινητό.

### Ενημέρωση της βάσης δεδομένων της εφαρμογής (αυτόματα)

Ο θερμιδομετρητής για να μπορεί να λειτουργεί και χωρίς σύνδεση αποθηκεύει τα δεδομένα τοπικά όπως έχουμε δείξει. Ο χρήστης παρόλο που μπορεί να κάνει ενημέρωση των δεδομένων όποτε θέλει, έχει δημιουργηθεί μια αυτοματοποιημένη διαδικασία (την οποία ο χρήστης μπορεί να απενεργοποιήσει από τις ρυθμίσεις της εφαρμογής) για έλεγχο και ενημέρωση των δεδομένων σε συγκεκριμένη χρονική περίοδο η οποία ορίζεται από τις ρυθμίσεις. Στις κατηγορίες και στην λίστα των φαγητών έχει δημιουργηθεί η παρακάτω διαδικασία.

```
//get default setting from SettingActivity
SharedPreferences sharedPrefs = PreferenceManager.getDefaultSharedPreferences(this);
boolean perform_updates = sharedPrefs.getBoolean("perform_updates",true);
int updates_interval = Integer.valueOf(sharedPrefs.getString("updates_interval", "7"));

//get last update time
 SharedPreferences prefs = getPreferences(0);

 lastUpdateTime =  prefs.getLong("lastUpdateTime", 0);

    if (((lastUpdateTime + (updates_interval*(24 * 60 * 60 * 1000))) < System.currentTimeMillis()) && perform_updates) {
        if (Utils.haveNetworkConnection(this)){
            Toast.makeText(getApplicationContext(), "Update from Service", Toast.LENGTH_LONG).show();
            lastUpdateTime = System.currentTimeMillis();
            SharedPreferences.Editor editor = getPreferences(0).edit();
            editor.putLong("lastUpdateTime", lastUpdateTime);
            editor.commit();

            new GetCategoriesFromService().execute();
         }else{
            new GetCategoriesFromLocalDB().execute();
         }

    }else{
        new GetCategoriesFromLocalDB().execute();
    }
```

**Εικόνα 76 - Ενημέρωση της βάσης δεδομένων της εφαρμογής (αυτόματα)**

,Χρησιμοποιώντας την κλάση SharedPreferences του Android κρατάμε/ αποθηκεύουμε τις επιλογές του χρήστη (χωρίς χρήση βάσης). Η εφαρμογή με βάση αυτές τις επιλογές ελέγχει αν ο χρήστης έχει ενεργοποιημένες τις αυτόματες ενημερώσεις και αν ο χρόνος που έγινε η τελευταία ενημέρωση υπερβαίνει τον χρόνο που έχει οριστεί στις ρυθμίσεις, καλείται η GetCategoriesFromService η οποία επικοινωνεί με το απομακρυσμένο service (αφού πρώτα ενημερωθεί η ημερομηνία που έγινε η τελευταία ενημέρωση). Σε κάθε άλλη περίπτωση η λίστα έρχεται από την τοπική βάση δεδομένων.

### Επιλογές χρήστη (Settings)

Για να αποθηκεύσουμε τις επιλογές του χρήστη κάνουμε extend την PreferenceActivity και όλες οι επιλογές είναι γραμμένες στο xml αρχείο preferences. Παρακάτω βλέπουμε πως δημιουργήθηκαν οι ρυθμίσεις.

```
public class SettingsActivity extends PreferenceActivity {

    @SuppressWarnings("deprecation")
    @Override
    public void onCreate(Bundle savedInstanceState) {
        this.requestWindowFeature(Window.FEATURE_NO_TITLE);
        super.onCreate(savedInstanceState);

        addPreferencesFromResource(R.xml.preferences);

        SharedPreferences sharedPrefs = PreferenceManager.getDefaultSharedPreferences(this);
    }
}
```

**Εικόνα 77- Επιλογές - Settings**

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">
    <PreferenceCategory
        android:title="Επιλογές"
        android:key="first_category">
        <CheckBoxPreference
            android:key="perform_updates"
            android:summary="Ενεργοποίηση απενεργοποίηση αύτοματων ενημερώσεων"
            android:title="Ενεργοποίηση αυτόματων ενημερώσεων"
            android:defaultValue="true"  />
        <ListPreference
            android:key="updates_interval"
            android:title="Συχνότητα ενημερώσεων"
            android:summary="Πόσο συχνά θέλετε να γίνονται οι ενημερώσεις?"
            android:defaultValue="7"
            android:entries="@array/updateInterval"
            android:entryValues="@array/updateIntervalValues"
            android:dependency="perform_updates" />
    </PreferenceCategory>

    <PreferenceCategory
        android:title="Προφίλ"
        android:key="second_category">
        <Preference
        android:title="Αλλαγή Προφίλ"
        android:summary="Ακολουθήστε τον οδήγο για να αλλάξετε το προφίλ σας.">
            <intent
                android:action="android.intent.action.VIEW"
                android:targetPackage="com.maroulis.thermidometritis"
                android:targetClass="com.maroulis.thermidometritis.WizardFragment"
            />
        </Preference>
    </PreferenceCategory>

</PreferenceScreen>
```

**Εικόνα 78 - Επιλογές - Settings Layout**

Όπως βλέπουμε γίνεται απλή χρήση του API για την δημιουργία των settings και όλο το layout/interface δημιουργείται με την χρήση xml. Το preferencecategory tag δηλώνει μια καινουργιά κατηγορία στις ρυθμίσεις και το preference, checkbox preference και list preference τον τύπο της επιλογής.