Πανεπιστήμιο Θεσσαλίας
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Υλοποίηση εφαρμογής διαχείρισης προϊόντων σε ευφυές ψυγείο με χρήση ΣΔΒΔ

Διπλωματική Εργασία

ΑΝΑΣΤΑΣΙΟΥ Π. ΔΡΟΣΟΥ

Επιβλέπων
Μιχαήλ Βασιλακόπουλος
Αναπληρωτής Καθηγητής

Βόλος, Ιούλιος 2019

Πανεπιστήμιο Θεσσαλίας
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Υλοποίηση εφαρμογής διαχείρισης προϊόντων σε ευφυές ψυγείο με χρήση ΣΔΒΔ

Διπλωματική Εργασία

ΑΝΑΣΤΑΣΙΟΥ Π. ΔΡΟΣΟΥ

Επιτροπή επίβλεψης

Επιβλέπων	Συνεπιβλέπουσα	Συνεπιβλέπουσα
Μιχαήλ Βασιλακόπουλος	Ελένη Τουσίδου	Ασπασία Δασκαλοπούλου
Αναπληρωτής Καθηγητής	Μέλος Ε.ΔΙ.Π.	Επίκουρη Καθηγήτρια

Βόλος, Ιούλιος 2019

Πανεπιστήμιο Θεσσαλίας
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Ο/Η συγγραφέας αυτής της εργασίας βεβαιώνει ότι κάθε βοήθεια την οποία είχε για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης βεβαιώνει ότι έχει αναφέρει τις όποιες πηγές από τις οποίες έκανε χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται επακριβώς, είτε παραφρασμένες.

University of Thessaly
Faculty of Engineering
Department of Electrical & Computer Engineering

Implementation product management in smart refrigerator using DBMS

Diploma Thesis

ANASTASIOS DROSOS

Supervisor
Michael Vassilakopoulos
Associate Professor

Volos, July 2019

Περίληψη

Η τεχνολογία αναπτύσσεται συνεχώς, με αποτέλεσμα οι ανθρώπινες συνήθειες να αλλάζουν και οι απαιτήσεις να αυξάνονται. Η δημιουργία ολοένα και περισσότερων έξυπνων συσκευών κρίνεται πλέον απαραίτητη για την ικανοποίηση των ανθρώπινων αναγκών, προσφέροντας συνεχώς περισσότερες δυνατότητες κι ανέσεις στην καθημερινή ζωή. Συνηθισμένα παραδείγματα αποτελούν οι έξυπνες τηλεοράσεις, τα έξυπνα κλιματιστικά, τα έξυπνα συστήματα φωτισμού κι ανίχνευσης καπνού, τα έξυπνα κουδούνια και κλειδαριές και πολλά άλλα. Η τεχνολογία αυτή, η οποία ονομάζεται Διαδίκτυο των πραγμάτων (Internet of Things - IoT), γίνεται πλέον τρόπος ζωής. Η κατοικία του ανθρώπου αποτελεί ένα βασικό πεδίο εφαρμογής αυτής της τεχνολογίας του IoT. Ένας από τους σημαντικότερους χώρους του σπιτιού είναι η κουζίνα, όπου υπάρχει (συνήθως) το ψυγείο, μια από τις βασικότερες και πιο αναγκαίες συσκευές. Η παρούσα εργασία παρουσιάζει μια διαδικτυακή εφαρμογή (web application) έξυπνου ψυγείου, η οποία δίνει τη δυνατότητα στους ανθρώπους να διαχειρίζονται με μεγαλύτερη ευκολία τα προϊόντα που υπάρχουν στο ψυγείο τους. Με χρήση έξυπνης τεχνολογίας, η εφαρμογή θα έχει τη δυνατότητα να κρατά αρχείο των τροφίμων που βρίσκονται μέσα στο ψυγείο και να ειδοποιεί τους χρήστες της για τις ημερομηνίες λήξης των προϊόντων. Μια ακόμα λειτουργία είναι αυτή των ηλεκτρονικών παραγγελιών. Ο άνθρωπος θα έχει τη δυνατότητα να ετοιμάσει και να στείλει ηλεκτρονικές παραγγελίες σε σουπερμάρκετ (supermarkets) για την αγορά των τροφίμων που επιθυμεί. Επίσης, θα μπορεί να ετοιμάσει και να προγραμματίσει αυτοματοποιημένες τέτοιες παραγγελίες, απαλλάσοντας τον άνθρωπο από το άγχος να μην ξεχαστεί η αγορά διάφορων κρίσιμων προϊόντων. Όλες αυτές οι λειτουργίες θα είναι διαθέσιμες σε οποιαδήποτε άλλη συσκευή συνδεδεμένη στο διαδίκτυο, πέρα από το ψυγείο. Έτσι, εάν ο χρήστης απουσιάζει, μπορεί να λαμβάνει ειδοποιήσεις προϊόντων και παραγγελιών της εφαρμογής στο κινητό του. Η εφαρμογή αυτή καλείται ”Smart Fridge” κι αναπτύχθηκε στα πλαίσια της εργασίας αυτής, με σκοπό να διευκολύνει τους χρήστες να επιμεληθούν τα προϊόντα που έχουν στο ψυγείο τους και να αυτοματοποιήσει την αγορά όσων κρίνονται ως απαραίτητα. Η εφαρμογή σχεδιάστηκε κι αναπτύχθηκε σε django δομή δικτύου (framework) και είναι γραμμένη σε γλώσσες προγραμματισμού python, html και css, χρησιμοποιώντας επίσης την βιβλιοθήκη Bootstrap4. Τέλος, για τη δημιουργία και διαχείριση της βάσης δεδομένων χρησιμοποιήθηκε το σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων sqlite3.

Λέξεις Κλειδιά

εφαρμογή ιστού, έξυπνο ψυγείο, διαδικτυακές παραγγελίες, πλαίσιο django, python, βάση δεδομένων

Abstract

Technology is constantly evolving and as a result human habits change and demands grow. Creating more and more smart devices is now necessary to meet human needs, offering more and more possibilities and amenities in everyday life. Typical examples are smart TVs, smart air conditioners, smart smoke detection and lighting systems, smart bells and locks and more. This technology, called the Internet of Things (IoT), is now a way of life. Human dwelling is a basic scope of this IoT technology. One of the most important areas of the house is the kitchen, where (usually) the refrigerator is one of the most basic and most necessary appliances. This paper presents a smart refrigerator web application that allows people to manage in a more efficient the products in their refrigerator. Using smart technology, the app will be able to keep a record of foods in the refrigerator and notify its users about the expiration dates of the products. Another function is that of electronic orders. The person will be able to prepare and send electronic orders at a supermarket to buy foods of preference. It will also be possible to prepare and plan automated orders, relieving people of the stress of forgetting to purchase various critical products. All of these features will be also available on any other device connected to the internet, beyond the refrigerator. So, if the user is absent, he/she can receive product and order notifications on his/her mobile app. This application is called ”Smart Fridge” and has been developed in the context of this work to make it easier for users to process their refrigerated products and automate the purchase of what is deemed necessary. The application was designed and developed into django web framework and is written in python, html and css programming languages, also using the Bootstrap4 library. Finally, the sqlite3 database management system was used to create and manage the database.

Keywords

web application, smart refrigerator, online orders, django framework, python, database

Αφιερωμένο στα νιάτα μου και στα φοιτητικά μου χρόνια.

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τους γονείς μου για την καθοδήγηση και την ηθική συμπαράσταση που μου προσέφεραν όλα αυτά τα χρόνια. Επίσης, ευχαριστώ ιδιαίτερα τον κύριο Βασιλακόπουλο Μιχαήλ για την επίβλεψη αυτής της διπλωματικής εργασίας και για την ευκαιρία που μου έδωσε να την εκπονήσω. Τέλος, θα ήθελα να ευχαριστήσω την κυρία Τουσίδου Ελένη, μιας και χωρίς την καθοδήγηση, στήριξη, συμβουλές και διορθώσεις της οποίας η παρούσα εργασία δεν θα αποτελούσε πραγματικότητα.

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

Κεφάλαιο 1

Εισαγωγή

Όλα τα σύγχρονα νοικοκυριά διαθέτουν ένα, τουλάχιστον, ψυγείο για τη διατήρηση των προϊόντων τους. Η τεχνολογία αναπτύσσεται συνεχώς, κι αυτό έχει ως αποτέλεσμα σήμερα να έχουμε στη διάθεσή μας ψυγεία πιο σύγχρονα, με περισσότερο αποθηκευτικό χώρο για πολλά προϊόντα, πολλές πόρτες, θέσεις, κλπ. Υπάρχουν ακόμα κι έξυπνα ψυγεία που δέχονται ηχητικές εντολές για να μπορεί ο χρήστης να ρυθμίσει τη θερμοκρασία ή ακόμα να χρησιμοποιήσει τις αγαπημένες του εφαρμογές, όπως facebook ή youtube, μέσω ενσωματωμένης οθόνης και λογισμικού. Όσο όμως αυξάνονται οι δυνατότητες, ο χώρος του ψυγείου και τα προϊόντα που χωρά, τόσο πιο δύσκολο είναι για τον χρήστη να τα επιβλέπει. Είναι πολλές οι περιπτώσεις που κάποιο προϊόν θα ξεχαστεί στο ψυγείο, ενώ θα είναι πια ληγμένο. Επίσης, είναι πολλές οι περιπτώσεις που κάποιο προϊόν που ο χρήστης θα ήθελε να βρίσκεται στο ψυγείο του, έχει ξεχάσει να το αγοράσει. Είναι δύσκολο για τον άνθρωπο να θυμάται ανά πάσα στιγμή τι προϊόντα υπάρχουν στο ψυγείο του, και τι χρειάζεται να αγοράσει.

1.1 Αντικείμενο της διπλωματικής

Σκοπός της διαδικτυακής εφαρμογής **Smart Fridge** που παρουσιάζει αυτή η διπλωματική είναι η καλύτερη διαχείριση των προϊόντων του ψυγείου και η αυτοματοποίηση της αγοράς αυτών που χρειάζονται, μέσω παραγγελιών σε σουπερμάρκετ. Η συνολική προσπάθεια βασίζεται στην τεχνολογία του Διαδικτύου των πραγμάτων. Για την βελτιστοποίηση της διαχείρισης των προϊόντων του ψυγείου, αρχικά χρειάζεται ένας τρόπος ενημέρωσης της εφαρμογής για το ποια προϊόντα εισέρχονται/εξέρχονται στο/από το ψυγείο, ώστε να υπάρχει μια λίστα με όλα τα προϊόντα που βρίσκονται μέσα στο ψυγείο ανά πάσα στιγμή. Η λίστα αυτή αποτελεί βασικό εργαλείο της εφαρμογής προκειμένου να ενημερώνει τον χρήστη για τις ημερομηνίες λήξεως των τροφίμων, ενώ παράλληλα τον βοηθά να ελέγχει τι υπάρχει μέσα στο ψυγείο του από την οθόνη του ψυγείου ή από απόσταση, μέσω κάποιας άλλης συσκευής, γρήγορα κι αποτελεσματικά. Για τη δημιουργία παραγγελιών είναι απαραίτητη η ύπαρξη αλγορίθμου θα που καθορίζει, με βάση τις οδηγίες που θα έχουν δοθεί από τον χρήστη, πότε πρέπει να σταλεί η παραγγελία και ποια προϊόντα πρέπει να περιέχει. Για τη λειτουργία της εφαρμογής, χρειάζεται επίσης μια βάση δεδομένων, η οποία θα περιέχει όλα τα διαθέσιμα προϊόντα για τον χρήστη. Η βάση αυτή θα περιλαμβάνει όλα τα

προϊόντα της βάσης δεδομένων του συνεργαζόμενου σουπερμάρκετ (μας ενδιαφέρουν μόνο τα τρόφιμα ψυγείου) και θα ανανεώνεται ανά τακτά χρονικά διαστήματα, κατόπιν συνεργασίας με τον διαχειριστή της βάσης του σουπερμάρκετ. Στα πλαίσια της εργασίας, θεωρούμε αυτή τη βάση ήδη αρχικοποιημένη με τα απαραίτητα στοιχεία για την παρουσίαση κι επίδειξη της εφαρμογής. Τέλος, επειδή πρόκειται για μια διαδικτυακή εφαρμογή η οποία μπορεί να χρησιμοποιηθεί είτε από την οθόνη του ψυγείου, είτε από μια διαφορετική συσκευή που επιθυμεί ο χρήστης, όπως smartphone ή tablet, για την απομακρυσμένη πρόσβαση και χρήση της εφαρμογής είναι απαραίτητη η σύνδεση των συσκευών και του ψυγείου στο διαδίκτυο.

1.1.1 Συνεισφορά

Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

1. Σχεδιάστηκε κι αναπτύχθηκε πλήρες σχήμα βάσης δεδομένων για την επίτευξη των σκοπών της εφαρμογής.

2. Δημιουργήθηκε σύστημα λογαριασμών χρηστών, ώστε πολλοί χρήστες να μπορούν να χρησιμοποιούν ταυτόχρονα την εφαρμογή και κάθε χρήστης να επεξεργάζεται τα δικά του δεδομένα.

3. Δημιουργήθηκαν αλγόριθμοι εισαγωγής/εξαγωγής προϊόντων στο ψυγείο, μέσω ανάγνωσης του αριθμού barcode κάθε προϊόντος.

4. Ενσωματώθηκε σύστημα ανάγνωσης αριθμού barcode (σαρωτής barcode). Το σύστημα αποτελείται από μια εφαρμογή android κι έναν διακομιστή (server) κι αυτό που επιτυγχάνει είναι η χρήση του android smartphone ως έναν barcode scanner, για καταγραφή των προϊόντων που εισέρχονται/εξέρχονται στο/από το ψυγείο.

5. Αναπτύχθηκε αλγόριθμος δημιουργίας πολλών και διαφορετικών ηλεκτρονικών παραγγελιών προς σουπερμάρκετ, με περιεχόμενο προϊόντα τα οποία επιλέγει μόνος του ο χρήστης.

6. Αναπτύχθηκε αλγόριθμος δημιουργίας αυτόματης παραγγελίας, με περιεχόμενο προϊόντα τα οποία κρίνει η εφαρμογή ότι θέλει ο χρήστης, με βάση τις προτιμήσεις του και τα προϊόντα που υπάρχουν ήδη στο ψυγείο.

1.2 Ιστορικό υπόβαθρο - Επιστημονικά έργα

Γενικότερα ως έννοια, το έξυπνο ψυγείο είναι ένα ψυγείο το οποίο έχει προγραμματιστεί να καταλαβαίνει τι είδους προϊόντα αποθηκεύονται μέσα σε αυτό και να κρατά αρχείο των προϊόντων αυτών μέσω τεχνολογίας barcode ή RFID (ηλεκτρονική ετικέτα). Τα ψυγεία αυτού του είδους είναι συχνά εξοπλισμένα έτσι ώστε να καθορίζουν αυτόματα πότε ένα προϊόν χρήζει αναπλήρωσης [25]. Επίσης, παρέχουν επιπλέον ειδικές πληροφορίες των προϊόντων στους χρήστες, όπως στοιχεία θρεπτικής αξίας ή το ιστορικό κατανάλωσης [1]. Ακόμα, ένα έξυπνο ψυγείο μπορεί να κατεβάσει συνταγές από το διαδίκτυο, να επικοινωνήσει με έναν φούρνο μικροκυμάτων για να προετοιμάσει

γεύματα ή να προτείνει ποτά κι αναψυκτικά στον χρήστη, λαμβάνοντας υπόψη τα περιεχόμενά του, τη θερμοκρασία και την ώρα της ημέρας [13].

Το πρώτο ψυγείο που συνδέθηκε στο διαδίκτυο καλείται Ποσοτικοποιημένο Ψυγείο (Quantified Fridge) και βρισκόταν σε ένα σπίτι 100 χρόνων στην Ολλανδία στις 12 Ιουλίου 1998, όπου μαζί με αυτό υπήρχαν διαδικτυακά φώτα, κουδούνι, γραμματοκιβώτιο και μια τουαλέτα. Η μόνη λειτουργικότητα που διαθέτει είναι η ηχογράφηση κι εκπομπή του σήματος κάθε φορά που η πόρτα του ψυγείου ανοίγει. Στα τελευταία 16 χρόνια, η πόρτα του ψυγείου έχει ανοίξει περίπου 70.000 φορές [11].

Το 1999, το Electrolux Screenfridge, ένα ψυγείο συνδεδεμένο στο διαδίκτυο, έδινε τη δυνατότητα στους χρήστες να κάνουν παραγγελίες προϊόντων παντοπωλείου μέσω ίντερνετ. Το 2000, η εταιρία Whirlpool/Cisco δημιούργησε ένα ψυγείο το οποίο έδινε τη δυνατότητα στους χρήστες να αναζητούν συνταγές ή να βλέπουν εκπομπές με τους αγαπημένους τους σεφ στην οθόνη του ψυγείου, όλα αυτά με τη χρήση ενός ενσωματομένου φυλλομετρητή. Το 2002, η Whirlpool επέκτεινε την ιδέα του έξυπνου ψυγείου, ώστε το ψυγείο να χρησιμοποιείται κι ως εργαλείο πολυμέσων επικοινωνίας, δίνοντας την ευκαιρία στον χρήστη να σερφάρει στο ίντερνετ, να λαμβάνει emails, να ακούει ραδιόφωνο, να βλέπει τηλεόραση και βίντεο ή ακόμα και να μιλά στο τηλέφωνο.

Το 2003, το ψυγείο 'Digital Multimedia Side-By-Side' της LG παρουσιάστηκε ως η τελευταία λέξη της τεχνολογίας στην κουζίνα. Είναι εξοπλισμένο με συσκευή αναπαραγωγής MP3 για κατέβασμα κι αναπαραγωγή μουσικής από το διαδίκτυο, διαθέτει εφαρμογές για προβολή καιρού και ειδήσεων, ενώ πέρα από την αποστολή email κειμένου υποστηρίζει και αποστολή βίντεο mail, χρησιμοποιώντας την ενσωματωμένη κάμερα και μικρόφωνο. Επίσης υποστηρίζει την αγορά προϊόντων μέσω διαδικτύου.

Το Electrolux Screenfridge έκανε την επανεμφάνισή του το 2006, εξοπλισμένο πλέον με μια οθόνη αφής 15 ιντσών και εικονικό πληκτρολόγιο. Διαθέτει ευρυζωνική σύνδεση (broadband) και είναι συνδεδεμένο με την τηλεόραση ασύρματα. Πέρα από υποστήριξη διαδικτύου, email, τηλεφώνου, ραδιοφώνου και συσκευής αναπαραγωγής MP3, διαθέτει ένα προηγμένο σύστημα ημερολογίου κι ανταλλαγής μηνυμάτων, ώστε η κουζίνα να γίνεται πραγματικά ο 'κεντρικός κόμβος' (hub) του σπιτιού. Το 2007 εμφανίστηκε επίσης το ψυγείο Whirlpool Centralpark, εξοπλισμένο με δορυφορικό ραδιόφωνο, μια διαδικτυακή ταμπλέτα (web tablet) με διαδραστικό πίνακα μηνυμάτων και οικογενειακό ημερολόγιο, ψηφιακό πλαίσιο εικόνων, συσκευή αναπαραγωγής DVD/CD κι άλλα χαρακτηριστικά [14].

Υπήρξαν πολλές προσπάθειες για τη δημιουργία ενός επιτυχημένου έξυπνου διαδικτυακού ψυγείου. Το 2009 η Samsung προσέφερε ένα ψυγείο με αποσπόμενη LCD οθόνη με πίνακα μηνυμάτων για την έξυπνη διαχείριση των προϊόντων του ψυγείου. Το 2010, η LG πρότεινε ένα ψυγείο το οποίο, πέρα από διάφορες ενέργειες, έδινε τη δυνατότητα στους χρήστες να παραμένουν συνδεδεμένοι στο διαδίκτυο. Το 2011, η Samsung για ακόμα μια φορά πρότεινε ένα νέο έξυπνο ψυγείο, το 'Futuristic RF4289', με ενσωματωμένη οθόνη αφής 8 ιντσών. Μια από τις τελευταίες προσπάθειες είναι το 'LG Home Chat', το οποίο παρουσιάστηκε στην Διεθνή έκθεση ηλεκτρονικών καταναλωτικών προϊόντων (International Consumer Electronics Show - CES) στο Λας Βέγκας το 2014, από την Korea LG. Η πρόταση είναι ότι οι χρήστες θα ρωτάνε το ψυγείο για το περιεχόμενό του [18]. Οι εφαρμογές του LG Home Chat επικοινωνούν με τους χρήστες μέσω της εφαρμογής

Line, η οποία είναι μια δημοφιλής εφαρμογή συνομιλίας στην Ασία κι έχει σχεδιαστεί για να κατανοεί την φυσική γλώσσα εντολών του χρήστη. Το ψυγείο διαθέτει ενσωματωμένη ευρυγώνια κάμερα, η οποία φωτογραφίζει τα περιεχόμενα του ψυγείου κάθε φορά που ανοίγουν και κλείνουν οι πόρτες του. Ως αποτέλεσμα, ο χρήστης μπορεί να αποφασίσει πιο εύκολα τι προϊόντα χρειάζεται κατά την επίσκεψη σε ένα κατάστημα, κοιτώντας τις φωτογραφίες αυτές. Επιπλέον, το λογισμικό 'Freshness Tracker' του ψυγείου παρέχει πληροφορίες σχετικά με τα προϊόντα που έχουν περάσει την ημερομηνία λήξης τους. Ωστόσο, αυτή η λειτουργικότητα απαιτεί από τον χρήστη να εισάγει λεπτομέρειες για το κάθε προϊόν που τοποθετεί στο ψυγείο [19] [12].

Το 2017, η Samsung εισήγαγε στην αγορά το Family Hub Refrigerator, ένα τετράπορτο έξυπνο ψυγείο, το οποίο έχει ενσωματωμένη μεγάλη οθόνη αφής 21,5 ιντσών και διαθέτει τρεις ενσωματωμένες κάμερες που παρέχουν οπτική πρόσβαση στο εσωτερικό του ψυγείου, δυνατότητα φωνητικών εντολών («Bixby»), δυνατότητα ελέγχου αποθεμάτων και δημιουργίας λίστας αγορών, δυνατότητα διαμερισμού σημειώσεων και μέσων όπως οι φωτογραφίες, δυνατότητα σύνδεσης με άλλες συσκευές, όπως τα κινητά τηλέφωνα και, επιπλέον, αποστολή της λίστας αγορών απευθείας σε κατάστημα με άμεση παραλαβή στον ιδιωτικό χώρο του κατόχου και σύνδεση με την οθόνη της τηλεόρασης ή και με εφαρμογές όπως το Spotify, για να μπορεί ο χρήστης να βλέπει τηλεόραση ή να ακούει μουσική αντίστοιχα, την ώρα που βρίσκεται στην κουζίνα του, μέσω φωνητικών εντολών.

Σκοπός του έξυπνου ψυγείου σαν εφεύρεση είναι να κάνει την κουζίνα ζωτικό σημείο του σπιτιού, παρέχοντας υπηρεσίες που διευκολύνουν κι επιταχύνουν διαδικασίες, οικιακες και μη, της καθημερινότητας, προσφέροντας ταυτόχρονα διασκέδαση. Πολλοί έχουν αναφερθεί ιδιαίτερα στο ζήτημα των αγορών και τη συμβολή της συγκεκριμένης τεχνολογίας σε αυτές [3], ενώ άλλοι κάνουν λόγο για τη συμβολή του ευφυούς συστήματος των έξυπνων ψυγείων στην κατανάλωση των φαγητών, την πρόληψη της λήξης τους και κατ' επέκταση της απόρριψής τους, αποσκοπώντας στην εξοικονόμηση χρημάτων και στο να μην πετώνται τρόφιμα στον κάδο απορριμάτων, ενώ θα μπορούσαν να είχαν καταναλωθεί [22]. Τα έξυπνα ψυγεία μπορούν επίσης να χρησιμοποιηθούν για έλεγχο της διατροφής και για την υιοθέτηση ενός πιο υγιεινού τρόπου ζωής [26].

Μελετώντας τις παραπάνω ιδέες και λειτουργίες, η παρούσα εργασία καταπιάνεται με τη δημιουργία μιας τέτοιας εφαρμογής στο πλαίσιο του έξυπνου ψυγείου με λειτουργίες καταγραφής προϊόντων του ψυγείου, με σκοπό την καλύτερη διαχείρησή τους (αποθέματα και ημερομηνίες λήξης), καθώς και με τη δημιουργία συστήματος διαδικτυακών παραγγελιών του χρήστη, επεκτείνοντας τη λειτουργικότητα αυτή με τον σχεδιασμό συστήματος αυτοματοποιημένων παραγγελιών· μια πρωτότυπη προσπάθεια στη μέχρι σήμερα τεχνολογία του έξυπνου ψυγείου. Το όλο σύστημα παραγγελιών φιλοδοξεί να καθιερώσει τις ηλεκτρονικές παραγγελίες σε σουπερμάρκετ στην καθημερινότητα του ανθρώπου, μέσω της συνεργασίας της εφαρμογής με τις ηλεκτρονικές ιστοσελίδες των διάφορων σουπερμάρκετ για την επίτευξη των παραγγελιών.

1.3 Οργάνωση του τόμου

Στο κεφάλαιο 2 παρουσιάζονται απαραίτητες έννοιες, μοντέλα και βοηθητικά εργαλεία για την κατανόηση του χώρου της εφαρμογής και της ανάπτυξής της. Αναλυτική παρουσίαση των κύριων

προβλημάτων που καλείται να επιλύσει η εφαρμογή καθώς κι επεξήγηση των βασικών στοιχείων της γίνεται στο κεφάλαιο 3, ενώ στο κεφάλαιο 4 ακολουθεί λεπτομερής επίδειξη μεθόδων κι αλγορίθμων επίλυσης των προβλημάτων και γενικότερα της ανάπτυξης της εφαρμογής. Ακολουθούν οι τεχνικές λεπτομέρειες ανάπτυξης και σχεδίασης της εφαρμογής και των εργαλείων που χρησιμοποιήθηκαν στην ενότητα 5. Τέλος, στο κεφάλαιο 6 αναφέρονται τα συμπεράσματα και τα τελικά αποτελέσματα της συνολικής προσπάθειας και παρατίθενται ιδέες μελλοντικής επέκτασης της εργασίας.

Κεφάλαιο 2

Εισαγωγικά στοιχεία - Έννοιες κι Εργαλεία

2.1 Εισαγωγή

Πρωτού προχωρήσουμε στην ανάλυση και σχεδίαση της εφαρμογής, είναι αναγκαία η κατανόηση ορισμένων εννοιών κι εργαλείων. Σε αυτό το κεφάλαιο θα μιλήσουμε για την έννοια του **Διαδικτύου των πραγμάτων** (**IoT**), στην οποία βασίζεται η όλη φιλοσοφία του έξυπνου ψυγείου, καθώς και για τη σημασία του όρου **έξυπνης (smart)** συσκευής. Επίσης, θα γίνει ανάλυση της έννοιας **"εφαρμογή δικτύου" (web application)** και του τρόπου λειτουργίας των εφαρμογών αυτών, ενώ τέλος θα ακολουθήσει μια σύντομη επεξήγηση του εργαλείου **django framework**, το οποίο χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής.

2.2 Θεωρητικό υπόβαθρο - Διαδίκτυο των πραγμάτων

2.2.1 Ορισμός

Το **Διαδίκτυο των πραγμάτων** αποτελεί το δίκτυο επικοινωνίας πληθώρας συσκευών, οικιακών συσκευών, αυτοκινήτων καθώς και κάθε αντικειμένου που ενσωματώνει ηλεκτρονικά μέσα, λογισμικό, αισθητήρες και συνδεσιμότητα σε δίκτυο ώστε να επιτρέπεται η σύνδεση και η ανταλλαγή δεδομένων. Απλούστερα, η φιλοσοφία του IoT είναι η σύνδεση όλων των ηλεκτρονικών συσκευών μεταξύ τους (τοπικό δίκτυο) ή με δυνατότητα σύνδεσης στο διαδίκτυο (παγκόσμιο ιστό). Η έννοια "Things" (πράγματα) δεν είναι αυστηρά συνδεδεμένη με ορισμένα προϊόντα. Αναφέρεται σε μία ευρεία ποικιλία συσκευών εντελώς διαφορετικών μεταξύ τους, όπως για παράδειγμα αυτοκίνητα με ενσωματωμένους αισθητήρες, κάμερες, κλιματιστικά, φώτα, συστήματα ασφαλείας, έξυπνα ρολόγια, ακόμα κι αυτοκίνητα των οποίων οι περίπλοκοι αισθητήρες εντοπίζουν αντικείμενα στην πορεία τους. Τα παραπάνω είναι μερικά από τα πολλά προϊόντα αυτής της τεχνολογίας. Βασικό χαρακτηριστικό αυτών των συσκευών είναι η σύνδεσή τους με το διαδίκτυο, έχοντας απώτερο σκοπό την δυνατότητα του χρήστη να τα ελέγχει από έναν υπολογιστή ή κινητό. Ο όρος διαδίκτυο των πραγμάτων αποδόθηκε την δεκαετία του 1990 από τον Kevin Ashton και είναι μία από

τις τρεις κορυφαίες τεχνολογικές εξελίξεις της επόμενης δεκαετίας, μαζί με το κινητό διαδίκτυο (mobileInternet) και την αυτοματοποίηση της εργασίας γνώσης (knowledgework), και αποτελεί το επόμενο μεγάλο βήμα στον χώρο της τεχνολογίας [24].

2.2.2 Λειτουργία και χρησιμότητα

Πολλαπλές συσκευές κι αντικείμενα, τα οποία έχουν ενσωματωμένους αισθητήρες, συνδέονται σε μια πλατφόρμα. Ο ρόλος αυτής της πλατφόρμας είναι να δέχεται δεδομένα από τις διάφορες συσκευές και να μοιράζει την πληροφορία σε εφαρμογές που έχουν δημιουργηθεί για την αντιμετώπιση συγκεκριμένων αναγκών. Οι συσκευές IoT μπορούν να εντοπίσουν ακριβώς ποιες πληροφορίες είναι χρήσιμες και να τις εκμεταλλευτούν κατάλληλα. Η δυνατότητα αυτή μπορεί να αυτοματοποιήσει επαναλαμβανόμενες, χρονοβόρες ή ακόμα κι επικίνδυνες εργασίες, όπως για παράδειγμα η αυτόματη ρύθμιση και λειτουργία μηχανημάτων στα εργοστάσια. Η μετάβαση στην ψηφιακή εποχή είναι γεγονός. Η χρησιμότητα του IoT είναι μεγάλη και η ζήτηση από τους υποψήφιους αγοραστές ακόμα μεγαλύτερη. Ως επί το πλείστον, οι άνθρωποι αποζητούν την αυτονομία σε πολλά πράγματα γύρω τους. Από ένα αυτόματο ξυπνητήρι μέχρι το έξυπνο ψυγείο (σχ. 2.1) που ενημερώνει τον χρήστη για βασικές ελλείψεις, ή ακόμα και την δυνατότητα ενεργοποίησης κλιματισμού πριν ακόμα ο χρήστης εισέλθει στο σπίτι, είναι μερικές από τις δυνατότητες που προσφέρει το IoT. Η χρήση του δεν παραμένει μόνο εκεί, αλλά επεκτείνεται και στις επιχειρήσεις οι οποίες εκμεταλλεύονται την δυνατότητα αποθήκευσης κι επεξεργασίας των δεδομένων από cloud συστήματα [24] [5].

2.2.3 Πεδία εφαρμογών

Το εκτεταμένο σύνολο εφαρμογών για συσκευές IoT συχνά χωρίζεται σε κατηγορίες που αφορούν σκοπούς καταναλωτικούς, εμπορικούς, βιομηχανικούς και υποδομικούς.

1. **Καταναλωτικές εφαρμογές**: Ένα αυξανόμενο τμήμα των συσκευών IoT δημιουργούνται για χρήση από τους καταναλωτές, συμπεριλαμβανομένων των συνδεδεμένων οχημάτων, του αυτοματισμού στο σπίτι, της φορητής τεχνολογίας, της συνδεδεμένης υγείας και των συσκευών με δυνατότητες απομακρυσμένης παρακολούθησης. Ας πάρουμε για παράδειγμα το σενάριο του **έξυπνου σπιτιού**. Οι συσκευές IoT αποτελούν μέρος της ευρύτερης έννοιας του αυτοματισμού στο σπίτι, που μπορεί να περιλαμβάνει φωτισμό, θέρμανση και κλιματισμό, μέσα μαζικής ενημέρωσης και συστήματα ασφαλείας. Τα μακροπρόθεσμα οφέλη θα μπορούσαν να περιλαμβάνουν την εξοικονόμηση ενέργειας με την αυτόματη εξασφάλιση απενεργοποίησης φώτων και ηλεκτρικών συσκευών.

2. **Εμπορικές εφαρμογές**:

    - **Ιατρική και υγειονομική περίθαλψη**: Το Διαδίκτυο των ιατρικών πραγμάτων είναι μια εφαρμογή του IoT για σκοπούς που σχετίζονται με την ιατρική και την υγεία, τη συλλογή κι ανάλυση δεδομένων για έρευνα και παρακολούθηση. Αυτή η ”έξυπνη υγειονομική περίθαλψη”, όπως καλείται επίσης, οδήγησε στη δημιουργία ενός ψηφιακού συστήματος υγειονομικής περίθαλψης, το οποίο θα συνδέει διαθέσιμους ιατρικούς

πόρους και υπηρεσίες υγειονομικής περίθαλψης. Για παράδειγμα, ορισμένα νοσοκομεία έχουν αρχίσει να εφαρμόζουν ”έξυπνα κρεβάτια” που μπορούν να ανιχνεύσουν πότε είναι κατειλημμένα και πότε ένας ασθενής προσπαθεί να σηκωθεί. Μπορούν επίσης να προσαρμοστούν για να εξασφαλίσουν την κατάλληλη πίεση και την υποστήριξη που εφαρμόζεται στον ασθενή χωρίς τη χειρωνακτική αλληλεπίδραση των νοσοκόμων.

- **Συστήματα μεταφορών**: Το IoT μπορεί να βοηθήσει στην ενοποίηση των επικοινωνιών, του ελέγχου και της επεξεργασίας πληροφοριών σε διάφορα συστήματα μεταφορών. Η εφαρμογή του IoT επεκτείνεται σε όλες τις πτυχές των συστημάτων μεταφοράς (δηλαδή του οχήματος, της υποδομής και του οδηγού ή του χρήστη). Η δυναμική αλληλεπίδραση μεταξύ αυτών των στοιχείων ενός συστήματος μεταφορών επιτρέπει την ενδοκοινοτική επικοινωνία, τον έξυπνο έλεγχο της κυκλοφορίας, τον έξυπνο χώρο στάθμευσης, τα ηλεκτρονικά συστήματα είσπραξης διοδίων, την εφοδιαστική και τη διαχείριση του στόλου, τον έλεγχο των οχημάτων, την ασφάλεια και την οδική βοήθεια.

3. **Βιομηχανικές εφαρμογές**:

   - **Βιομηχανία**: Το IoT μπορεί να πραγματοποιήσει την ομαλή ενσωμάτωση διαφόρων κατασκευαστικών συσκευών εξοπλισμένων με δυνατότητες ανίχνευσης, αναγνώρισης, επεξεργασίας, επικοινωνίας, ενεργοποίησης και δικτύωσης. Με βάση έναν τόσο ολοκληρωμένο έξυπνο κυβερνο-φυτικό χώρο, ανοίγει την πόρτα για τη δημιουργία ολόκληρων νέων επιχειρηματικών κι εμπορικών ευκαιριών για την κατασκευή. Ο έλεγχος του δικτύου και η διαχείριση του εξοπλισμού κατασκευής, η διαχείριση περιουσιακών στοιχείων και καταστάσεων ή ο έλεγχος των παραγωγικών διαδικασιών φέρνουν το IoT στο πεδίο των βιομηχανικών εφαρμογών και της έξυπνης κατασκευής. Τα έξυπνα συστήματα IoT επιτρέπουν την ταχεία κατασκευή νέων προϊόντων, τη δυναμική ανταπόκριση στις απαιτήσεις των προϊόντων και τη βελτιστοποίηση σε πραγματικό χρόνο των παραγωγικών δικτύων παραγωγής κι αλυσίδας εφοδιασμού, με τη δικτύωση μηχανημάτων, αισθητήρων και συστημάτων ελέγχου.
   - **Γεωργία**: Υπάρχουν πολλές εφαρμογές του IoT στην γεωργία, όπως η συλλογή δεδομένων σχετικά με τη θερμοκρασία, τις βροχοπτώσεις, την υγρασία, την ταχύτητα του ανέμου, την προσβολή των παρασίτων και το περιεχόμενο του εδάφους. Αυτά τα δεδομένα μπορούν να χρησιμοποιηθούν για την αυτοματοποίηση των γεωργικών τεχνικών, να λάβουν τεκμηριωμένες αποφάσεις για τη βελτίωση της ποιότητας και της ποσότητας, την ελαχιστοποίηση του κινδύνου και των αποβλήτων και τη μείωση της προσπάθειας που απαιτείται για τη διαχείριση των καλλιεργειών. Για παράδειγμα, οι γεωργοί μπορούν τώρα να παρακολουθούν τη θερμοκρασία και την υγρασία του εδάφους από μακριά και να εφαρμόζουν ακόμη και τα δεδομένα που αποκτήθηκαν από το IoT σε προγράμματα λιπασματοποίησης.

4. **Υποδομικές εφαρμογές**: Η παρακολούθηση κι ο έλεγχος των ενεργειών των βιώσιμων αστικών και αγροτικών υποδομών όπως οι γέφυρες, οι σιδηροδρομικές διαδρομές και τα αιολικά

πάρκα αποτελεί βασική εφαρμογή του IoT. Το μοντέλο IoT μπορεί να χρησιμοποιηθεί για την παρακολούθηση τυχόν γεγονότων ή αλλαγών στις δομικές συνθήκες που μπορούν να θέσουν σε κίνδυνο την ασφάλεια και να αυξήσουν τον κίνδυνο. Το IoT μπορεί να ωφελήσει την κατασκευαστική βιομηχανία με εξοικονόμηση κόστους, μείωση χρόνου, καλύτερη ποιότητα εργασίας κι αύξηση της παραγωγικότητας. Μπορεί επίσης να χρησιμοποιηθεί για τον αποτελεσματικό προγραμματισμό των δραστηριοτήτων επισκευής και συντήρησης, με συντονισμό των καθηκόντων μεταξύ διαφορετικών παρόχων υπηρεσιών και χρηστών αυτών των εγκαταστάσεων. Οι συσκευές IoT μπορούν επίσης να χρησιμοποιηθούν για τον έλεγχο κρίσιμης υποδομής, όπως γέφυρες για την παροχή πρόσβασης σε πλοία. Η χρήση συσκευών διαδικτύου για την παρακολούθηση και τη λειτουργία των υποδομών είναι πιθανό να βελτιώσει τη διαχείριση συμβάντων και τον συντονισμό της αντιμετώπισης καταστάσεων έκτακτης ανάγκης, την ποιότητα των υπηρεσιών, την αναβάθμιση και τη μείωση του κόστους λειτουργίας σε όλες τις περιοχές που σχετίζονται με τις υποδομές. Ακόμη και τομείς όπως η διαχείριση των αποβλήτων μπορούν να επωφεληθούν από την αυτοματοποίηση και τη βελτιστοποίηση που προσφέρει το διαδίκτυο των πραγμάτων.

2.2.4 Πλεονεκτήματα

1. **Δεδομένα**: Όσο μεγαλύτερος είναι ο όγκος των πληροφοριών, τόσο πιο εύκολο είναι να παρθεί η σωστή απόφαση. Ως παράδειγμα ας θεωρήσουμε την τεχνολογία του έξυπνου ψυγείου, το οποίο γνωρίζοντας τί πρέπει να αγοραστεί από ένα σούπερμάρκετ ενώ οι άνθρωποι απουσιάζουν, χωρίς να χρειάζεται να το ελέγξουν και να το αποφασίσουν οι ίδιοι, δεν τους εξοικονομεί μόνο χρόνο, αλλά είναι επίσης βολικό.

2. **Παρακολούθηση**: Οι συσκευές παρακολουθούν τόσο την ποιότητα, όσο και τη βιωσιμότητα των αντικειμένων στο σπίτι. Η γνώση της ημερομηνίας λήξης των προϊόντων πριν από την κατανάλωσή τους βελτιώνει την ασφάλεια και την ποιότητα ζωής.

3. **Χρόνος**: Η ποσότητα του χρόνου που εξοικονομείται από την παρακολούθηση των αντικειμένων στο σπίτι είναι πολύ μεγάλη.

4. **Χρήματα**: Η οικονομική πτυχή είναι ένα από τα σημαντικότερα πλεονεκτήματα. Για παράδειγμα, η τεχνολογία του IoT μπορεί να αντικαταστήσει το ανθρώπινο δυναμικό που είναι υπεύθυνο για την παρακολούθηση και τη συντήρηση των προμηθειών μιας επιχείρησης [16].

2.2.5 Προκλήσεις και μειονεκτήματα

1. **Συμβατότητα**: Το μεγαλύτερο ποσοστό των συσκευών που υλοποιούν το σενάριο του διαδικτύου των πραγμάτων έχει ως σκοπό την παρακολούθηση γεγονότων ή αντικειμένων. Μέχρι σήμερα, δεν υπάρχει κάποιος καθολικός τρόπος για την παρακολούθηση με αισθητήρες. Θα ήταν προτιμότερο να υπήρχε μια ενιαία έννοια για τον σκοπό αυτόν, για να αποφεύγονται προβλήματα συμβατότητας.

2. **Αποθήκευση δεδομένων**: Μια πρόκληση για τους παραγωγούς εφαρμογών διαδικτύου είναι να καθαρίζουν, να επεξεργάζονται και να ερμηνεύουν τον τεράστιο όγκο των δεδομένων που συγκεντρώνουν οι αισθητήρες των συσκευών. Υπάρχει μια λύση που προτείνεται για την ανάλυση των πληροφοριών η οποία αναφέρεται ως "ασύρματα δίκτυα αισθητήρων". Αυτά τα δίκτυα μοιράζονται δεδομένα μεταξύ των κόμβων των αισθητήρων που αποστέλλονται σε ένα κατανεμημένο σύστημα για την ανάλυση των αισθητηριακών δεδομένων.

   Μια άλλη πρόκληση είναι η αποθήκευση αυτών των μαζικών δεδομένων. Ανάλογα με την εφαρμογή, ενδέχεται να υπάρχουν υψηλές απαιτήσεις απόκτησης δεδομένων, οι οποίες με τη σειρά τους οδηγούν σε υψηλές απαιτήσεις αποθήκευσης. Επί του παρόντος, το Διαδίκτυο είναι ήδη υπεύθυνο για ένα ποσοστό 5% από τη συνολική ενέργεια που παράγεται και μια "τρομακτική πρόκληση στην ενέργεια" παραμένει ακόμα για τη συλλογή κι αποθήκευση δεδομένων από τις συσκευές IoT.

3. **Προστασία προσωπικών δεδομένων**: Η προστασία προσωπικών δεδομένων είναι η μεγαλύτερη ανησυχία για την υιοθέτηση του Διαδικτύου για την τεχνολογία των πραγμάτων. Ειδικότερα, καθώς το Διαδίκτυο των πραγμάτων εξαπλώνεται ευρέως, οι ηλεκτρονικές επιθέσεις (cyber attacks) είναι πιθανό να γίνουν μια όλο και πιο φυσική (κι όχι απλώς εικονική) απειλή. Ο τρέχων χώρος IoT έρχεται με πολλές αδυναμίες ασφαλείας. Αυτά τα τρωτά σημεία περιλαμβάνουν την αδυναμία ελέγχου πιστοποίησης (συσκευές IoT χρησιμοποιούνται με προεπιλεγμένα διαπιστευτήρια-credentials), μη κρυπτογραφημένα μηνύματα που αποστέλλονται μεταξύ συσκευών, ενέσεις SQL (SQL injections) κι έλλειψη επαλήθευσης ή κρυπτογράφησης ενημερώσεων λογισμικού. Αυτό επιτρέπει στους επιτιθέμενους να παρακολουθούν εύκολα τα δεδομένα για να συλλέγουν προσωπικά δεδομένα, να κλέβουν τα διαπιστευτήρια του χρήστη κατά τη σύνδεση ή να εισάγουν κακόβουλα προγράμματα σε πρόσφατα ενημερωμένο υλικολογισμικό.

4. **Ασφάλεια**: Τα συστήματα IoT ελέγχονται συνήθως από έξυπνες εφαρμογές που βασίζονται σε συμβάντα και λαμβάνουν ως είσοδο είτε δεδομένα που ανιχνεύονται, εντολές χρηστών, είτε άλλες ειδοποιήσεις για γεγονότα (από το Διαδίκτυο) κι εντοπίζουν έναν ή περισσότερους ενεργοποιητές για την παροχή διαφορετικών μορφών αυτοματισμού και υπηρεσιών. Παραδείγματα αισθητήρων περιλαμβάνουν ανιχνευτές καπνού, αισθητήρες κίνησης κι αισθητήρες επαφής. Παραδείγματα ενεργοποιητών περιλαμβάνουν έξυπνες κλειδαριές, έξυπνες πρίζες ισχύος και χειριστήρια πόρτας. Οι δημοφιλείς πλατφόρμες ελέγχου, στις οποίες οι προγραμματιστές τρίτων κατασκευαστών μπορούν να δημιουργήσουν έξυπνες εφαρμογές που αλληλεπιδρούν ασύρματα με αυτούς τους αισθητήρες κι ενεργοποιητές, περιλαμβάνουν τα SmartThings της Samsung, το HomeKit της Apple και το Alexa του Amazon, μεταξύ άλλων.

   Ένα συγκεκριμένο πρόβλημα στα συστήματα IoT είναι ότι οι ελαττωματικές εφαρμογές, οι απρόβλεπτες αλληλεπιδράσεις μιας εφαρμογής ή οι αποτυχίες συσκευών/επικοινωνίας μπορεί να προκαλέσουν ανασφαλείς κι επικίνδυνες φυσικές καταστάσεις, π.χ. "ξεκλειδώστε την πόρτα εισόδου όταν κανείς δεν είναι στο σπίτι" ή "απενεργοποιήστε τον θερμαντήρα

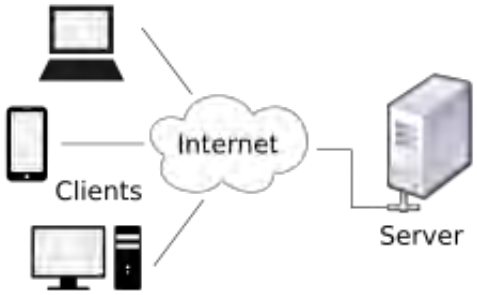
Σχήμα 2.1: Έξυπνο ψυγείο (Πηγή: [24])

όταν η θερμοκρασία είναι κάτω από τους 0 βαθμούς Κελσίου και οι άνθρωποι κοιμούνται τη νύχτα". Η ανίχνευση ατελειών που οδηγούν σε τέτοιες καταστάσεις απαιτεί μια ολιστική άποψη των εγκατεστημένων εφαρμογών, των συσκευών, των διαμορφώσεων τους και, το πιο σημαντικό, του τρόπου αλληλεπίδρασής τους [15].

2.2.6 Έξυπνες συσκευές

Μια **έξυπνη συσκευή** είναι μια ηλεκτρονική συσκευή, η οποία συνδέεται γενικά με άλλες συσκευές ή δίκτυα μέσω διαφορετικών ασύρματων πρωτοκόλλων όπως Bluetooth, NFC, Wi-Fi, LiFi, 3G κ.λπ. που μπορούν να λειτουργήσουν σε κάποιο βαθμό αλληλεπιδραστικά κι αυτόνομα. Αρκετά σημαντικοί τύποι έξυπνων συσκευών είναι τα smartphones, έξυπνα αυτοκίνητα, έξυπνοι θερμοστάτες, έξυπνα κουδούνια, έξυπνες κλειδαριές, έξυπνα ψυγεία, phablets και tablets, έξυπνα ρολόγια κι άλλα. Ο όρος μπορεί επίσης να αναφέρεται σε μια συσκευή που παρουσιάζει μερικές ιδιότητες πανταχού παρουσίας, συμπεριλαμβανομένης - αν και όχι απαραίτητα - τεχνητής νοημοσύνης.

Οι έξυπνες συσκευές μπορούν να σχεδιαστούν για να υποστηρίξουν διάφορους παράγοντες μορφής (form factors), μια ποικιλία ιδιοτήτων που σχετίζονται με την πανταχού παρούσα πληροφορική, και να χρησιμοποιηθούν σε τρία βασικά περιβάλλοντα συστήματος: φυσικό κόσμο, περιβάλλοντα ανθρωποκεντρικά και κατανεμημένα περιβάλλοντα υπολογιστών [23].



Σχήμα 2.2: Μοντέλο πελάτη-διακομιστή (Πηγή: [6])

2.3 Η εφαρμογή δικτύου ως έννοια

Διαδικτυακή εφαρμογή ονομάζεται κάθε εφαρμογή που είναι διαθέσιμη στους χρήστες της μέσω του Διαδικτύου (Internet) ή του ενδοδικτύου (Intranet) μιας εταιρίας, όπου ο χρήστης χρειάζεται μόνο τον φυλλομετρητή του (web browser) για να την χρησιμοποιήσει. Οι εφαρμογές αυτές συνήθως εκτελούνται σε ισχυρές υπολογιστικές μηχανές, οι οποίες έχουν τον ρόλο του σταθμού διακομιστή (server) και παρέχουν τις υπηρεσίες τους σε περισσότερους του ενός χρήστες [30]. Στον κόσμο των υπολογιστών, μια εφαρμογή δικτύου είναι ένα πρόγραμμα επικοινωνίας πελάτη-διακομιστή (client-server), όπου ο πελάτης κι όλο το περιβάλλον του χρήστη (user interface) "τρέχει" στον φυλλομετρητή. Για να γίνει κατανοητή η έννοια της εφαρμογής δικτύου, θα αναλύσουμε συνοπτικά το μοντέλο πελάτη-διακομιστή, το ρόλο και την επικοινωνία μεταξύ πελάτη και διακομιστή [27].

2.3.1 Μοντέλο, ρόλος κι επικοινωνία πελάτη-διακομιστή

Το μοντέλο πελάτη-διακομιστή (Σχ. 2.2) είναι μια κατανεμημένη δομή εφαρμογής που διαχωρίζει τις εργασίες ή τους φόρτους εργασίας μεταξύ των παρόχων ενός πόρου ή υπηρεσίας, που ονομάζονται διακομιστές (ή εξυπηρετητές) και των αιτούντων υπηρεσίας, που καλούνται πελάτες. Συχνά, οι πελάτες και οι διακομιστές επικοινωνούν μεταξύ τους μέσω ενός δικτύου υπολογιστών σε ξεχωριστό υλικό (hardware), αλλά μπορεί να βρίσκονται και στο ίδιο σύστημα. Ένας κεντρικός υπολογιστής διακομιστή εκτελεί ένα ή περισσότερα προγράμματα εξυπηρετητών που μοιράζονται τους πόρους τους με τους πελάτες. Ένας πελάτης δεν μοιράζεται κανέναν από τους πόρους του, αλλά ζητά τη λειτουργία ενός περιεχομένου ή μιας υπηρεσίας ενός διακομιστή. Συνεπώς, οι πελάτες ξεκινούν συνεδρίες επικοινωνίας με εξυπηρετητές που περιμένουν εισερχόμενες αιτήσεις. Παραδείγματα εφαρμογών υπολογιστή που χρησιμοποιούν το μοντέλο πελάτη-διακομιστή είναι το ηλεκτρονικό ταχυδρομείο (Email) κι ο Παγκόσμιος Ιστός.

Συνεχίζουμε στον ρόλο του πελάτη-διακομιστή, όπου κύριο χαρακτηριστικό είναι η σχέση των συνεργαζόμενων προγραμμάτων σε μια εφαρμογή. Η πλευρά του διακομιστή παρέχει μια συνάρτηση ή μια υπηρεσία σε έναν ή πολλούς πελάτες, οι οποίοι εκκινούν αιτήματα για τέτοιες υπηρεσίες. Οι εξυπηρετητές ξεχωρίζουν ανάλογα με τις υπηρεσίες που παρέχουν. Για παράδειγμα, ένας διακομιστής ιστού (web server) εξυπηρετεί ιστοσελίδες (web pages) κι ένας διακομιστής αρχείων

(file server) εξυπηρετεί αρχεία υπολογιστή. Ένας κοινός πόρος μπορεί να είναι οποιοδήποτε από τα λογισμικά κι ηλεκτρονικά στοιχεία του υπολογιστή του εξυπηρετητή, από προγράμματα και δεδομένα έως επεξεργαστές και συσκευές αποθήκευσης. Η κοινή χρήση πόρων ενός διακομιστή αποτελεί υπηρεσία. Το αν ένας υπολογιστής είναι πελάτης, εξυπηρετητής, ή και τα δύο, εξαρτάται από τη φύση της εφαρμογής που απαιτεί τις λειτουργίες της υπηρεσίας. Για παράδειγμα, ένας μόνο υπολογιστής μπορεί να τρέξει συγχρόνως λογισμικό διακομιστή ιστού και διακομιστή αρχείων για να εξυπηρετήσει διαφορετικά δεδομένα σε πελάτες που κάνουν διαφορετικά είδη αιτημάτων. Το λογισμικό πελάτη μπορεί επίσης να επικοινωνεί με λογισμικό εξυπηρετητή στον ίδιο υπολογιστή. Κάποιες φορές καλείται επικοινωνία μεταξύ εξυπηρετητών, όπως συγχρονισμός δεδομένων ή επικοινωνία διακομιστή-διακομιστή.

Τέλος, θα μιλήσουμε για την επικοινωνία πελάτη-διακομιστή. Γενικά, μια υπηρεσία είναι σε αφαιρετικό επίπεδο κάποιος από τους πόρους του υπολογιστή κι ένας πελάτης δεν χρειάζεται να ασχολείται ή να γνωρίζει τον τρόπο με τον οποίο ο εξυπηρετητής εκτελεί την εκπλήρωση του αιτήματος (request) και την παράδοση της απάντησης (response). Ο πελάτης πρέπει να κατανοήσει μόνο την απάντηση που βασίζεται στο γνωστό πρωτόκολλο εφαρμογής (application protocol), δηλαδή το περιεχόμενο και τη μορφή των δεδομένων για την ζητούμενη υπηρεσία. Οι πελάτες και οι διακομιστές ανταλλάσσουν μηνύματα σε ένα πρότυπο μηνυμάτων αίτησης-απάντησης (request-response). Ο πελάτης στέλνει ένα αίτημα κι ο εξυπηρετητής επιστρέφει μια απάντηση. Αυτή η ανταλλαγή μηνυμάτων είναι ένα παράδειγμα επικοινωνίας μεταξύ των διαδικασιών (inter-process communication ή IPC). Για να επικοινωνούν οι υπολογιστές πρέπει να έχουν μια κοινή γλώσσα και πρέπει να ακολουθούν κανόνες, έτσι ώστε τόσο ο πελάτης όσο και ο εξυπηρετητής να γνωρίζουν τί να περιμένουν. Η γλώσσα και οι κανόνες επικοινωνίας ορίζονται σε ένα πρωτόκολλο επικοινωνιών (communication protocol). Όλα τα πρωτόκολλα πελάτη-εξυπηρετητή λειτουργούν στο επίπεδο εφαρμογής.

Το πρωτόκολλο επιπέδου εφαρμογής (application layer protocol) ορίζει τα βασικά πρότυπα του διαλόγου. Για να επισημοποιήσει ακόμη περισσότερο την ανταλλαγή δεδομένων, ο εξυπηρετητής μπορεί να εφαρμόσει μια διεπαφή προγραμματισμού εφαρμογών (API).Το API είναι ένα επίπεδο αφαίρεσης (abstraction layer) για την πρόσβαση σε μια υπηρεσία. Περιορίζοντας την επικοινωνία σε μια συγκεκριμένη μορφή περιεχομένου, διευκολύνεται η ανάλυση. Απελευθερώνοντας την πρόσβαση, διευκολύνεται την ανταλλαγή δεδομένων μεταξύ των διαφορετικών πλατφορμών.

Ένας εξυπηρετητής ενδέχεται να λάβει αιτήματα από πολλούς διαφορετικούς πελάτες σε σύντομο χρονικό διάστημα. Ένας υπολογιστής μπορεί να εκτελέσει μόνο ένα περιορισμένο αριθμό εργασιών ανά πάσα στιγμή και βασίζεται σε ένα σύστημα προγραμματισμού για να δώσει προτεραιότητα στις εισερχόμενες αιτήσεις από πελάτες για να τους φιλοξενήσει. Για να αποτρέψει την κατάχρηση και τη μεγιστοποίηση της διαθεσιμότητας, το λογισμικό εξυπηρετητή ενδέχεται να περιορίσει τη διαθεσιμότητα σε πελάτες. Οι επιθέσεις άρνησης εξυπηρέτησης (DoS attacks) έχουν σχεδιαστεί για να εκμεταλλεύονται την υποχρέωση ενός διακομιστή να επεξεργάζεται τα αιτήματα υπερφορτωμένα με υπερβολικά ποσοστά αιτημάτων [6].

2.3.2 Πώς δουλεύει το διαδίκτυο

Αρχικά θα αναλύσουμε τι συμβαίνει όταν ο χρήστης πληκτρολογεί στον φυλλομετρητή μια ηλεκτρονική διεύθυνση (URL) και πατά αναζήτηση. Κάθε ιστοσελίδα μεταδίδεται στον φυλλομετρητή του χρήστη ως HTML, μια γλώσσα προγραμματισμού που χρησιμοποιείται από τους περιηγητές για να περιγράψει το περιεχόμενο και τη δομή μιας ιστοσελίδας. Η εφαρμογή που είναι υπεύθυνη να στείλει το HTML αρχείο στον φυλλομετρητή είναι ο διακομιστής που αναφέραμε προηγουμένως. Αυτός είναι μάλιστα και ο τελικός σκοπός μιας διαδικτυακής εφαρμογής, να στείλει HTML στον περιηγητή, ανεξάρτητα από το πόσο πολύπλοκη είναι η λογική της εφαρμογής. Πώς γνωρίζει η εφαρμογή τί πρέπει να στείλει στον περιηγητή; Στέλνει ό,τι ζητήσει ο ίδιος (αίτημα).

2.3.3 Πρωτόκολλο HTTP

Οι φυλλομετρητές "κατεβάζουν" ιστοσελίδες από τους διακομιστές δικτύου χρησιμοποιώντας το πρωτόκολλο HTTP (ένα πρωτόκολλο, στον τομέα του προγραμματισμού, είναι μια καθολικά γνωστή μορφή δεδομένων κι ακολουθία βημάτων που επιτρέπουν την επικοινωνία μεταξύ δυο πλευρών). Το πρωτόκολλο HTTP βασίζεται στο μοντέλο αίτημα-απάντηση.

Κάθε μήνυμα στο HTTP πρωτόκολλο έχει μια σχετική μέθοδο. Οι διάφορες HTTP μέθοδοι αντιστοιχούν σε λογικά διαφορετικούς τύπους αιτημάτων που μπορεί να στείλει ο πελάτης, τα οποία αιτήματα με τη σειρά τους αντιπροσωπεύουν διαφορετικές προθέσεις στην πλευρά του πελάτη. Δύο από τις κυριότερες και πιο χρησιμοποιούμενες μεθόδους είναι η GET και η POST. Η μέθοδος GET παίρνει (ζητάει) δεδομένα από τον διακομιστή δικτύου και είναι το πιο σύνηθες αίτημα. Η GET δεν θα πρέπει να κάνει τίποτε περισσότερο από το να σερβίρει την απάντηση με το HTML περιεχόμενο της σελίδας που ζητήθηκε, δηλαδή δεν θα πρέπει να αλλάξει την κατάσταση της εφαρμογής (για παράδειγμα, να δημιουργηθεί ένας λογαριασμός χρήστη με χρήση της μεθόδου GET). Από την άλλη, η μέθοδος POST έχει περισσότερη αλληλεπίδραση με την εφαρμογή. Με ένα POST αίτημα, ο χρήστης μπορεί να στείλει δεδομένα στην εφαρμογή, για παράδειγμα μέσω της συμπλήρωσης μιας φόρμας, για την εγγραφή χρήστη σε έναν ιστότοπο. Σε αντίθεση με την GET, οι αιτήσεις POST επιφέρουν αλλαγές στην κατάσταση της εφαρμογής και δεν οδηγούν πάντα σε μια νέα σελίδα HTML που αποστέλεται στον πελάτη.

Μια εφαρμογή web λοιπόν είναι υπεύθυνη για τη λήψη ενός HTTP αιτήματος (όπως GET, POST) και για την απάντηση με μια HTTP απάντηση, που συνήθως περιέχει HTML που αντιπροσωπεύει τη ζητούμενη σελίδα. Κάθε απάντηση έχει κι έναν κωδικό απόκρισης HTTP (HTTP response code). Για παράδειγμα, αν όλα πάνε καλά, ο εξυπηρετητής επιστρέφει έναν κωδικό 200, ο οποίος σημαίνει OK. Οι κωδικοί απόκρισης είναι πάντα τριψήφιοι αριθμοί. Ένα άλλο συνηθισμένο παράδειγμα κωδικού απόκρισης είναι αν δεν βρεθεί η σελίδα την οποία αιτήθηκε ο πελάτης, οπότε επιστρέφεται κωδικός 404.

2.3.4 Η δουλειά του διακομιστή δικτύου

Αν επρόκειτο να συνεχίσουμε να βασιζόμαστε στο παραπάνω παράδειγμα ως βάση για μια εφαρμογή στο διαδίκτυο, υπάρχουν αρκετά προβλήματα που θα έπρεπε να επιλύσουμε:

1. Πώς ελέγχουμε την ζητούμενη διεύθυνση URL κι επιστρέφουμε την κατάλληλη σελίδα ως απάντηση;
2. Πώς αντιμετωπίζουμε τα POST αιτήματα, πέρα από τα απλά αιτήματα;
3. Πώς αντιμετωπίζουμε έννοιες όπως περιόδους σύνδεσης και cookies;
4. Πώς μπορούμε να κλιμακώσουμε την εφαρμογή για να χειριζόμαστε πολλές ταυτόχρονες συνδέσεις;

Όπως μπορείτε να φανταστείτε, κανείς δεν θέλει να λύσει αυτά τα προβλήματα κάθε φορά που χτίζει μια εφαρμογή δικτύου. Για το λόγο αυτό, υπάρχουν πακέτα τα οποία χειρίζονται τις λεπτομέρειες του HTTP πρωτοκόλλου κι έχουν λογικές λύσεις στα ανωτέρω προβλήματα. Λάβετε υπόψη, ωστόσο, ότι στον πυρήνα τους λειτουργούν με τον ίδιο τρόπο όπως το παράδειγμά μας: ακρόαση αιτημάτων κι αποστολή HTTP απαντήσεων με κάποια επιστροφή HTML.

2.3.5 Πλαίσιο Δικτύου

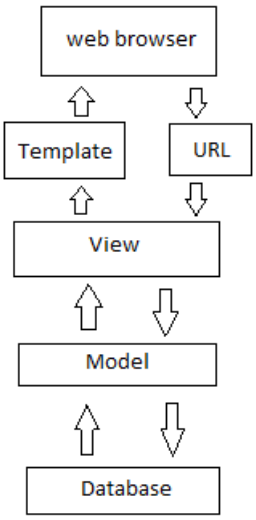
Ένα πλαίσιο δικτύου (web framework) είναι ένα λογισμικό το οποίο έχει σχεδιαστεί για τη βοήθεια της ανάπτυξης διαδικτυακών εφαρμογών. Από όλα τα θέματα γύρω από την οικοδόμηση μιας διαδικτυακής εφαρμογής, δυο είναι αυτά που ξεχωρίζουν:

1. Πώς αντιστοιχίζουμε μια ζητούμενη διεύθυνση URL στον κώδικα που προορίζεται να τη χειριστεί;
2. Πώς δημιουργούμε δυναμικά τη ζητούμενη HTML, εισάγοντας υπολογισμένες τιμές ή πληροφορίες που ανακτήθηκαν από μια βάση δεδομένων;

Κάθε πλαίσιο δικτύου λύνει αυτά τα θέματα με κάποιον τρόπο και υπάρχουν πολλές διαφορετικές προσεγγίσεις [28].

2.4 Πλαίσιο δικτύου Django

Παρότι υπάρχουν διάφορα άλλα λογισμικά ως επιλογές, εμείς θα ασχοληθούμε με το πλαίσιο Django. Κύριο χαρακτηριστικό του django είναι ότι χρησιμοποιεί το μοτίβο MVC κι απαιτεί ο κώδικας που χρησιμοποιεί το πλαίσιο να κάνει το ίδιο. MVC (Model-View-Controller) είναι απλά ένας τρόπος για τον λογικό διαχωρισμό των διαφορετικών ευθυνών της εφαρμογής. Πόροι όπως οι πίνακες της βάσης δεδομένων αντιπροσωπεύονται από μοντέλα, με τον ίδιο ακριβώς τρόπο που μια κλάση (class) στην Python συχνά διαμορφώνει κάποιο αντικείμενο πραγματικού κόσμου. Οι controllers περιέχουν όλη την επιχειρηματική λογική (business logic) της εφαρμογής και λειτουργούν πάνω σε μοντέλα. Τα views δέχονται όλες τις πληροφορίες που χρειάζονται για τη δυναμική δημιουργία της αναπαράστασης HTML κώδικα της σελίδας. Να επισημάνουμε σε αυτό το σημείο ότι, παραδόξως, στο django, οι controllers καλούνται views, και τα views καλούνται templates. Το πιο απλό αλλά και βασικό σενάριο πορείας μιας αίτησης από τον φυλλομετρητή στον διακομιστή, όσον αφορά το django, που πρέπει να κατανοήσει ο αναγνώστης απεικονίζεται στο σχήμα 2.3.



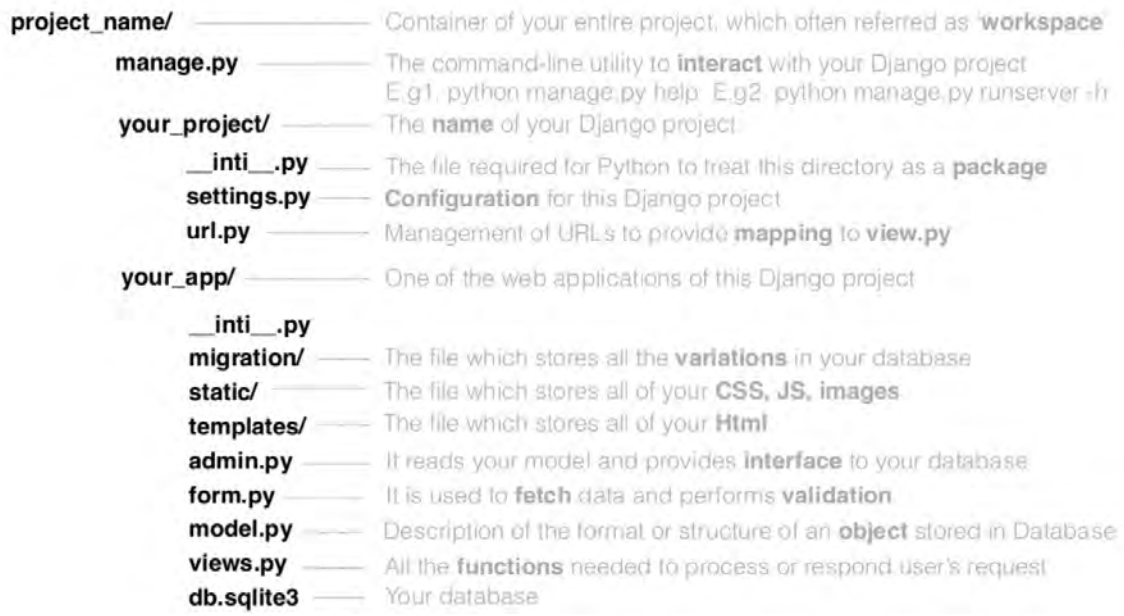
Σχήμα 2.3: Βασική ροή εργασίας στο Django

2.4.1 Python

Το django βασίζεται στη γλώσσα προγραμματισμού python. Η Python είναι διερμηνευόμενη (interprited), γενικού σκοπού (general-purpose) κι υψηλού επιπέδου, γλώσσα προγραμματισμού. Ανήκει στις γλώσσες προστακτικού προγραμματισμού (Imperative programming) κι υποστηρίζει τόσο το διαδικαστικό (procedural programming) όσο και το αντικειμενοστρεφές (object-oriented programming) προγραμματιστικό υπόδειγμα (programming paradigm). Είναι δυναμική γλώσσα προγραμματισμού (dynamically typed) και υποστηρίζει συλλογή απορριμμάτων (garbage collection ή GC). Ο κύριος στόχος της είναι η αναγνωσιμότητα του κώδικά της και η ευκολία χρήσης της. Το συντακτικό της επιτρέπει στους προγραμματιστές να εκφράσουν έννοιες σε λιγότερες γραμμές κώδικα από ότι θα ήταν δυνατόν σε γλώσσες όπως η C++ ή η Java. Διακρίνεται λόγω του ότι έχει πολλές βιβλιοθήκες που διευκολύνουν ιδιαίτερα αρκετές συνηθισμένες εργασίες καθώς και για την ταχύτητα εκμάθησης της. Οι διερμηνευτές της Python είναι διαθέσιμοι για εγκατάσταση σε πολλά λειτουργικά συστήματα, επιτρέποντας στην Python την εκτέλεση κώδικα σε ευρεία γκάμα συστημάτων. Χρησιμοποιώντας εργαλεία τρίτων, όπως το Py2exe ή το Pyinstaller, ο κώδικας της Python μπορεί να πακεταριστεί σε αυτόνομα εκτελέσιμα προγράμματα για μερικά από τα πιο δημοφιλή λειτουργικά συστήματα, επιτρέποντας τη διανομή του βασισμένου σε Python λογισμικού για χρήση σε αυτά τα περιβάλλοντα χωρίς να απαιτείται εγκατάσταση του διερμηνευτή της Python. [20]

2.4.2 Ροή εργασίας

Ας επανέλθουμε στο σχήμα 2.3 για να το μελετήσουμε με λεπτομέρεια. Καταρχάς, ο ρόλος του **φυλλομετρητή**, όπως συζητήσαμε και προηγουμένως, είναι να στείλει στον εξυπηρετητή την αίτηση του χρήστη και να του επιστρέψει την απάντηση που θα στείλει ο εξυπηρετητής, σύμφωνα με



Σχήμα 2.4: Δομή καταλόγου Django. Πηγή: [9]

το αίτημα. Το **URL**, δηλαδή η ηλεκτρονική διεύθυνση, παρέχει μια αντιστοίχηση του αιτήματος με ένα **view** (ή αλλιώς view function). Δηλαδή, όταν έρχεται ένα νέο αίτημα, αυτό αντιστοιχίζεται σε ένα view για επεξεργασία και παραγωγή απάντησης. Πιο αναλυτικά, τα django **views** είναι μεσίτες πληροφοριών μιας εφαρμογής. Δέχονται δεδομένα από τα μοντέλα και τα μεταφέρουν σε κάποιο template. Το view αποφασίζει σχετικά με τα δεδομένα που παρέχονται στο template, είτε ενεργώντας με είσοδο πληροφορίας (input) από τον χρήστη, είτε ως απάντηση σε άλλες διαδικασίες επιχειρηματικής λογικής.

Το **template** περιέχει όλα τα HTML αρχεία, δηλαδή περιέχει το τελικό αποτέλεσμα που θα δει ο χρήστης στον φυλλομετρητή του, σαν απάντηση στο αίτημα που έκανε. Ο χρήστης στέλνει ένα αίτημα, γίνεται η επεξεργασία του σε κάποιο view, και το view στέλνει την πληροφορία στο template για να γίνει η απόδοση (rendering) της σελίδας.

Ένα **μοντέλο** είναι η μοναδική κι οριστική πηγή πληροφοριών για τα δεδομένα μας. Περιέχει τα βασικά πεδία και τις συμπεριφορές των δεδομένων που αποθηκεύουμε ή που σερβίρουμε. Γενικά, κάθε μοντέλο αντιστοιχεί σε έναν ενιαίο πίνακα βάσης δεδομένων. Ένα ερώτημα (query) σε ένα μοντέλο, πηγαίνει κατευθείαν στη βάση δεδομένων (**database**) για την εκτέλεση ενεργειών που επιθυμούμε (create, update, delete, κτλ). [7] [8]

Τα βασικά:

- Κάθε μοντέλο είναι μια κλάση Python.
- Κάθε ιδιότητα του μοντέλου αντιπροσωπεύει ένα πεδίο βάσης δεδομένων.
- Συνδυάζοντας τα παραπάνω, το django μας παρέχει αυτόματα ένα API πρόσβασης στη βάση δεδομένων.

Κεφάλαιο 3

Τα βασικά μέρη της εφαρμογής

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναλύσουμε τα βασικότερα μέρη της εφαρμογής και τα προβλήματα που καλείται να επιλύσει. Θα μιλήσουμε για την ανάγκη ύπαρξης μιας βάσης δεδομένων για ανάκτηση, αποθήκευση κι επεξεργασία δεδομένων, ικανή να ανταποκριθεί στις ανάγκες της λειτουργικότητας της εφαρμογής, την ανάγκη ύπαρξης συστήματος λογαριασμών χρηστών (user accounts), συστήματος καταγραφής προϊόντων του ψυγείου, συστήματος δημιουργίας κι αποστολής ηλεκτρονικών παραγγελιών σε σουπερμάρκετ, καθώς και την αναγκαιότητα της ύπαρξης λίστας επιθυμιών στο αλγοριθμικό κομμάτι.

3.2 Βάση Δεδομένων

Πριν τον σχεδιασμό της εφαρμογής, ο προγραμματιστής καλείται να δημιουργήσει ένα σχήμα βάσης δεδομένων (database) το οποίο μπορεί να καλύψει τη λειτουργικότητα που επιθυμεί. Με τον όρο βάση δεδομένων εννοείται μία συλλογή από συστηματικά μορφοποιημένα σχετιζόμενα δεδομένα στα οποία είναι δυνατή η ανάκτηση δεδομένων μέσω αναζήτησης κατ' απαίτηση. [29] Τα κύρια σημεία που πρέπει να εστιάσουμε, τα οποία απαιτούν αποθήκευση και διαχείριση δεδομένων, είναι:

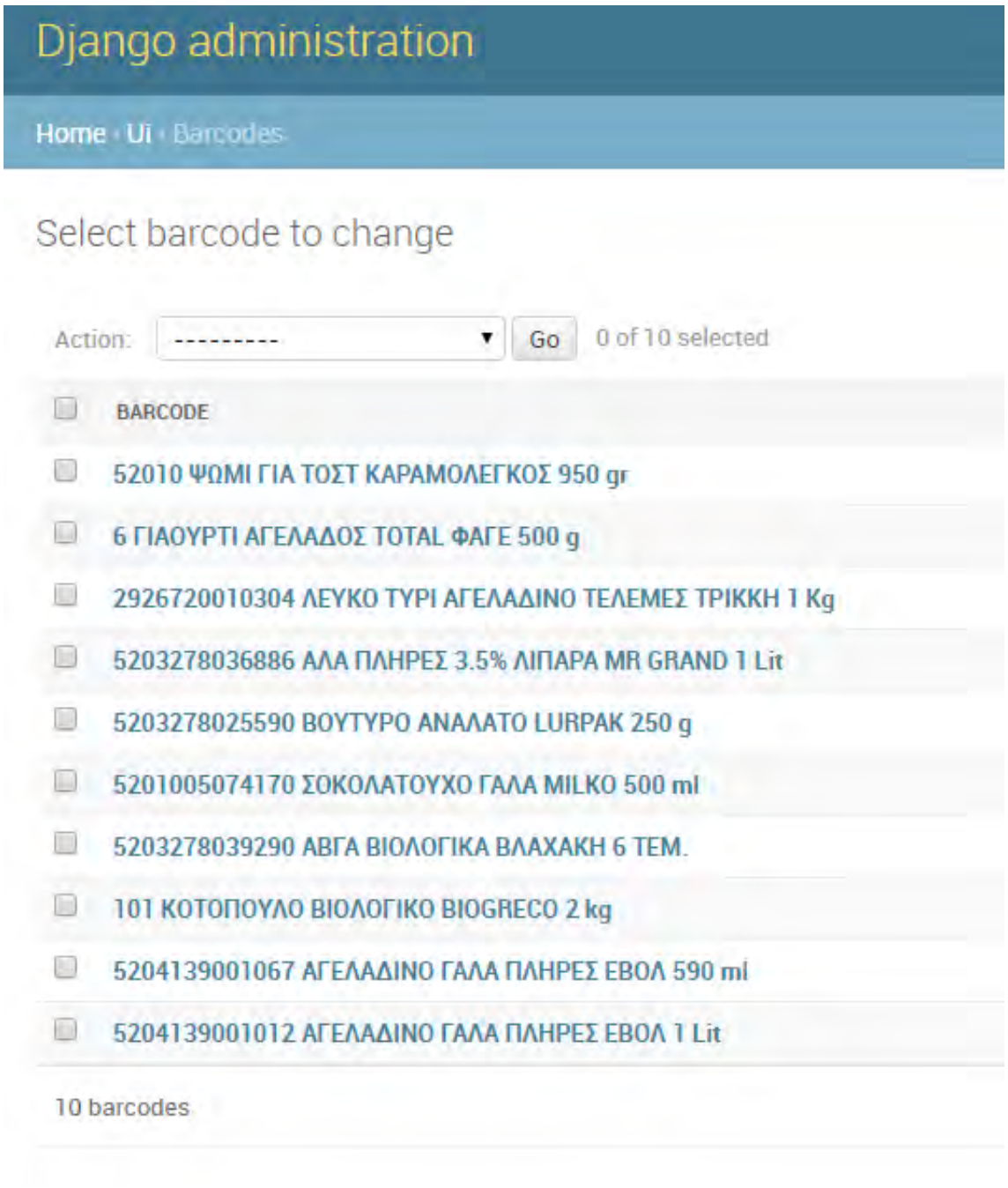
- λογαριασμοί χρηστών της εφαρμογής δικτύου
- λίστα με τα προϊόντα που βρίσκονται μέσα στο ψυγείο
- λίστα με τα προϊόντα που βρίσκονται στη λίστα επιθυμιών του χρήστη
- οι παραγγελίες που δημιουργεί ο χρήστης και τα προϊόντα που περιέχει η κάθε μια
- η αυτόματη παραγγελία που δημιουργείται για τον χρήστη και τα προϊόντα που περιέχονται σε αυτήν
- το ιστορικό παρακολούθησης παραγγελιών.

3.2.1 Αρχικοποίηση της βάσης δεδομένων

Για τη λειτουργία της εφαρμογής είναι απαραίτητη η ύπαρξη δεδομένων τα οποια χρησιμοποιεί ο χρήστης για την ικανοποίηση των διάφορων αναγκών του μέσω της λειτουργικότητας της εφαρμογής. Ας πάρουμε ως παράδειγμα την ιστοσελίδα του amazon [2], η λειτουργικότητα της οποίας μπορεί να παρομοιαστεί με το κομμάτι των ηλεκτρονικών παραγγελιών της εφαρμογής μας. Ο χρήστης του amazon μπορεί να επιλέξει μεταξύ διάφορων χιλιάδων προϊόντων αυτά που τον ενδιαφέρουν και να τα χρησιμοποιήσει σε ορισμένες λειτουργίες (αγορά, προσθήκη στα αγαπημένα, κτλ). Όλα αυτά τα προϊόντα υπάρχουν στη βάση δεδομένων του amazon. Αυτό λοιπόν είναι απαραίτητο και στο δικό μας σενάριο, να υπάρχει μια βάση δεδομένων αρχικοποιημένη με προϊόντα έτοιμα για χρήση, για την ικανοποίηση των αναγκών του χρήστη, όπως για παράδειγμα την εισαγωγή ενός προϊόντος σε μια λίστα παραγγελίας. Συγκεκριμένα, κατά τη σχεδίαση της εφαρμογής έγινε χειροκίνητη αρχικοποίηση της βάσης δεδομένων με διάφορα προϊόντα (σχ. 3.1), για σκοπούς ανάπτυξης (development) και δοκιμής (testing) κι όχι για παραγωγή (production). Ο τρόπος που πραγματικά θέλουμε να γίνεται η αρχικοποίηση της βάσης είναι μέσω της τροφοδοσίας δεδομένων (μόνο όσων μας ενδιαφέρουν) από τη βάση δεδομένων του συνεργαζόμενου σουπερμάρκετ. Δηλαδή χρειάζεται μια "επικοινωνία" μεταξύ της βάσης της εφαρμογής μας και της βάσης του σουπερμάρκετ. Με τον τρόπο αυτόν, η εφαρμογή έχει έγκυρα δεδομένα (προϊόντα) χωρίς τη χρονοβόρα διαδικασία αρχικοποίησης της βάσης, προσθέτοντας χειροκίνητα τα προϊόντα, ένα τη φορά. Επίσης, η διαχείριση της βάσης είναι πιο εύκολη, καθώς κάθε αλλαγή στη βάση δεδομένων του σουπερμάρκετ εφαρμόζεται και στη βάση της εφαρμογής (προσθήκη/διαγραφή/επεξεργασία προϊόντος). Για να επιτευχθούν τα παραπάνω, χρειάζεται η κατάλληλη συνεργασία μεταξύ των δυο πλευρών (διαχειριστή εφαρμογής και σουπερμάρκετ). Επισημαίνεται ότι η βάση της εφαρμογής δεν αποτελεί αντιγραφή της βάσης του εκάστοτε σουπερμάρκετ. Η εφαρμογή διαθέτει το δικό της σχήμα βάσης κι αξιοποιεί κατάλληλα τα δεδομένα που τροφοδοτείται από τη βάση δεδομένων του σουπερμάρκετ.

3.2.2 Κατηγοριοποίηση των διαθέσιμων προϊόντων

Για τη διευκόλυνση της περιήγησης και για λόγους χρηστικούς, η εφαρμογή κατηγοριοποιεί όλα τα διαθέσιμα προϊόντα που υπάρχουν στη βάση δεδομένων, ανάλογα με το είδος τους. Ο χρήστης, πατώντας το αναπτυσσόμενο κουμπί `Categories`, μπορεί είτε να περιηγηθεί στη σελίδα με όλες τις υπάρχουσες κατηγορίες (σχ. 3.2α΄) πατώντας το κουμπί `Categories`, είτε να διαλεξεί απευθείας μια κατηγορία, πατώντας το κουμπί με το όνομά της στο αναπτυσσόμενο μενού. Για παράδειγμα, η κατηγορία Dairy & Eggs περιέχει όλα τα γαλακτοκομικά είδη και τα αυγά (σχ. 3.2β΄). Μέσα από οποιαδήποτε σελίδα κατηγορίας προϊόντων, ο χρήστης μπορεί να προσθέσει ένα προϊόν στη λίστα επιθυμιών, πατώντας το κουμπί `Add to Wishlist`, ή σε μια παραγγελία, πατώντας το κουμπί `Add to Order`. Επίσης, τα ονόματα των προϊόντων είναι σύνδεσμοι (links) και πατώντας τους, ο χρήστης ανακατευθύνεται στη σελίδα προβολής λεπτομερειών για το εκάστοτε προϊόν (σχ. 3.3).

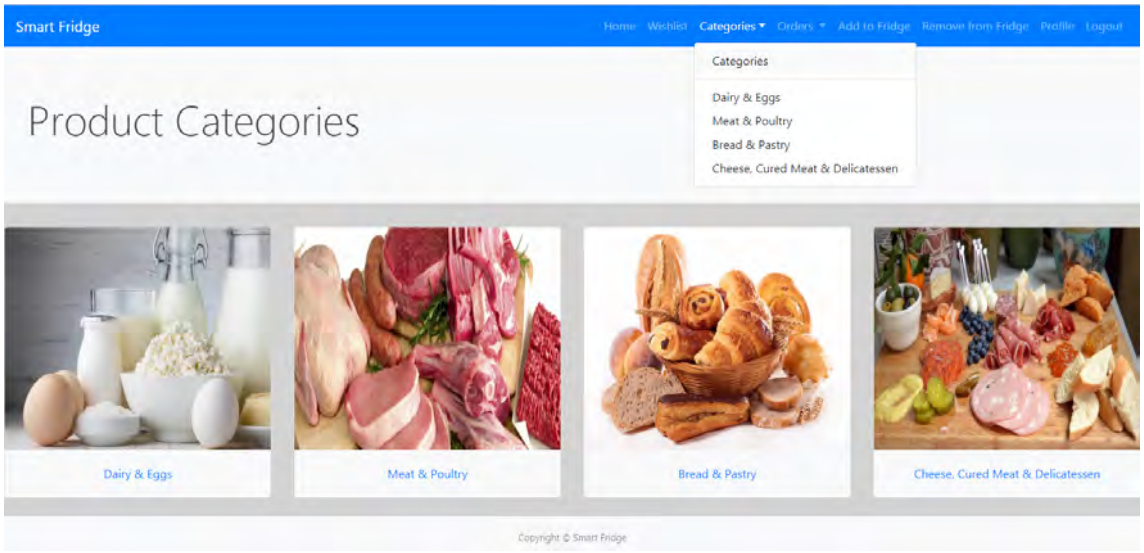


Σχήμα 3.1: Αρχικοποιημένη βάση δεδομένων. Ο πίνακας barcodes, ο οποίος περιέχει όλα τα διαθέσιμα προϊόντα, είναι αρχικοποιημένος με 10 προϊόντα

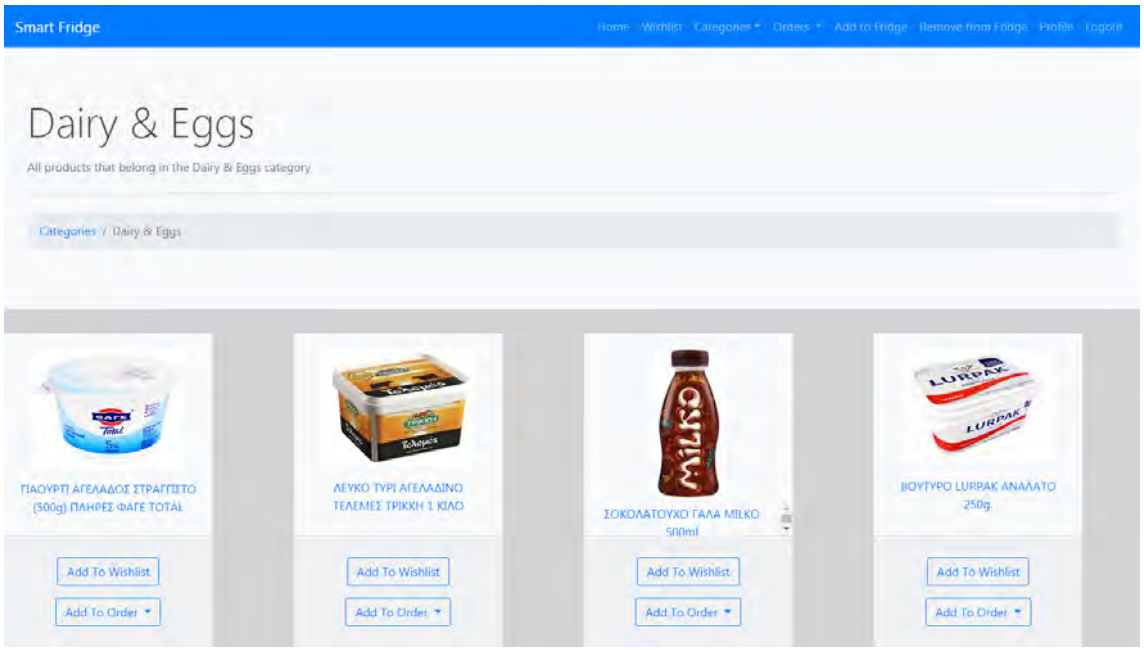
3.3 Λογαριασμοί χρηστών

Κατά την ενσωμάτωση των χρηστών σε μια εφαρμογή, υπάρχουν τρεις βασικές πτυχές που πρέπει να εξεταστούν:

1. αντιπροσώπευση χρηστών στη βάση δεδομένων
2. πιστοποίηση χρήστη

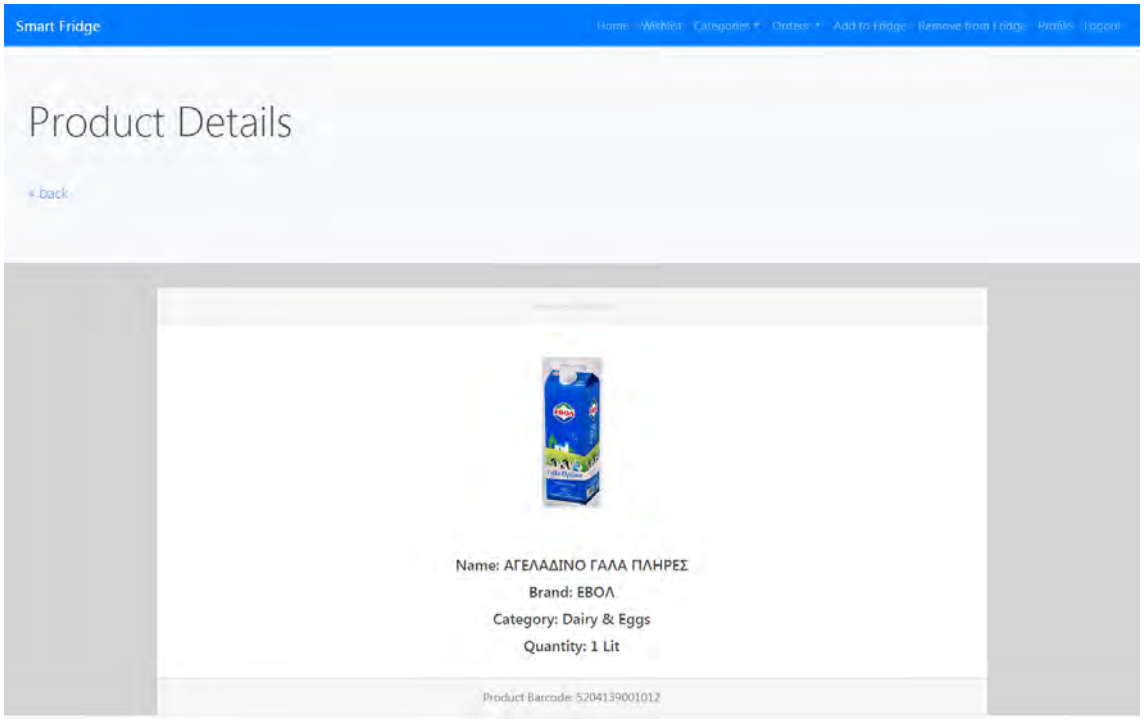


(α΄) Διαθέσιμες κατηγορίες



(β΄) Κατηγορία Dairy & Eggs

Σχήμα 3.2: Κατηγορίες προϊόντων



Σχήμα 3.3: Λεπτομέρειες προϊόντος

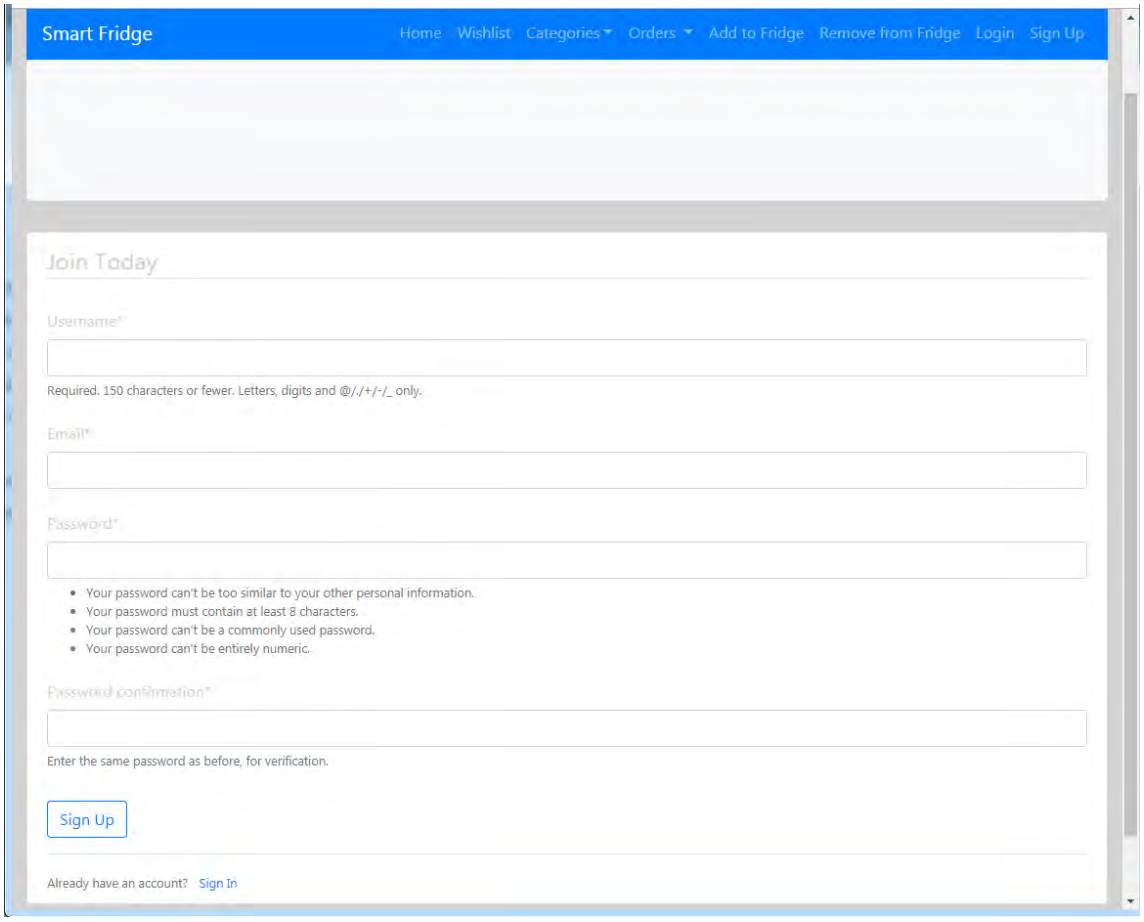
3. έλεγχος πρόσβασης

Η εφαρμογή επιτρέπει στους χρήστες να δημιουργήσουν έναν ή περισσότερους λογαριασμούς για τη σύνδεσή τους στην εφαρμογή. Για τη δημιουργία ενός λογαριασμού, ο χρήστης αρκεί να πατήσει το κουμπί Sign Up στο μενού επιλογών και να υποβάλλει έγκυρα διαπιστευτήρια (όνομα χρήστη, κωδικό, email) στη φόρμα εγγραφής (σχ. 3.4). Μάλιστα, μετά την εγγραφή του χρήστη δημιουργείται αυτόματα κι ένα προφίλ για αυτόν, όπου έχει τη δυνατότητα να δει και να επεξεργαστεί τα στοιχεία του, ή ακόμα και να προσθέσει μια φωτογραφία της επιλογής του (σχ. 3.5).

Πέρα από την εγγραφή, καμμία άλλη ενέργεια δεν μπορεί να πραγματοποιηθεί εάν ο χρήστης δεν είναι πιστοποιημένος. Για να χρησιμοποιήσει ένας χρήστης τις λειτουργίες της εφαρμογής, πρέπει πρώτα να συνδεθεί με τον λογαριασμό του. Πατώτας το κουμπί login, ανακατευθείνεται στην αντίστοιχη σελίδα για είσοδο στην εφαρμογή, πληκτρολογώντας τα στοιχεία του. Αν είναι ήδη συνδεδεμένος, μπορεί να αποσυνδεθεί πατώντας το κουμπί logout. (σχ. 3.6)

3.3.1 Ο σκοπός της επαλήθευσης χρήστη

Στην πραγματικότητα, είμαστε ήδη εξοικειωμένοι με τις κύριες πτυχές του ελέγχου ταυτότητας χρήστη: σύνδεση και έξοδος. Ο σκοπός της σύνδεσης είναι να επιτρέψουμε στην εφαρμογή να γνωρίζει ποιοι είμαστε, έτσι ώστε να μπορεί να μας παραχωρήσει τα απαραίτητα προνόμια (ή να μας περιορίσει την πρόσβαση σε ορισμένες λειτουργίες και δεδομένα). Αυτή η έννοια - της χορήγησης και του περιορισμού της πρόσβασης - αποκαλείται έλεγχος πρόσβασης. Άρα ο προγραμματιστής πρέπει να φροντίσει να ορίσει τα δικαιώματα του απλού χρήστη της εφαρμογής, και τις δυνατότητες που έχει, όπως επίσης και να ορίσει χρήστες με δικαιώματα διαχειριστή (superuser) για την



Σχήμα 3.4: Δημιουργία λογαριασμού χρήστη

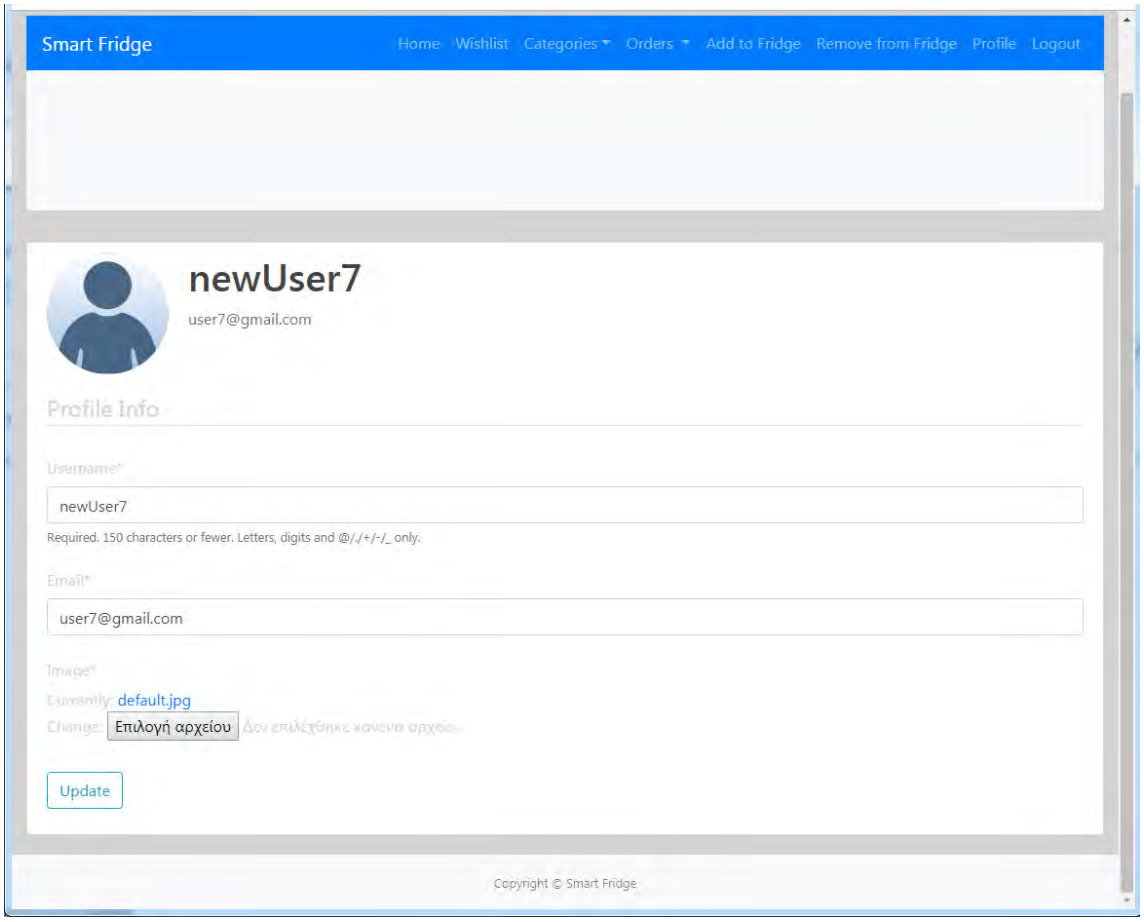
καλύτερη διαχείριση κι επίβλεψη της εφαρμογής.

3.3.2 Δεδομένα ειδικά για τον κάθε χρήστη

Κατά τον σχεδιασμό της εφαρμογής δεν πρέπει να ξεχνάμε ποια δεδομένα σχετίζονται με τον εκάστοτε χρήστη και ποια γενικότερα με την εφαρμογή. Για παράδειγμα, ο χρήστης Α τοποθετεί στο ψυγείο του το προϊόν Χ, ενώ ο χρήστης Β τοποθετεί το προϊόν Υ. Όταν ο χρήστης Α θελήσει να δει τη λίστα με όλα τα προϊόντα που βρίσκονται στο ψυγείο του, θα πρέπει να δει το προϊόν Χ, δηλαδή αυτά τα δεδομένα που είναι σχετικά με τον λογαριασμό του. Παρόμοια, η λίστα με τα προϊόντα του ψυγείου για τον χρήστη Β θα περιέχει μόνο το προϊόν Υ.

3.4 Καταγραφή των προϊόντων του ψυγείου

Οι προσδοκίες μας είναι ότι το ψυγείο, πέρα από τη διατήρηση των προϊόντων, θα μπορεί να παρακολουθήσει τα τρόφιμα που υπάρχουν μέσα στο ψυγείο. Δηλαδή, η εφαρμογή πρέπει να αναγνωρίσει με κάποιον τρόπο τα προϊόντα που εισέρχονται στο ψυγείο. Δύο πιθανές προσεγγίσεις αυτού του προβλήματος είναι τα barcodes και η τεχνολογία RFID (ταυτοποίηση μέσω ραδιοσυχνοτήτων).

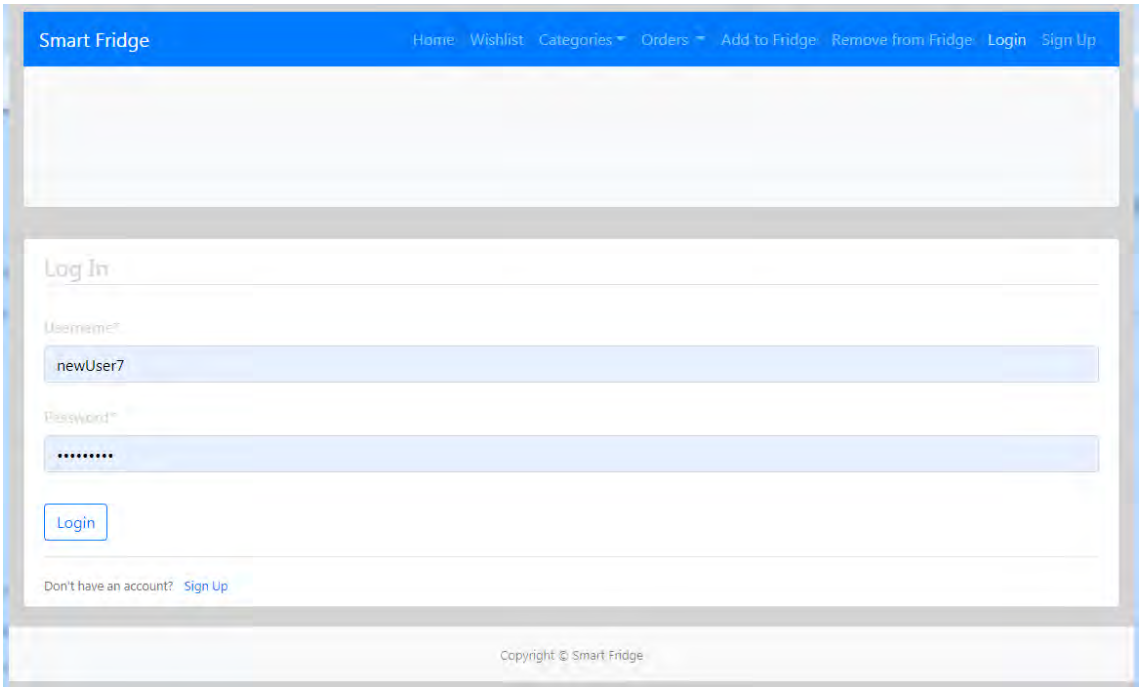


Σχήμα 3.5: Προφίλ χρήστη

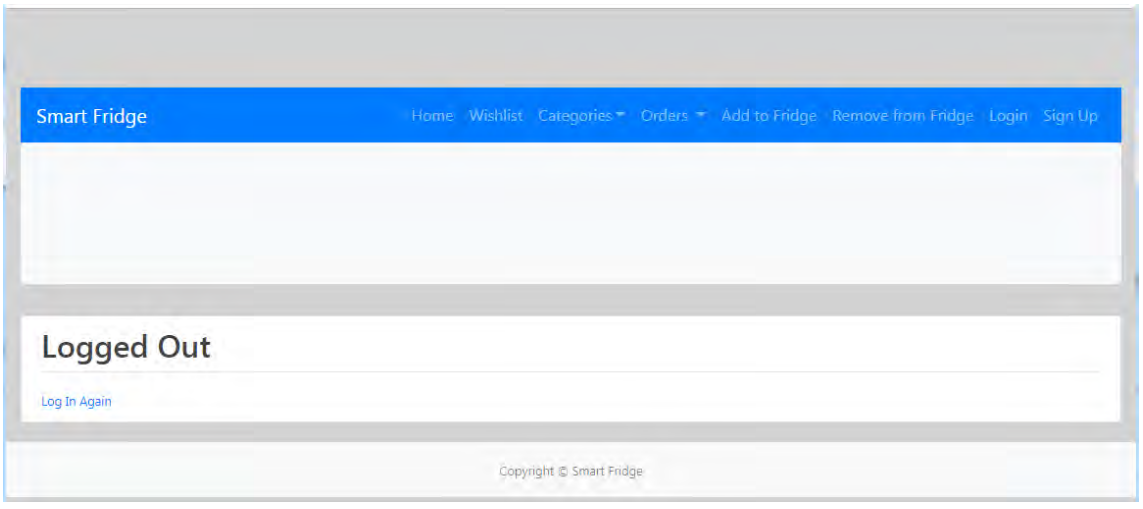
barcode: Το barcode είναι δεδομένα αναγνώσιμα από ένα οπτικό μηχάνημα σαρωτή barcode (barcode scanner/reader) το οποίο μας παρέχει διάφορα δεδομένα για το εκάστοτε προϊόν, όπως το όνομα του κατασκευαστή ή τη διάρκειά του (ημερομηνία λήξης). Όμως τα barcodes δεν είναι μοναδικά. Δηλαδή όλα τα προϊόντα του ίδιου τύπου θα έχουν τον ίδιο αριθμό barcode. Συνεπώς, η εφαρμογή πρέπει να έχει κατάλληλη προσέγγιση κατά την αφαίρεση προϊόντων με βάση τον αριθμό barcode από το ψυγείο, καθώς όπως είπαμε ο αριθμός barcode δεν αναγνωρίζει μοναδικά κάθε προϊόν.

RFID: Η τεχνολογία RFID (Radio Frequency Identification) ή αλλιώς ”ηλεκτρονική ετικέτα”, έχει την ικανότητα να εντοπίζει και να ταυτοποιεί τα προϊόντα, με μοναδικό τρόπο, χρησιμοποιώντας ραδιοκύματα, με την προϋπόθεση ότι κάθε αντικείμενο που χρειάζεται ταυτοποίηση έχει μια RFID ετικέτα (tag) [17].

Στην παρούσα εργασία θα ασχοληθούμε μόνο με την προσέγγιση του barcode, καθώς η τεχνολογία RFID δεν χρησιμοποιείται (τουλάχιστον μέχρι σήμερα) στα σουπερμάρκετ και τα προϊόντα δεν έχουν RFID ετικέτες για να αναγνωριστούν από το ψυγείο.



(α΄) Σελίδα Login, σύνδεση χρήστη



(β΄) Σελίδα Logout, αποσύνδεση χρήστη

Σχήμα 3.6: Σύνδεση κι αποσύνδεση χρήστη

3.4.1 Μέθοδος εισαγωγής προϊόντων στο ψυγείο

Όπως αναφέρθηκε, θέλουμε η εφαρμογή να γνωρίζει ποια προϊόντα εισέρχονται στο ψυγείο, ώστε να μας παρέχει τις διάφορες λειτουργίες και δυνατότητές της. Ο ιδανικός τρόπος για την καταγραφή προϊόντων του ψυγείου κατά την εισαγωγή, χρησιμοποιώντας την τεχνολογία barcode, είναι η σάρωση του αριθμού barcode του εκάστοτε προϊόντος από έναν ενσωματωμένο σαρωτή barcode του ψυγείου, παρόμοιας λειτουργικότητας με αυτή των σαρωτών του σουπερμάρκετ. Ιδανικά, ο σαρωτής αυτός θα βρίσκεται στην πόρτα του ψυγείου, κάτω από την οθόνη επιλογών της εφαρμογής (σχ. 2.1). Επειδή όμως τα παραπάνω αφορούν το κατασκευαστικό κομμάτι του ψυγείου, για τους σκοπούς της εργασίας και της εφαρμογής επιτυγχάνουμε τη λειτουργικότητα της εισαγωγής προϊόντων με σάρωση barcode με τη χρήστη μιας εφαρμογής android, η οποία μετατρέπει την κάμερα του smartphone σε σαρωτή barcode. Δηλαδή ο χρήστης, πατώντας το κουμπί `Add To Fridge` του μενού, ανακατευθείνεται στη σελίδα σάρωσης προϊόντος εισαγωγής (σχ. 3.7α΄), όπου σαρώνει τον αριθμό barcode με την κάμερα του κινητού του, μέσω της εφαρμογής `Barcode to pc` [4], προσθέτοντας το προϊόν αυτό στη λίστα προϊόντων της εφαρμογής. Σε περίπτωση αποτυχίας του σαρωτή, ο χρήστης έχει τη δυνατότητα να εισάγει χειροκίνητα τον αριθμό barcode.

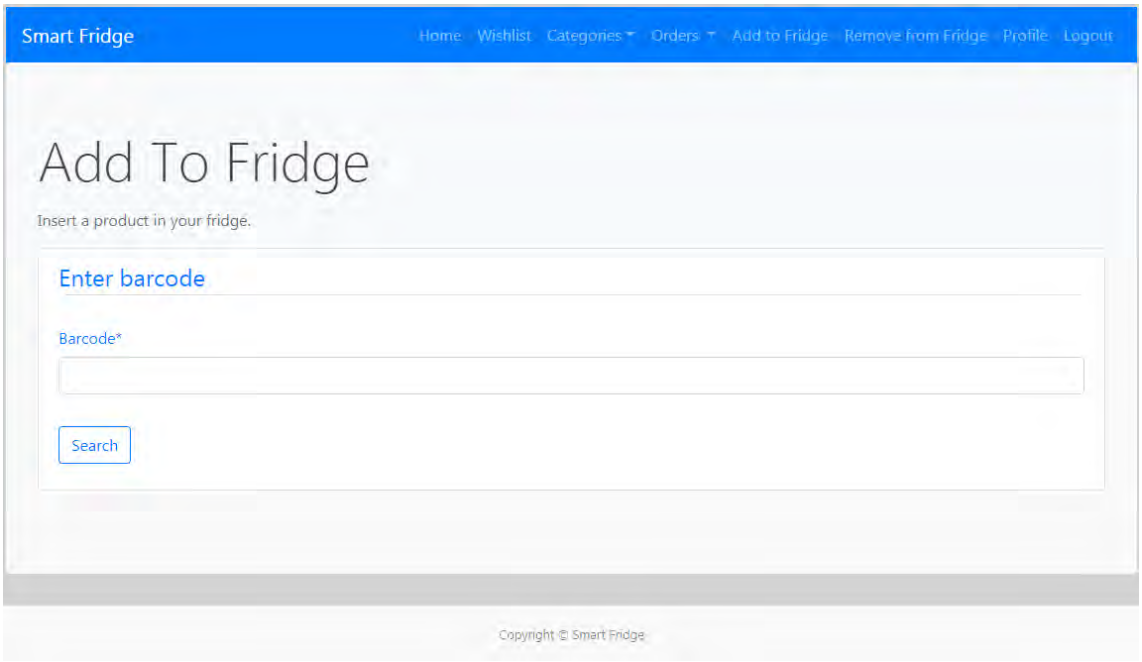
3.4.2 Μέθοδος εξαγωγής προϊόντων από το ψυγείο

Για την καταγραφή προϊόντων του ψυγείου δεν αρκεί μόνο η λειτουργικότητα εισαγωγής. Είναι απαραίτητο η εφαρμογή να γνωρίζει και ποια προϊόντα απομακρύνονται. Επομένως, το ψυγείο χρειάζεται να έχει είτε έναν επιπλέον σαρωτή barcode για εξαγωγή, δίπλα από αυτόν της εισαγωγής που αναφέρθηκε παραπάνω, είτε μια λειτουργικότητα διακόπτη εισαγωγής/εξαγωγής, εάν τελικά χρησιμοποιηθεί ένας σαρωτής που αναλαμβάνει και τις δυο λειτουργίες. Ο τρόπος λειτουργίας της εφαρμογής, όσον αφορά την εξαγωγή προϊόντων του ψυγείου, είναι παρόμοιος με αυτόν της εισαγωγής. Στη σελίδα `Remove from Fridge` (σχ. 3.7β΄) ο χρήστης σαρώνει τον αριθμό barcode του προϊόντος προς εξαγωγή, χρησιμοποιώντας την ίδια εφαρμογή android που αναφέρθηκε προηγουμένως. Επειδή όμως ο κωδικός barcode δεν είναι μοναδικός για κάθε προϊόν, που σημαίνει ότι δυο ή περισσότερα όμοια προϊόντα θα έχουν τον ίδιο αριθμό barcode, η πολιτική της εξαγωγής σε αυτήν την περίπτωση είναι η αφαίρεση του πιο παλιού χρονολογικά προϊόντος, δηλαδή αυτού που εισήχθει πρώτο στο ψυγείο μέσω της εφαρμογής.

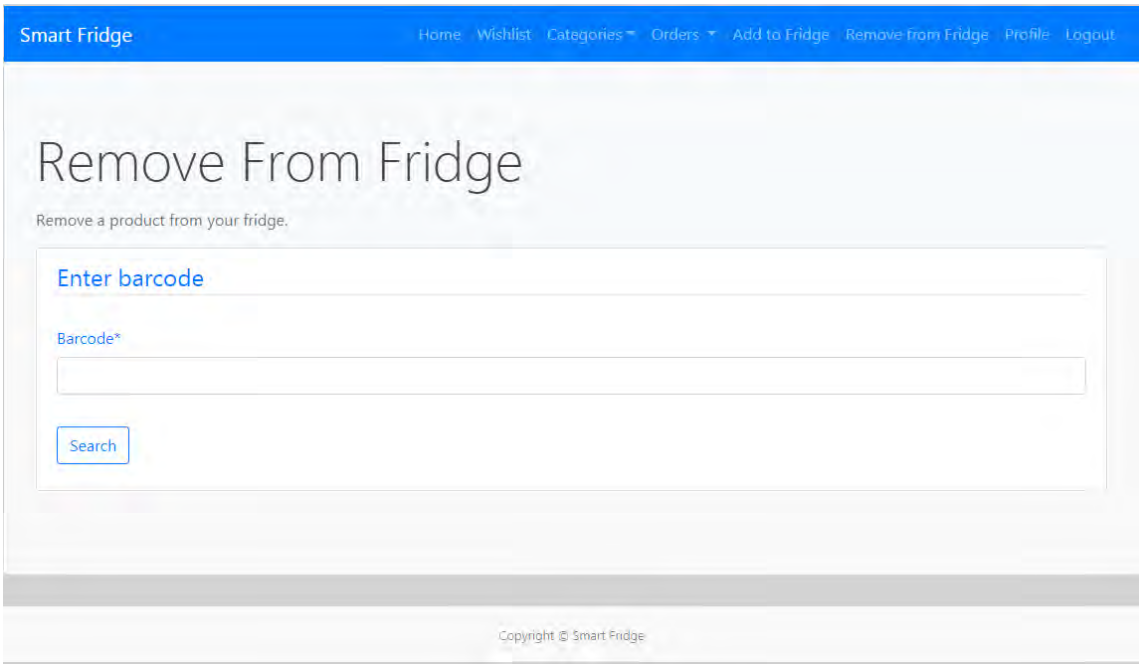
3.4.3 Προβολή καταγεγραμμένων προϊόντων του ψυγείου

Ο χρήστης έχει τη δυνατότητα να δει όλα τα προϊόντα που έχει καταχωρήσει στην εφαρμογή, στη σελίδα Home (σχ. 3.8). Επίσης, κάθε προϊόν έχει διαφορετική απόχρωση περιγράμματος, ανάλογα με την ημερομηνία λήξης του, δηλαδή:

- **πράσινο**: το προϊόν είναι φρέσκο
- **πορτοκαλί**: το προϊόν πλησιάζει την ημερομηνία λήξης του (εμφανίζεται και σχετική ειδοποίηση στο μενού επιλογών)

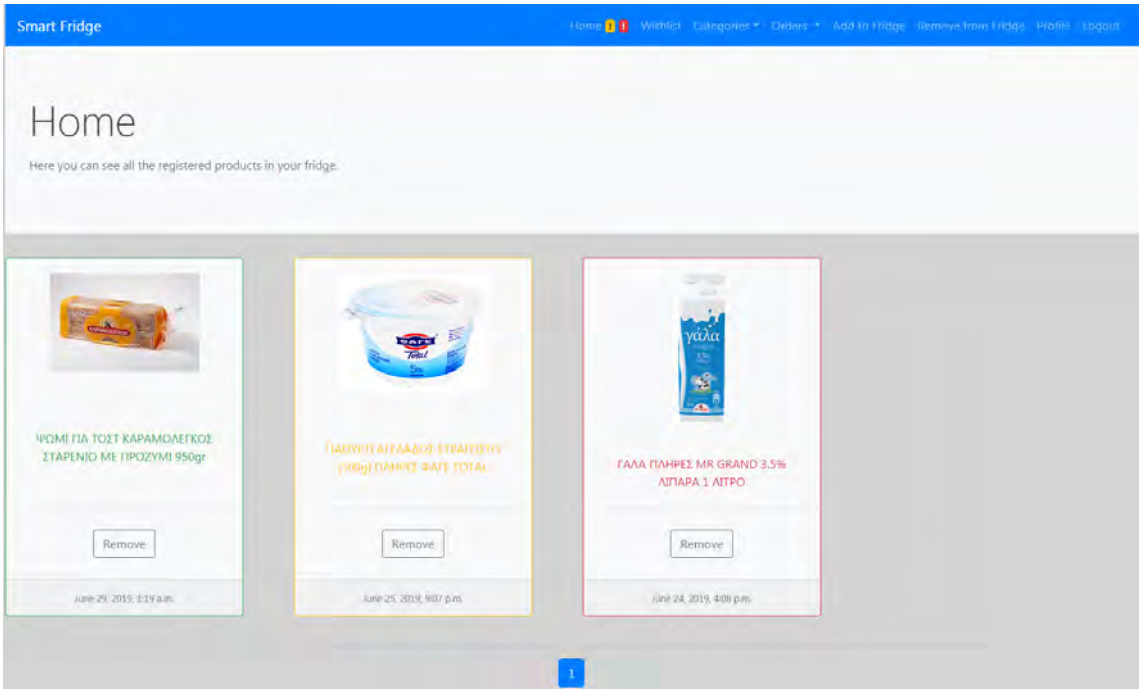


(α΄) Σελίδα Add to Fridge, εισαγωγή προϊόντος



(β΄) Σελίδα Remove from Fridge, εξαγωγή προϊόντος

Σχήμα 3.7: Σελίδες εισαγωγής κι εξαγωγής προϊόντων



Σχήμα 3.8: Σελίδα Home, προβολή καταγεγραμμένων προϊόντων

- **κόκκινο**: το προϊόν έχει φτάσει/περάσει την ημερομηνία λήξης του (εμφανίζεται και σχετική ειδοποίηση στο μενού επιλογών)

Τέλος, ο χρήστης μπορεί να αφαιρέσει ένα προϊόν από τη λίστα προϊόντων του ψυγείου, πατώντας το κουμπί `Remove`. Αυτός ο τρόπος αφαίρεσης προϊόντων αντικαθιστά αυτόν με τη χρήση του σαρωτή barcode και είναι απαραίτητο εργαλείο σε περίπτωση που ο χρήστης δεν επιθυμεί να αφαιρέσει το πιο παλιό προϊόν εάν δυο ή περισσότερα προϊόντα είναι ίδια, δηλαδή έχουν τον ίδιο αριθμό barcode.

3.5 Ηλεκτρονικές παραγγελίες σε σουπερμάρκετ

Στην υλοποίησή μας θέλουμε ο χρήστης να μπορεί να δημιουργήσει όποτε χρειάζεται κάποια παραγγελία προϊόντων που θα σταλεί σε ένα σουπερμάρκετ, όπως επίσης και να δημιουργούνται αυτόματα παραγγελίες και να αποστέλλονται. Για να πετύχουμε το τελευταίο, χρειάζεται η εφαρμογή να γνωρίζει κάθε στιγμή τι βρίσκεται στο ψυγείο και τι θέλει ο χρήστης να υπάρχει στο ψυγείο, και με βάση τη διαφορά αυτών των δύο, να δημιουργεί μια λίστα με προϊόντα προς παραγγελία. Ακόμα, πρέπει να ορίσουμε μια συνθήκη η οποία θα ενεργοποιεί την αποστολή της αυτόματης παραγγελίας, όπως για παράδειγμα, τον ορισμό ημέρας κι ώρας αποστολής. Άρα η εφαρμογή πρέπει με κάποιον τρόπο να κάνει περιοδικούς ελέγχους για τον έλεγχο ημέρας κι ώρας (πχ. αν ο χρήστης έχει ορίσει αποστολή παραγγελιών κάθε Τετάρτη στις 11 π.μ, βρίσκουμε την τωρινή ώρα και ημέρα και τις συγκρίνουμε με την επιλογή του χρήστη). Αυτή η λειτουργικότητα διαφέρει από αυτή της αίτησης-απάντησης που συζητήσαμε στο προηγούμενο κεφάλαιο και χρειάζεται διαφορετική αντιμετώπιση κατά τη σχεδίαση της εφαρμογής.

3.5.1 Παραγγελίες χρήστη

Η εφαρμογή δίνει τη δυνατότητα στον χρήστη να δημιουργήσει παραγγελίες προϊόντων οι οποίες θα σταλούν σε σουπερμάρκετ ηλεκτρονικά. Πατώντας το αναπτυσσόμενο κουμπί `Orders` και στη συνέχεια `My Orders`, ο χρήστης ανακατευθύνεται στη σελίδα δημιουργίας κι επεξεργασίας των παραγγελιών του. Μια νέα παραγγελία δημιουργείται πατώντας το κουμπί `New Order`. Κάθε παραγγελία αναγνωρίζεται από την ημερομηνία δημιουργίας της. Δεν υπάρχει περιορισμός στον αριθμό παραγγελιών, ενώ ο χρήστης μπορεί να στείλει μια παραγγελία πάνω από μια φορές. Στην εικόνα 3.9 ο χρήστης έχει δημιουργήσει δυο παραγγελίες. Για κάθε παραγγελία, υπάρχουν οι εξής δυνατότητες:

- **προσθήκη προϊόντων** στη λίστα, πατώντας το κουμπί `Edit`. Ο χρήστης ανακατευθύνεται στη σελίδα με τις κατηγορίες προϊόντων, ώστε να βρει στη συνέχεια τα προϊόντα που τον ενδιαφέρουν και να τα προσθέσει στην παραγγελία που θέλει πατώντας το κουμπί `Add to Order` που συζητήσαμε σε προηγούμενη ενότητα.

- **προσθήκη τεμαχίου/αφαίρεση τεμαχίου/διαγραφή προϊόντος** από τη λίστα πατώντας τα αντίστοιχα κουμπιά −/+/x δίπλα από το όνομα του εκάστοτε προϊόντος.

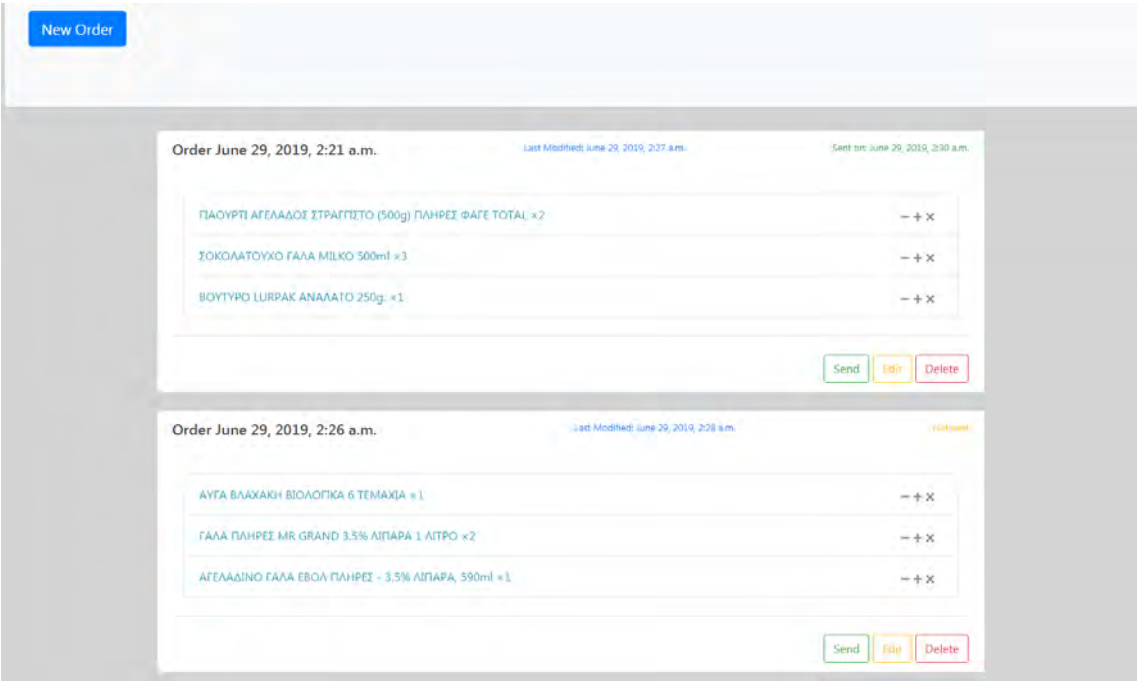
- **αποστολή της παραγγελίας** πατώντας το κουμπί `Send`

- **διαγραφή της παραγγελίας** πατώντας το κουμπί `Delete`

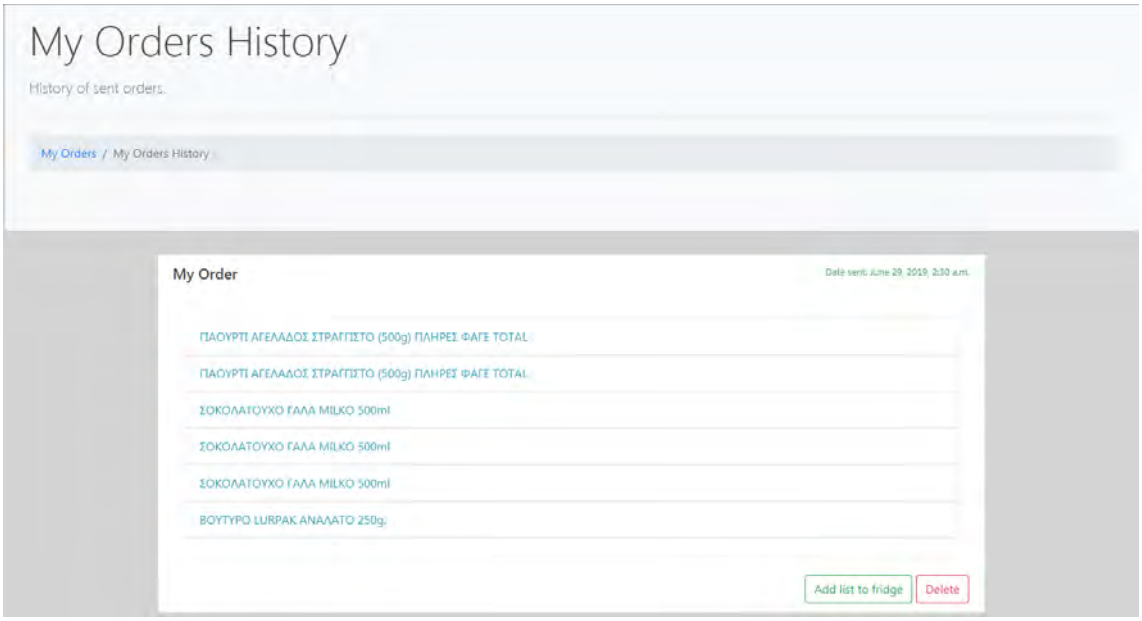
Επίσης, για κάθε παραγγελία αναγράφεται η ημερομηνία τελευταίας τροποποίησης, καθώς και η ημερομηνία τελευταίας αποστολής (εάν έχει σταλεί). Ο χρήστης μπορεί να δει το ιστορικό των παραγγελιών που έχουν σταλεί στη σελίδα `My Orders History` (σχ. 3.10), όπου πατώντας το κουμπί `Add List to Fridge` προσθέτει αυτόματα όλα τα προϊόντα της παραγγελίας στη λίστα με τα προϊόντα που υπάρχουν μέσα στο ψυγείο.

3.5.2 Αυτόματη Παραγγελία - ρόλος λίστας επιθυμιών

Ο σκοπός της ύπαρξης της αυτόματης παραγγελίας είναι το ψυγείο να περιέχει κάθε φορά τα προϊόντα που ο χρήστης επιθυμεί ή θεωρεί απαραίτητα. Για τον ορισμό αυτών των προϊόντων μπορεί να χρησιμοποιήσει τη λίστα επιθυμιών. Το τελικό περιεχόμενο της αυτόματης παραγγελίας θα είναι τα προϊόντα που υπάρχουν στη λίστα επιθυμιών, αλλά απουσιάζουν από το ψυγείο. Στη σελίδα `Scheduled Order` (σχ. 3.11) γίνεται ο ορισμός της ημέρας κι ώρας της εβδομάδας όπου θα γίνεται κάθε φορά η αποστολή της αυτόματης παραγγελίας. Το περιεχόμενο της λίστας ανανεώνεται πατώντας το κουμπί `Update`. Η ακύρωση της προγραμματισμένης αυτόματης παραγγελίας μπορεί να γίνει ανά πάσα στιγμή πατώντας το κουμπί `Cancel`. Η εφαρμογή ειδοποιεί τον χρήστη κάθε φορά που η παραγγελία είναι έτοιμη κι ο χρήστης εγκρίνει τελικά την αποστολή της πατώντας το κουμπί `Send`. Η παραγγελία δηλαδή αποστέλλεται στο σουπερμάρκετ μόνο εφόσον εγκριθεί από τον χρήστη. Στη σελίδα `Scheduled Orders History` (σχ. 3.12) υπάρχει το ιστορικό των αυτόματων παραγγελιών που έχουν σταλεί.



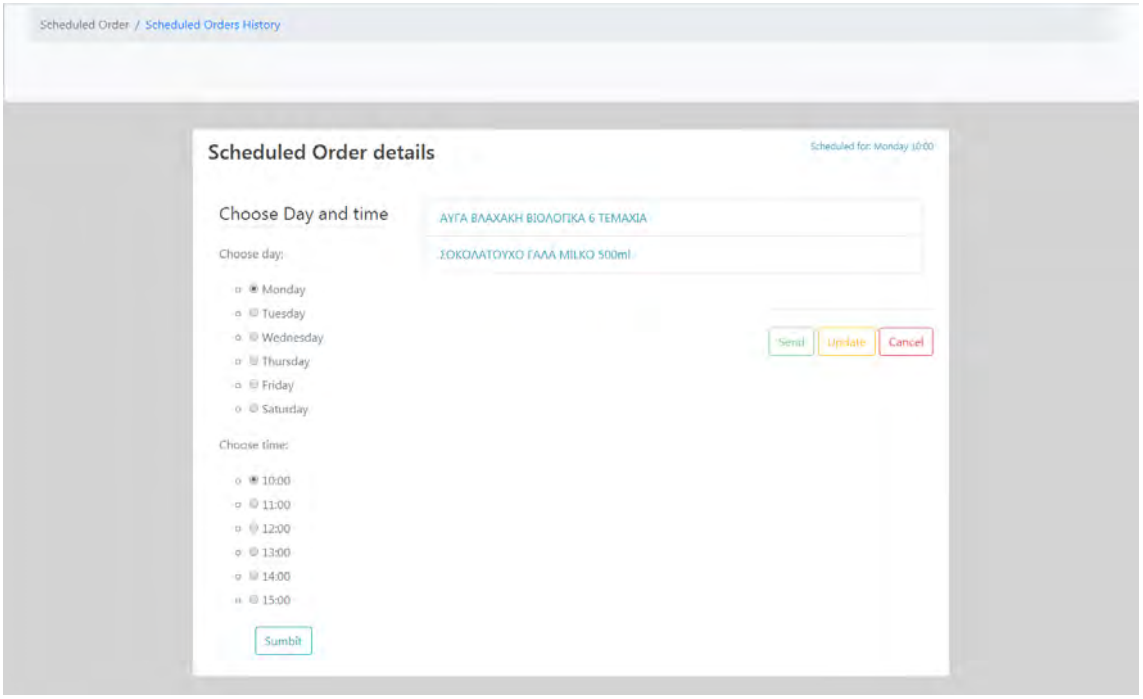
Σχήμα 3.9: Σελίδα My Orders - Επεξεργασία παραγγελιών χρήστη



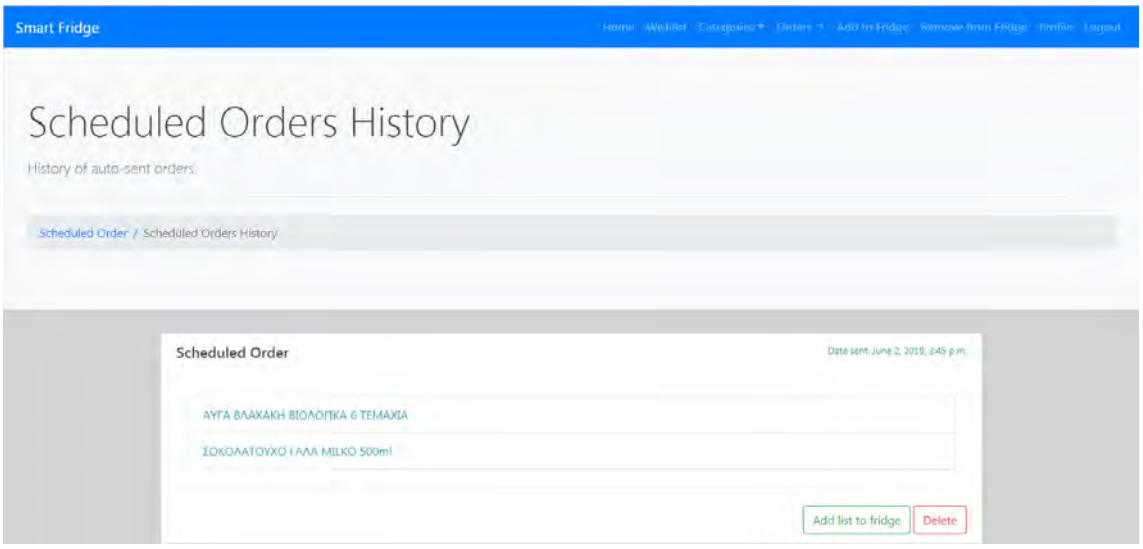
Σχήμα 3.10: Σελίδα My Orders History - Προβολή ιστορικού παραγγελιών χρήστη

3.6 Απομακρυσμένη σύνδεση και χρήση της εφαρμογής

Επειδή πρόκειται για μια εφαρμογή δικτύου η οποία θα είναι διαθέσιμη για χρήση από την οθόνη του έξυπνου ψυγείου ακολουθώντας το μοντέλο του IoT, θέλουμε ο χρήστης να έχει τη δυνατότητα να συνδεθεί και να χρησιμοποιήσει την εφαρμογή από μια διαφορετική συσκευή, πέρα από το ψυγείο. Αυτό είναι εφικτό, αρκεί η συσκευή αυτή να είναι συνδεδεμένη στο διαδίκτυο.



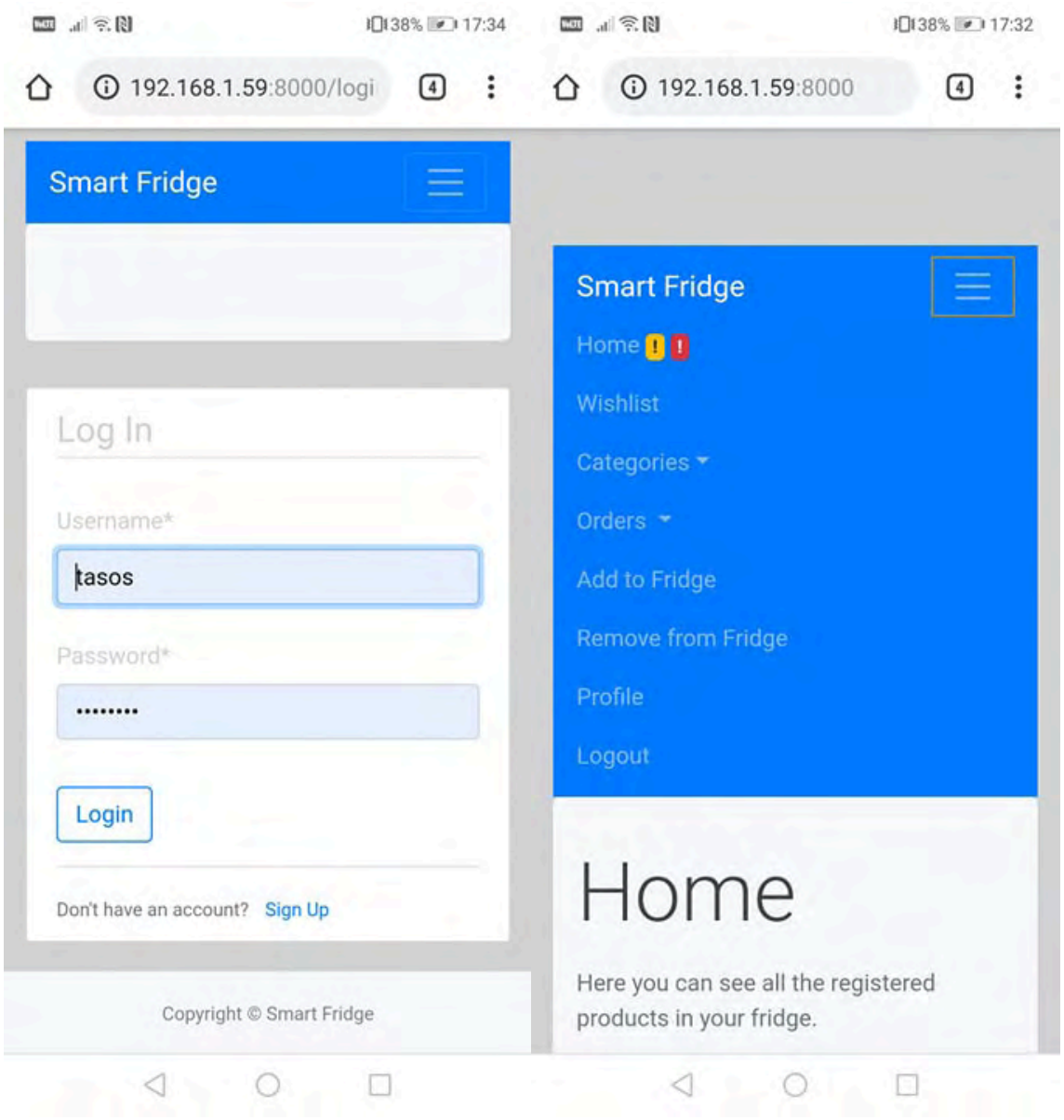
Σχήμα 3.11: Σελίδα Scheduled Order - Διαχείριση αυτόματης παραγγελίας χρήστη



Σχήμα 3.12: Σελίδα Scheduled Orders History - Προβολή ιστορικού αυτόματων παραγγελιών χρήστη

Στην εικόνα 3.13α΄ φαίνεται ένα στιγμιότυπο χρήσης της εφαρμογής από συσκευή smartphone. Το μενού επιλογών εμφανίζεται με το αναπτυσσόμενο κουμπί (σχ. 3.13β΄). Όλες οι λειτουργίες που συζητήθηκαν προηγουμένως είναι διαθέσιμες στις διάφορες απομακρυσμένες συσκευές. Για πρόσβαση στην εφαρμογή, ο χρήστης πληκτρολογεί τη διεύθυνση της ιστοσελίδας της εφαρμογής στον φυλλομετρητή του, όπως θα έκανε για οποιαδήποτε άλλη ιστοσελίδα. Επισημαίνεται ότι για τους σκοπούς της διπλωματικής, η εφαρμογή βρίσκεται σε στάδιο ανάπτυξης και η πρόσβαση

επιτυγχάνεται πληκτρολογώντας την διεύθυνση ip του μηχανήματος που “τρέχει” ο διακομιστής.



(α΄) Σελίδα Login (β΄) Βασικό μενού

Σχήμα 3.13: Χρήση εφαρμογής μέσω smartphone - Περιβάλλον χρήστη

Κεφάλαιο 4

Υλοποίηση της εφαρμογής

4.1 Εισαγωγή

Σε αυτό το κεφάλαιο γίνεται παρουσίαση της εφαρμογής κι ανάλυση των τρόπων υλοποίησης των βασικών δυνατοτήτων της που συζητήθηκαν στο προηγούμενο κεφάλαιο. Αρχικά, γίνεται περιγραφή της βάσης δεδομένων και του γραφικού περιβάλλοντος του χρήστη (UI - user interface). Στη συνέχεια ακολουθεί επεξήγηση των αλγορίθμων για την αποθήκευση προϊόντων στο ψυγείο και στη λίστα επιθυμιών. Τέλος, θα γίνει ανάλυση του τρόπου πραγματοποίησης παραγγελιών χειροκίνητα από τον χρήστη κι αυτόματα από το ψυγείο.

Επισημαίνεται ότι κάθε project στο django αποτελείται από επιμέρους εφαρμογές, από τις οποίες απαρτίζεται ολόκληρη η εφαρμογή μας. Το συγκεκριμένο project αποτελείται από δυο επιμέρους εφαρμογές, την **UI**, η οποία περιέχει γενικότερα τη λειτουργικότητα της διαδικτυακής εφαρμογής και την **users**, η οποία αφορά τη λειτουργικότητα των χρηστών, όπως τη δημιουργία προφίλ (profile). Η δομή καταλόγου (directory) του συνολικού project απεικονίζεται στο σχήμα 4.1.

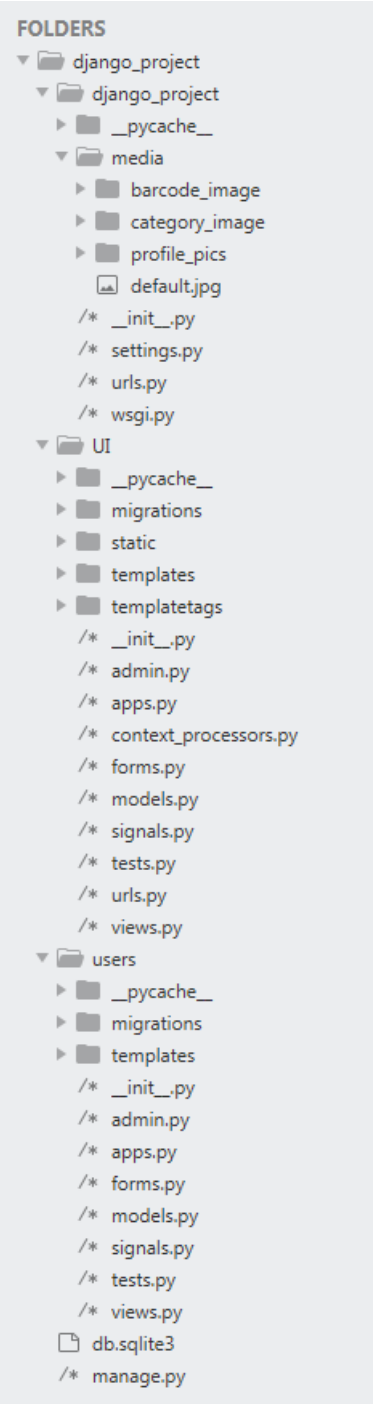
4.2 Βάση Δεδομένων

Η βάση δεδομένων αποτελείται από 12 πίνακες. Όπως είπαμε, κάθε πίνακας βάσης δεδομένων αντιστοιχεί σε ένα μοντέλο django, οπότε αυτός θα είναι ο όρος που θα χρησιμοποιείται. Για να δημιουργήσουμε νέα μοντέλα στο project μας, τα δηλώνουμε ως κλάσεις python στο αρχείο models.py της εφαρμογής που επιθυμούμε. Το σχήμα βάσης δεδομένων απεικονίζεται στο σχήμα 4.2.

4.2.1 Μοντέλο User

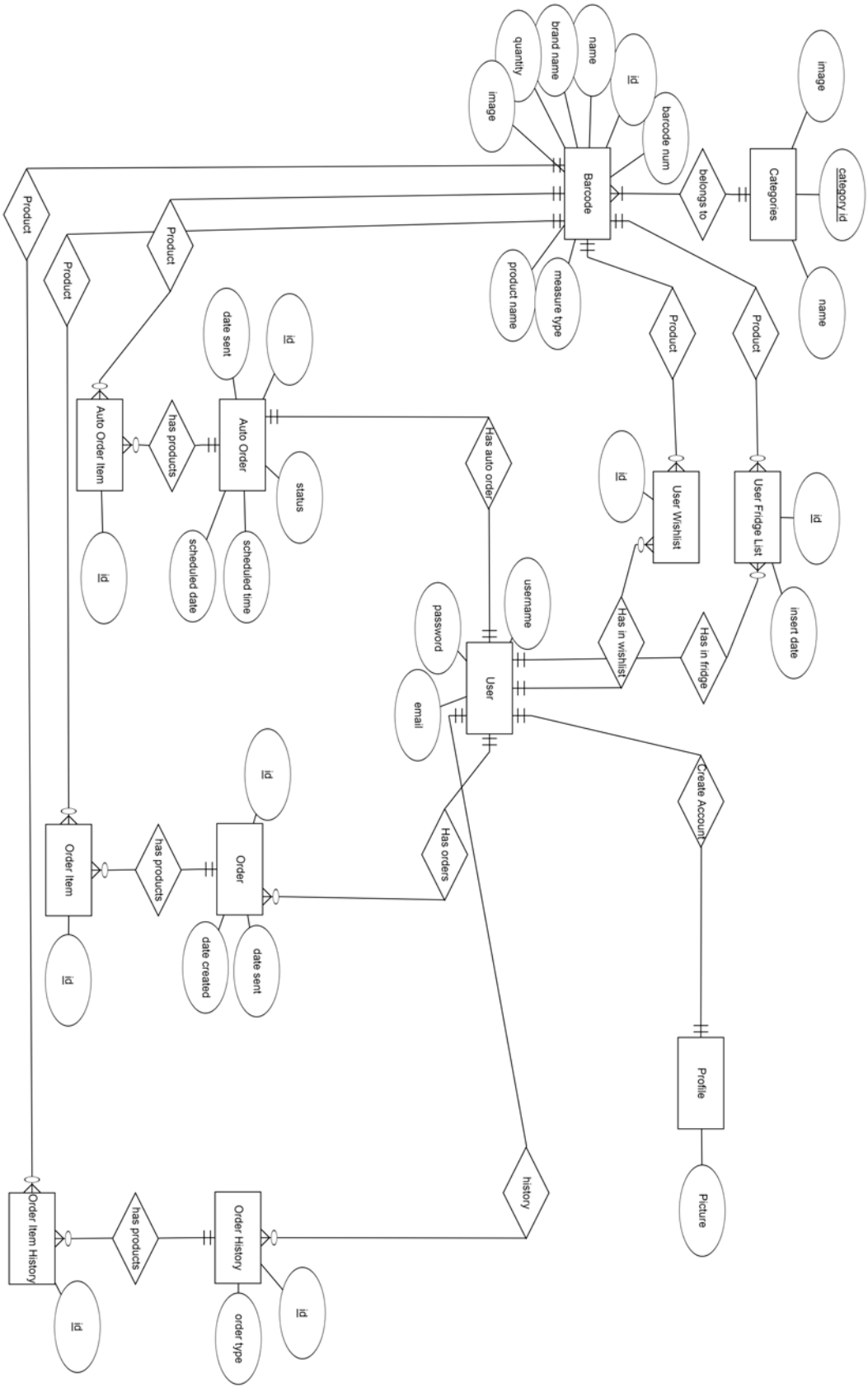
Το μοντέλο User είναι ήδη υλοποιημένο στο django και παρέχεται στους προγραμματιστές από το βασικό σύστημα πιστοποίησής (authentication system) του. Για να το χρησιμοποιήσουμε, αρκεί να το κάνουμε εισαγωγή στον κώδικα με την εντολή import:

```
1 from django.contrib.auth.models import User
```



Σχήμα 4.1: Δομή καταλόγου του συνολικού Project

Από τα πεδία που ήδη περιέχει, εμείς θα χρειαστούμε τα username, password και email (πίνακας 4.1).



Σχήμα 4.2: Σχήμα βάσης δεδομένων. Μόνο τα βασικά πεδία των πινάκων φαίνονται στο σχήμα.

Όνομα Πεδίου	Περιγραφή
username	όνομα χρήστη για έναν λογαριασμό
password	κωδικός χρήστη για έναν λογαριασμό
email	email χρήστη για έναν λογαριασμό

Πίνακας 4.1: Μοντέλο User

Όνομα Πεδίου	Περιγραφή
id	πρωτεύον Κλειδί
user	ο χρήστης στον οποίο ανήκει το προφίλ
image	εικόνα προφίλ του χρήστη

Πίνακας 4.2: Μοντέλο Profile

4.2.2 Μοντέλο Profile

Το μοντέλο Profile (πίνακας 4.2) βρίσκεται στην εφαρμογή users και είναι υπεύθυνο για την πληροφορία του προφίλ του χρήστη, αφού δημιουργήσει έναν νέο λογαριασμό.

Ορισμός του profile μοντέλου στο αρχείο models.py:

```
1  from django.db import models
2  from django.contrib.auth.models import User
3  from PIL import Image
4  class Profile(models.Model):
5      user = models.OneToOneField(User, on_delete=models.CASCADE)
6
7      image = models.ImageField(default='default.jpg', upload_to='profile_pics')
8
9
10  def __str__(self):
11   return f'{self.user.username} Profile'
12
13  def save(self, *args, **kwargs):
14   super(Profile, self).save(*args, **kwargs)
15
16   img = Image.open(self.image.path)
17
18   if img.height > 300 or img.width > 300:
19    output_size = (300, 300)
20    img.thumbnail(output_size)
21    img.save(self.image.path)
```

Υπάρχει συσχέτιση 1-1 μεταξύ των πινάκων profile και user, καθώς κάθε χρήστης έχει ένα και μοναδικό προφίλ κι αντιστρόφως, κάθε προφίλ ανήκει σε έναν μοναδικό χρήστη. Στη γραμμή 5, ορίζοντας “on delete = models.CASCADE” δηλώνουμε ότι αν διαγραφεί το User αντικείμενο

Όνομα Πεδίου	Περιγραφή
category_id	μοναδικό αναγνωριστικό κάθε κατηγορίας προϊόντων
category_name	όνομα της κατηγορίας προϊόντων
image	εικόνα της κατηγορίας προϊόντων

Πίνακας 4.3: Μοντέλο Categories

(object), τότε διαγράφεται και το Profile αντικείμενο της συσχέτισης. Δηλαδή αν ένας χρήστης διαγράψει τον λογαριασμό του, αυτόματα διαγράφεται και το προφίλ του. Γενικά, η μέθοδος save επεκτείνει τη λειτουργικότητα του μοντέλου. Μια μέθοδος save εκτελείται πάντα μετά τη δημιουργία του μοντέλου στο οποίο περιέχεται. Συγκεκριμένα εδώ, αφού δημιουργηθεί το προφίλ του χρήστη, προσαρμόζουμε αυτόματα το μέγεθος της εικόνας του προφίλ του ώστε να μην ξεπερνά το μέγεθος 300x300. Η γραμμή 3 είναι απαραίτητη για την χρήση πεδίου τύπου ImageField. Βλέπουμε ότι στο μοντέλο Profile δεν ορίζεται κάποιο πρωτεύον κλειδί (primary key) για τον πίνακα. Στην πραγματικότητα, εάν ο προγραμματιστής δεν ορίσει κάποιο πεδίο ως πρωτεύον κλειδί κατά τη δήλωση ενός μοντέλου, δημιουργείται αυτόματα από το django ένα πεδίο “id” με την ιδιότητα του μοναδικού κλειδιού, το οποίο μπορούμε να χρησιμοποιήσουμε εάν χρειαστεί.

4.2.3 Μοντέλο Categories

Το Categories μοντέλο (πίνακας 4.3) είναι ο πίνακας με όλες τις διαθέσιμες κατηγορίες προϊόντων (γαλακτοκομικά, κρεατικά, κλπ.) και βρίσκεται στο αρχείο models.py της εφαρμογής UI, όπως κι όλα τα υπόλοιπα μοντέλα της εφαρμογής UI.

Κώδικας:

```
1 from django.db import models
2
3 class Categories(models.Model):
4  category_id = models.IntegerField(primary_key=True)
5  category_name = models.TextField(max_length=100)
6  image = models.ImageField(upload_to='category_image', blank='True')
```

4.2.4 Μοντέλο Barcode

Το μοντέλο Barcode (πίνακας 4.4) περιέχει όλη την πληροφορία που χρειάζεται να ορίσουμε για ένα προϊόν (όνομα, κατασκευαστής, κτλ.). Έχει ένα-προς-πολλά (one-to-many) συσχέτιση με το μοντέλο Categories, καθώς πολλά προϊόντα μπορούν να ανήκουν σε μια κατηγορία.

Κώδικας στο αρχείο models.py της εφαρμογής UI:

```
1 from django.db import models
2
3 class Barcode(models.Model):
4  item_id = models.IntegerField()
5  name = models.TextField(max_length=100)
```

Όνομα Πεδίου	Περιγραφή
id	πρωτεύον κλειδί
item_id	αριθμός barcode του προϊόντος
name	ολόκληρο όνομα του προϊόντος (όνομα, μάρκα, ποσότητα)
category	κατηγορία στην οποία ανήκει το προϊόν
product_name	όνομα του προϊόντος
brand_name	όνομα της μάρκας του προϊόντος
measure_type	τύπος μέτρησης (κιλά, γραμμάρια, κτλ.)
quantity	ποσότητα του προϊόντος
image	εικόνα του προϊόντος

Πίνακας 4.4: Μοντέλο Barcode

Όνομα Πεδίου	Περιγραφή
item_id	μοναδικό αναγνωριστικό του Userfridgelist αντικειμένου
user	ο χρήστης στον οποίο ανήκει το Userfridgelist αντικείμενο
barcode	προϊόν που περιέχεται στο Userfridgelist αντικείμενο
insert_date	ημερομηνία που προστέθηκε το Userfridgelist αντικείμενο

Πίνακας 4.5: Μοντέλο Userfridgelist

```
6   category = models.ForeignKey(Categories, on_delete=models.DO_NOTHING)
7   product_name = models.TextField(max_length=100, default='unknown')
8   brand_name = models.TextField(max_length=100, default='unknown')
9   measure_type = models.TextField(max_length=100, default='unknown')
10  quantity = models.IntegerField(default=0)
11  image = models.ImageField(upload_to='barcode_image', blank='True')
```

4.2.5 Μοντέλο Userfridgelist

Το Μοντέλο Userfridgelist (πίνακας 4.5) είναι η λίστα με τα προϊόντα που έχει ο κάθε χρήστης στο ψυγείο του. Δηλαδή, όταν ο χρήστης προσθέσει ένα αντικείμενο στο ψυγείο του, αυτόματα αυτό προστίθεται σε αυτόν τον πίνακα. Επίσης, όταν ο χρήστης αφαιρεί ένα προϊόν από το ψυγείο, πρέπει να αφαιρείται κι από τη λίστα. Βλέπουμε ότι υπάρχει συσχέτιση ένα-προς-πολλά μεταξύ Userfridgelist και του User. Πρέπει να γίνει κατανοητή στον προγραμματιστή η λειτουργικότητα του Userfridgelist, καθώς η λέξη ”λίστα”, προγραμματιστικά δεν είναι ”σωστή”. Στην ουσία, όταν ο χρήστης προσθέτει ένα προϊόν στο ψυγείο, αποθηκεύεται ένα αντικείμενο της κλάσης Userfridgelist, το οποίο αντικείμενο περιέχει αυτό το προϊόν, για αυτόν τον χρήστη. Δηλαδή, εάν ο χρήστης έχει στο ψυγείο του 3 προϊόντα, θα υπάρχουν 3 Userfridgelist objects, όπου το κάθε αντικείμενο θα έχει την πληροφορία του προϊόντος (barcode γνώρισμα), την πληροφορία του χρήστη

Όνομα Πεδίου	Περιγραφή
item_id	μοναδικό αναγνωριστικό του Userproductlist object
user	ο χρήστης στον οποίο ανήκει το Userproductlist object
product	προϊόν που περιέχεται στο Userproductlist object

Πίνακας 4.6: Μοντέλο Userproductlist

(user γνώρισμα), το αναγνωριστικό του (item_id γνώρισμα) και την ημερομηνία προσθήκης (insert_date γνώρισμα). Στο πεδίο DateTimeField, δηλώνοντας default=timezone.now κατά τη δημιουργία του αντικειμένου, αν δεν ορίσουμε κάποια ημερομηνία, η προκαθορισμένη ημερομηνία θα είναι αυτή της δημιουργίας του αντικειμένου.

Κώδικας:

```
1 from django.db import models
2 from django.utils import timezone
3
4 class Userfridgelist(models.Model):
5  item_id = models.AutoField(primary_key=True, serialize=False, verbose_name='ID
      ')
6  user = models.ForeignKey(User, on_delete=models.CASCADE, default='')
7  barcode = models.ForeignKey(Barcode, on_delete=models.DO_NOTHING, blank=True,
      null=True)
8  insert_date = models.DateTimeField(default=timezone.now)
```

4.2.6 Μοντέλο Userproductlist

Το Μοντέλο Userfridgelist (πίνακας 4.6) είναι ο πίνακας με τα προϊόντα που προσθέτει ο χρήστης στη λίστα επιθυμιών του. Η λειτουργικότητα είναι παρόμοια με αυτή του Userfridgelist μοντέλου: κάθε προϊόν στη λίστα επιθυμιών του χρήστη, αντιστοιχεί σε ένα Userproductlist object.

Κώδικας:

```
1 from django.db import models
2 from django.utils import timezone
3
4 class Userproductlist(models.Model):
5  item_id = models.AutoField(primary_key=True, serialize=False, verbose_name='ID
      ')
6  user = models.ForeignKey(User, on_delete=models.CASCADE, default='')
7  product = models.ForeignKey(Barcode, on_delete=models.DO_NOTHING, blank=True,
      null=True)
```

4.2.7 Μοντέλο Order

Το μοντέλο Order (πίνακας 4.7) είναι ο πίνακας που περιέχει όλες τις παραγγελίες που δημιουργεί ο χρήστης. Υπάρχει μία ένα-προς-πολλά συσχέτιση με τον πίνακα User, αφού κάθε χρή-

Όνομα Πεδίου	Περιγραφή
order_id	μοναδικό αναγνωριστικό της παραγγελίας
user	ο χρήστης στον οποίο ανήκει η παραγγελία
order_num	αριθμός παραγγελίας
date_sent	ημερομηνία αποστολής παραγγελίας
last_modified	ημερομηνία τροποποίησης του order αντικειμένου
created	ημερομηνία δημιουργίας της παραγγελίας

Πίνακας 4.7: Μοντέλο Order

Όνομα Πεδίου	Περιγραφή
order_item_id	μοναδικό αναγνωριστικό του OrderItem αντικειμένου
order	η παραγγελία στην οποία ανήκει το OrderItem αντικείμενο
product	προϊόν που περιέχεται στο OrderItem αντικείμενο
quantity	ποσότητα προϊόντος στην παραγγελία
desc	περιγραφή προϊόντος στην παραγγελία
last_modified	ημερομηνία τροποποίησης του order αντικειμένου
created	ημερομηνία δημιουργίας του order αντικειμένου

Πίνακας 4.8: Μοντέλο OrderItem

στης μπορεί να έχει πολλές παραγγελίες, αλλά κάθε μια από αυτές τις παραγγελίες ανήκει σε έναν μοναδικό χρήστη, αυτόν που τη δημιούργησε.

Κώδικας:

```
1 from django.db import models
2
3 class Order(models.Model):
4  order_id = models.AutoField(primary_key=True, serialize=False, verbose_name='
      ID')
5  user = models.ForeignKey(User, on_delete=models.CASCADE, default='')
6  order_num = models.IntegerField()
7  date_sent = models.DateTimeField(null=True)
8  last_modified = models.DateTimeField(auto_now=True)
9  created = models.DateTimeField(auto_now_add=True, null=True)
```

4.2.8 Μοντέλο OrderItem

Ο πίνακας OrderItem (πίνακας 4.8) έχει μία ένα-προς-πολλά συσχέτιση με τον πίνακα Order, και περιλαμβάνει όλα τα προϊόντα μιας παραγγελίας. Δηλαδή, αν θέλουμε η παραγγελία να περιέχει 3 προϊόντα, τότε θα δημιουργηθούν 3 OrderItem αντικείμενα, με “order = A”. Κάθε OrderItem αντικείμενο περιέχει και το αντίστοιχο προϊόν.

Όνομα Πεδίου	Περιγραφή
auto_order_id	μοναδικό αναγνωριστικό της αυτόματης παραγγελίας
user	ο χρήστης στον οποίο ανήκει η αυτόματη παραγγελία
status	κατάσταση της αυτόματης παραγγελίας (not init, scheduled, sent, cancelled)
scheduled_date	προγραμματισμένη ημέρα αποστολής παραγγελίας
scheduled_time	προγραμματισμένη ώρα αποστολής παραγγελίας
date_sent	ημερομηνία αποστολής παραγγελίας
last_modified	ημερομηνία τροποποίησης του AutoOrder αντικειμένου
created	ημερομηνία δημιουργίας του AutoOrder αντικειμένου

Πίνακας 4.9: Μοντέλο AutoOrder

Κώδικας:

```
1  from django.db import models
2
3  class OrderItem(models.Model):
4   order_item_id = models.AutoField(primary_key=True, serialize=False,
        verbose_name='ID')
5   order = models.ForeignKey(Order, on_delete=models.CASCADE, null=True) #an
        diagrafei to order, diagrafete kai to orderItem object
6   product = models.ForeignKey(Barcode, on_delete=models.DO_NOTHING)
7   quantity = models.IntegerField(default=1)
8   desc = models.TextField(max_length=500, null=True, blank=True)
9   last_modified = models.DateTimeField(auto_now=True)
10  created = models.DateTimeField(auto_now_add=True)
```

4.2.9 Μοντέλο AutoOrder

Ο πίνακας AutoOrder (πίνακας 4.9) περιέχει πληροφορία για την αυτόματη παραγγελία που δημιουργείται για τον χρήστη από την εφαρμογή. Υπάρχει συσχέτιση ένα-προς-ένα μεταξύ των πινάκων User κι Auto Order, καθώς κάθε χρήστης έχει μια μοναδική αυτόματη παραγγελία. Το αντικείμενο AutoOrder δημιουργείται με τη δημιουργία του αντικειμένου User, και δεν μπορεί να δημιουργηθεί άλλο ή να διαγραφεί (εκτός κι αν διαγραφεί ο λογαριασμός του χρήστη).

Κώδικας:

```
1  from django.db import models
2
3  class AutoOrder(models.Model):
4   auto_order_id = models.AutoField(primary_key=True, serialize=False,
        verbose_name='ID')
5   user = models.OneToOneField(User, on_delete=models.CASCADE)
6   status = models.TextField(max_length=20, default='not_init')
7   scheduled_date = models.TextField(null=True)
8   scheduled_time = models.TextField(null=True)
9   date_sent = models.DateTimeField(null=True)
```

Όνομα Πεδίου	Περιγραφή
order_item_id	μοναδικό αναγνωριστικό του AutoOrderItem object
order	η παραγγελία στην οποία ανήκει το AutoOrderItem αντικείμενο
product	προϊόν που περιέχεται στο AutoOrderItem αντικείμενο
last_modified	ημερομηνία τροποποίησης του AutoOrder αντικειμένου
created	ημερομηνία δημιουργίας του AutoOrder αντικειμένου

Πίνακας 4.10: Μοντέλο AutoOrderItem

Όνομα Πεδίου	Περιγραφή
order_id	μοναδικό αναγνωριστικό του OrderHistory αντικειμένου
user	ο χρήστης στον οποίο ανήκει το OrderHistory αντικείμενο
orderType	τύπος της παραγγελίας (αυτόματη ή όχι)
date_sent	ημερομηνία αποστολής παραγγελίας

Πίνακας 4.11: Μοντέλο OrderHistory

```
10   last_modified = models.DateTimeField(auto_now=True)
11   created = models.DateTimeField(auto_now_add=True)
```

4.2.10 Μοντέλο AutoOrderItem

Το μοντέλο AutoOrderItem (πίνακας 4.10) περιλαμβάνει όλα τα προϊόντα που περιέχονται σε κάθε αυτόματη παραγγελία. Ο τρόπος λειτουργίας είναι παρόμοιος με αυτόν του OrderItem.

Κώδικας:

```
1  from django.db import models
2
3  class AutoOrderItem(models.Model):
4   order_item_id = models.AutoField(primary_key=True, serialize=False,
        verbose_name='ID')
5   order = models.ForeignKey(AutoOrder, on_delete=models.CASCADE, null=True)
6   product = models.ForeignKey(Barcode, on_delete=models.DO_NOTHING)
7   last_modified = models.DateTimeField(auto_now=True)
8   created = models.DateTimeField(auto_now_add=True)
```

4.2.11 Μοντέλο OrderHistory

Ο πίνακας OrderHistory (πίνακας 4.11) είναι το ιστορικό παραγγελιών. Έχει μία ένα-προς-πολλά συσχέτιση με τον πίνακα User, καθώς κάθε φορά που μια παραγγελία, είτε αυτόματη είτε όχι, πρέπει να αποθηκευτεί στο ιστορικό, δημιουργείται ένα OrderHistory αντικείμενο που κρατά πληροφορία της παραγγελίας.

Κώδικας:

Όνομα Πεδίου	Περιγραφή
order_item_id	μοναδικό αναγνωριστικό του OrderItemHistory αντικειμένου
product	προϊόν της αντίστοιχης παραγγελίας
order	η παραγγελία στην οποία ανήκει το OrderItemHistory αντικείμενο

Πίνακας 4.12: Μοντέλο OrderItemHistory

```
1 from django.db import models
2
3 class OrderHistory(models.Model):
4  order_id = models.AutoField(primary_key=True, serialize=False, verbose_name='
       ID')
5  user = models.ForeignKey(User, on_delete=models.CASCADE, default='')
6  orderType = models.TextField(max_length=20, default='unknown')
7  date_sent = models.DateTimeField()
```

4.2.12 Μοντέλο OrderItemHistory

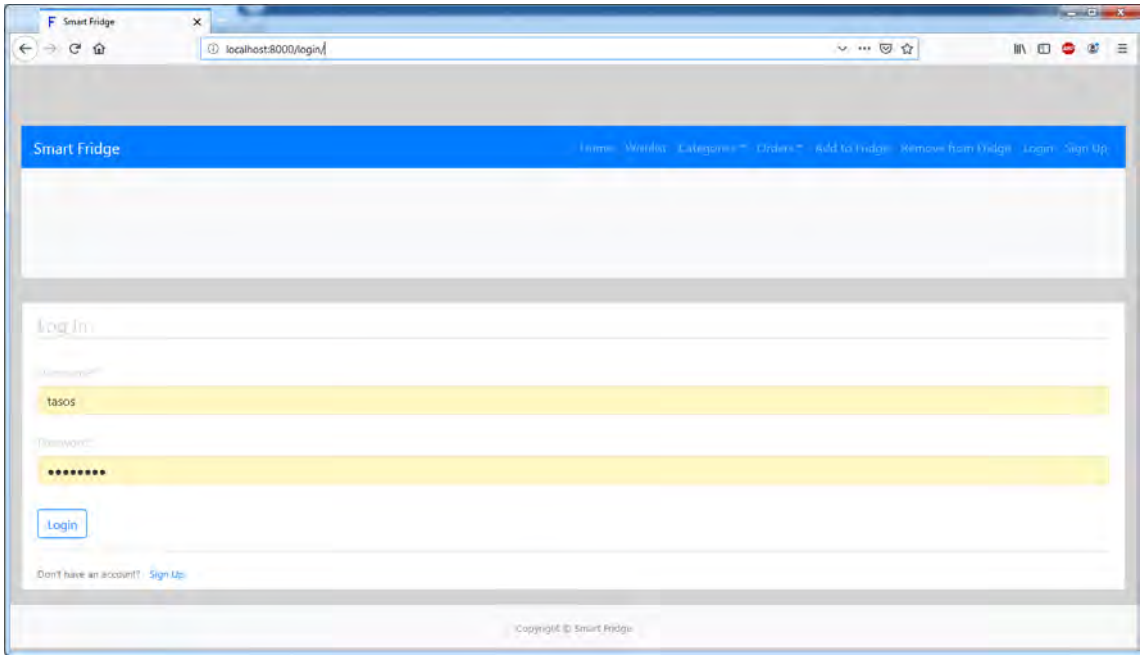
Το μοντέλο OrderItemHistory (πίνακας 4.12) περιλαμβάνει όλα τα αντικείμενα που ανήκουν σε ένα OrderHistory αντικείμενο. Δηλαδή, περιλαμβάνει όλα τα προϊόντα μιας παραγγελίας που βρίσκεται στο ιστορικό παραγγελιών. Υπάρχει συσχέτιση ένα-προς-πολλά με τον πίνακα OrderHistory, καθώς για κάθε ένα προϊόν της παραγγελίας του ιστορικού, υπάρχει κι ένα αντικείμενο OrderItemHistory που ανήκει σε αυτήν.

Κώδικας:

```
1 from django.db import models
2
3 class OrderItemHistory(models.Model):
4  order_item_id = models.AutoField(primary_key=True, serialize=False,
       verbose_name='ID')
5  product = models.ForeignKey(Barcode, on_delete=models.DO_NOTHING)
6  order = models.ForeignKey(OrderHistory, on_delete=models.CASCADE, null=True)
```

4.3 Λογαριασμός Χρήστη

Πληκτρολογώντας τη διεύθυνση της σελίδας της εφαρμογής, η οποία προς το παρόν είναι η ”http://localhost:8000/” καθώς τρέχουμε τον διακομιστή ανάπτυξης (development server) στη θύρα 8000, ο χρήστης ανακατευθύνεται στη σελίδα εισόδου χρήστη (σελίδα login, εικόνα 4.3), όπου μπορεί να συνδεθεί με έναν ήδη υπάρχοντα λογαριασμό, είτε να δημιουργήσει έναν καινούριο πατώντας sign up κάτω αριστερά στη φόρμα εισόδου, ή πάνω δεξιά στο πάνελ επιλογών. Όσο ο χρήστης δεν είναι συνδεδεμένος στην εφαρμογή, δηλαδή δεν είναι πιστοποιημένος, δεν μπορεί να περιηγηθεί σε κάποια άλλη σελίδα του πάνελ επιλογών κι ανακατευθύνεται στη σελίδα login.



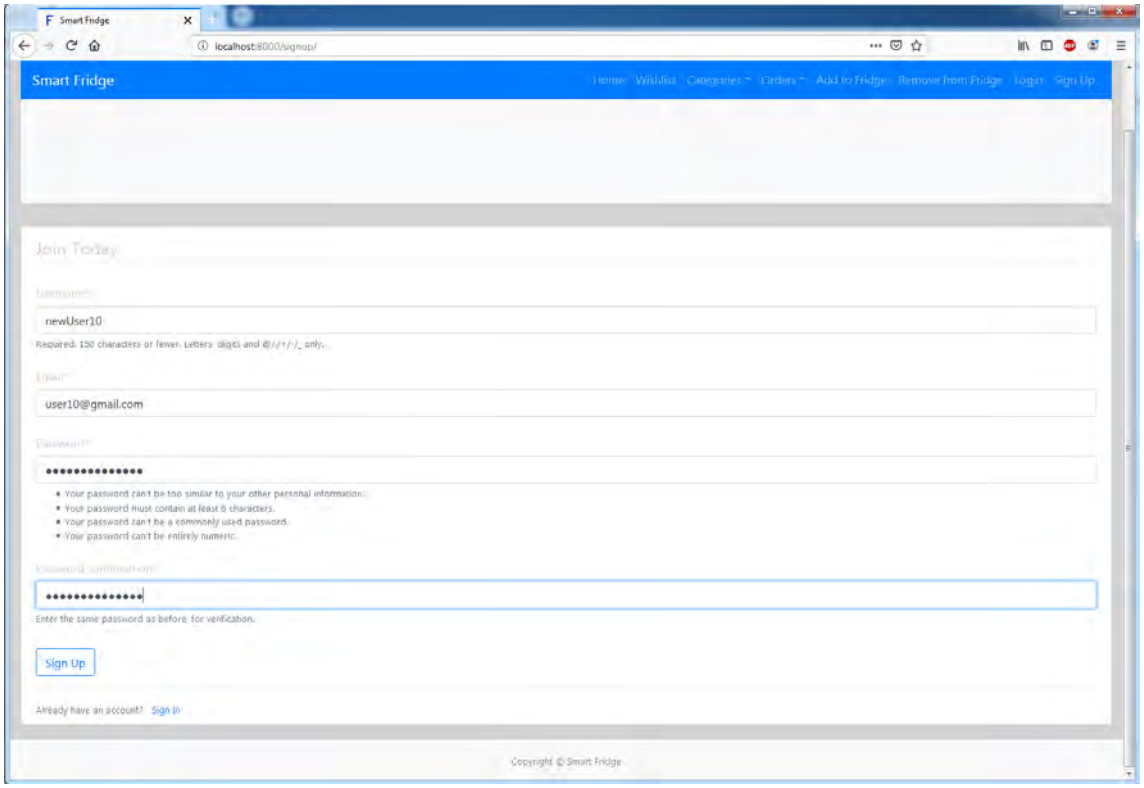
Σχήμα 4.3: Σελίδα login.

4.3.1 Δημιουργία λογαριασμού - Sign up

Πατώντας Sign up, ο χρήστης ανακατευθύνεται στη σελίδα signup (εικόνα 4.4) για τη δημιουργία λογαριασμού, όπου πρέπει να πληκτρολογήσει όνομα χρήστη, κωδικό και email. Τα δεδομένα πρέπει να είναι έγκυρα για να δημιουργηθεί ο λογαριασμός, αλλιώς ο χρήστης προτρέπεται να συμπληρώσει εκ νέου τη φόρμα. Για την αποθήκευση δεδομένων φόρμας στο django γίνεται χρήση ενός django form στο αρχείο forms.py, όπου βρίσκονται όλες οι φόρμες για μια εφαρμογή, και μιας συνάρτησης view στο αρχείο views.py. Στην ουσία, όταν ο χρήστης συμπληρώνει τη φόρμα και πατά το κουμπί submit, στέλνει στον διακομιστή ένα POST αίτημα. Αυτό το αίτημα θα το επεξεργαστεί η αντίστοιχη συνάρτηση view, η οποία θα πάρει τα δεδομένα που έχει στείλει ο χρήστης από τη φόρμα.

Συνάρτηση register (users/views.py), η οποία είναι υπεύθυνη για την εγγραφή του χρήση:

```
1  def register(request):
2
3   if request.method == 'POST':
4    form = UserRegisterForm(request.POST)
5    if form.is_valid():
6     form.save()
7     username = form.cleaned_data.get('username')
8     messages.success(request, f'Your account has been successfully created!')
9     return redirect('login')
10   else:
11    form = UserRegisterForm()
12
13   return render(request, 'users/register.html', {'form': form})
```



Σχήμα 4.4: Sign up Page.

Το μόνο που πρέπει να κάνουμε, είναι form.save() στη γραμμή 6, ώστε να αναλάβει η φόρμα την αποθήκευση δεδομένων στη βάση δεδομένων

Django signup form (users/forms.py):

```
1 class UserRegisterForm(UserCreationForm):
2  email = forms.EmailField()
3
4  class Meta:
5   model = User
6   fields = ['username', 'email', 'password1', 'password2']
```

Στη γραμμή 5 ορίζουμε το μοντέλο στο οποίο θα γίνει αποθήκευση δεδομένων, και στην επόμενη γραμμή τα πεδία που τροποποιούνται.

Αφού δημιουργηθεί ο λογαριασμός χρήστη, δηλαδή δημιουργηθεί ένα νέο User αντικείμενο (πίνακας 4.1), αυτόματα δημιουργείται κι ένα προφίλ (πίνακας 4.2) για αυτόν. Αυτή η αυτόματη δημιουργία του προφίλ επιτυγχάνεται με τη χρήση σημάτων του django (django signals).

users/signals.py:

```
1 from django.db.models.signals import post_save
2 from django.contrib.auth.models import User
3 from django.dispatch import receiver
4 from .models import Profile
5
6 @receiver(post_save, sender=User)
7 def create_profile(sender, instance, created, **kwargs):
```

```
8   if created:
9     Profile.objects.create(user=instance)
10
11  @receiver(post_save, sender=User)
12  def save_profile(sender, instance, **kwargs):
13    instance.profile.save()
```

Αφού η δημιουργία του προφίλ γίνεται μετά τη δημιουργία του λογαριασμού, χρησιμοποιούμε `post_save` κι αυτός που στέλνει το σήμα (sender) είναι το User αντικείμενο. Παρόμοια δημιουργείται και η **αυτόματη παραγγελία** για τον λογαριασμό.

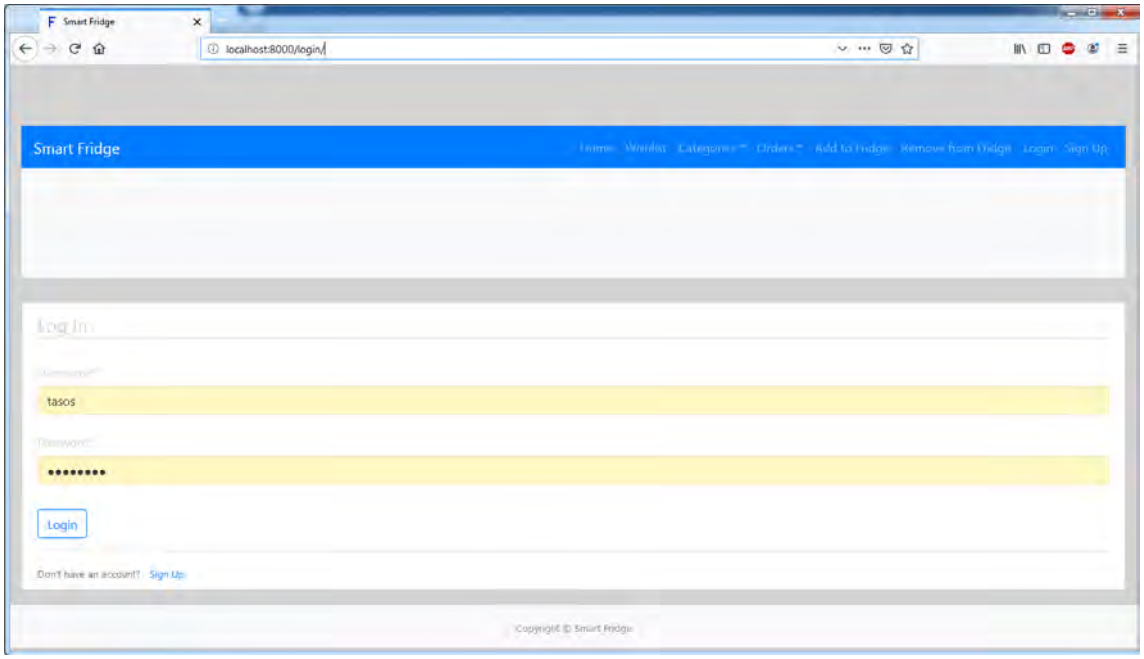
UI/signals.py:

```
1  from django.db.models.signals import post_save
2  from django.contrib.auth.models import User
3  from django.dispatch import receiver
4  from .models import AutoOrder
5
6  @receiver(post_save, sender=User)
7  def create_autoorder(sender, instance, created, **kwargs):
8    if created:
9      AutoOrder.objects.create(user=instance)
10
11 @receiver(post_save, sender=User)
12 def save_autoorder(sender, instance, **kwargs):
13   instance.autoorder.save()
```

4.3.2 Σύνδεση κι αποσύνδεση χρήστη (login, logout)

Ο χρήστης μπορεί να χρησιμοποιήσει όλη τη λειτουργικότητα της εφαρμογής, αφού πρώτα έχει συνδεθεί στον λογαριασμό του. Αφού ο χρήστης δημιουργήσει τον λογαριασμό του επιτυχώς, ανακατευθύνεται στη σελίδα login για να συνδεθεί με τα στοιχεία του. Ο χρήστης πιστοποιείται χρησιμοποιώντας τη λειτουργικότητα που παρέχεται από το django, μέσω του LoginView. Με τον ίδιο τρόπο γίνεται και η αποσύνδεση χρήστη (Logout), με το LogoutView. Αυτά τα views δεν υπάρχουν στο αρχείο views.py μας, αρκεί απλά να τα δηλώσουμε σαν views στο url αρχείο. Δηλαδη, μετά τη λειτουργικότητα εγγραφής, σύνδεσης κι αποσύνδεσης χρήστη, το αρχείο django-project/urls.py, που συνδέει τα URL-view-template, πρέπει να είναι ως εξής:

```
1  from django.contrib import admin
2  from django.contrib.auth import views as auth_views
3  from django.urls import path, include
4  from django.conf import settings
5  from django.conf.urls.static import static
6  from users import views as user_views
7
8  urlpatterns = [
9      path('admin/', admin.site.urls),
10     path('signup/', user_views.register, name='register'),
11     path('login/', auth_views.LoginView.as_view(template_name='users/login.html
           '), name='login'),
```



Σχήμα 4.5: Σελίδα Profile.

```
12      path('logout/', auth_views.LogoutView.as_view(template_name='users/logout.
            html'), name='logout'),
13      path('', include('UI.urls')), #set UI as home page
14
15  ]+static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

4.3.3 Προφίλ

Στη σελίδα Profile (εικόνα 4.5) ο χρήστης μπορεί να δει τα στοιχεία του (όνομα, κωδικός), να τα τροποποιήσει, ή ακόμα και να ανεβάσει μια καινούρια εικόνα προφίλ (αντικαθιστώντας την προκαθορισμένη).

Υπάρχουν δύο django forms οι οποίες αναλαμβάνουν την αποθήκευση των δεδομένων του χρήστη κατά την τροποποίηση των στοιχείων του και την αλλαγή της εικόνας προφίλ

users/forms.py:

```
1  class UserUpdateForm(forms.ModelForm):
2   email = forms.EmailField()
3
4   class Meta:
5    model = User
6    fields = ['username', 'email']
7
8  class ProfileUpdateForm(forms.ModelForm):
9   class Meta:
10   model = Profile
11   fields = ['image']
```

Η view συνάρτηση που αναλαμβάνει το αίτημα POST της φόρμας προφίλ (users/views.py):

```
1  @login_required
2  def profile(request):
3   if request.method == 'POST':
4    u_form = UserUpdateForm(request.POST, instance=request.user)
5    p_form = ProfileUpdateForm(request.POST, request.FILES, instance=request.user
         .profile)
6
7    if u_form.is_valid() and p_form.is_valid():
8     u_form.save()
9     p_form.save()
10    messages.success(request, f'Your account has been updated!')
11    return redirect('profile')
12  else:
13   u_form = UserUpdateForm(instance=request.user)
14   p_form = ProfileUpdateForm(instance=request.user.profile)
15
16  context = {'u_form':u_form, 'p_form':p_form}
17
18  return render(request, 'users/profile.html', context)
```

Στη γραμμή 1 ορίζεται ένας 'django decorator', ο οποίος εμποδίζει κάθε αίτημα προς την αντίστοιχη σελίδα (στη συγκεκριμένη περίπτωση για τη σελίδα /profile/) εάν ο χρήστης δεν είναι πιστοποιημένος, δηλαδή αν δεν έχει συνδεθεί.

4.4 Εισαγωγή/Εξαγωγή προϊόντων στο/από το ψυγείο

Η εφαρμογή χρειάζεται να γνωρίζει τα προϊόντα που υπάρχουν στο ψυγείο. Για να επιτευχθεί αυτό, ο χρήστης πρέπει με κάποιον τρόπο να ενημερώνει το ψυγείο για γεγονότα εισαγωγής κι εξαγωγής προϊόντων. Οι δύο αυτές ενέργειες γίνονται με παρόμοιο τρόπο:

1. εισαγωγή μέσω του barcode σαρωτή εισαγωγής
2. εξαγωγή μέσω του barcode σαρωτή εξαγωγής, είτε χειροκίνητα με πάτημα κουμπιού

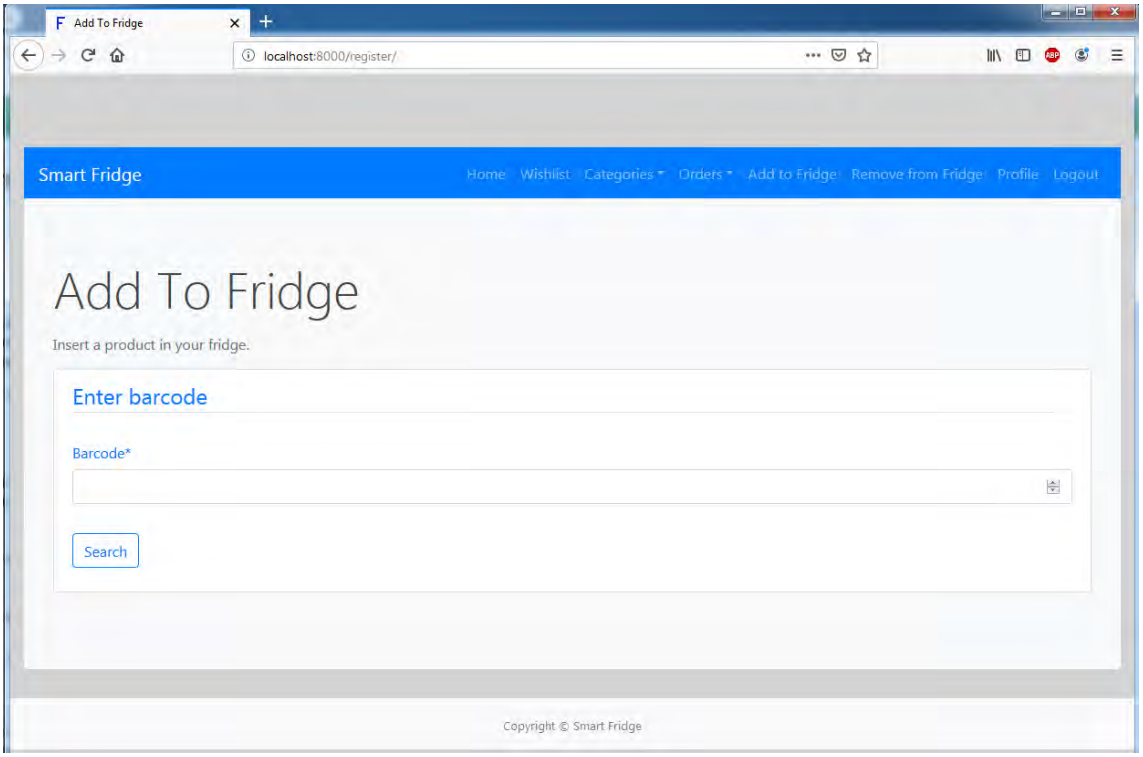
4.4.1 Εισαγωγή προϊόντων

Για να προσθέσουμε προϊόντα στο ψυγείο, πηγαίνουμε στη σελίδα **Add to Fridge** (εικόνα 4.6) του μενού. Η εισαγωγή γίνεται μέσω του barcode σαρωτή. Ο χρήστης σαρώνει τον αριθμό barcode του προϊόντος προς εισαγωγή στον barcode σαρωτή εισαγωγής κι αυτομάτως γράφεται στη φόρμα, ώστε να γίνει αναζήτηση του αριθμού στον πίνακα της βάσης που περιέχει όλα τα προϊόντα, για την επαλήθευση του συγκεκριμένου προϊόντος.

Η **HomeForm** Django form που είναι υπεύθυνη για την είσοδο του barcode (UI/forms.py):

```
1  class HomeForm(forms.Form):
2   barcode = forms.IntegerField()
```

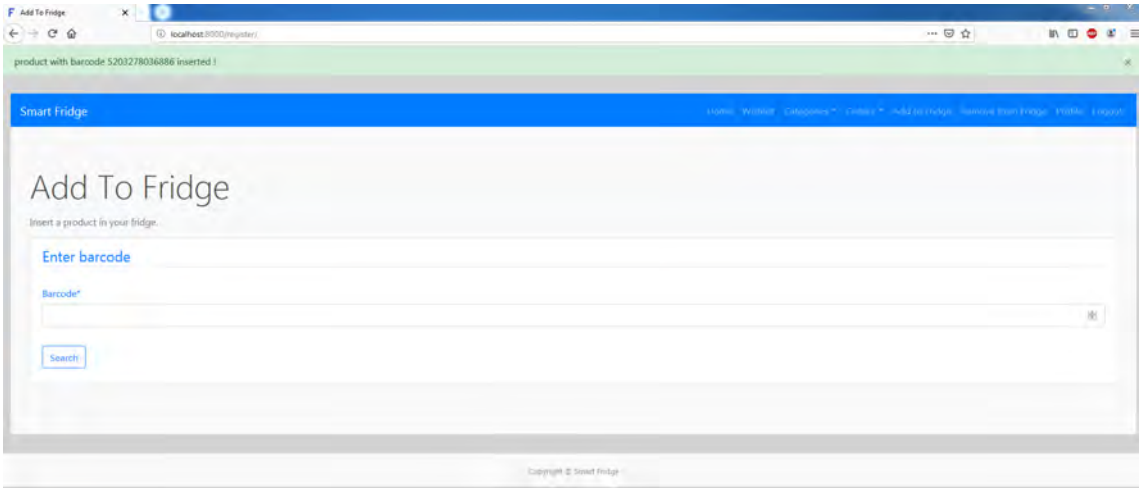
View function της σελίδας Add to Fridge (UI/views.py):



Σχήμα 4.6: Σελίδα Add to Fridge.

```
1  @login_required
2  def register(request):
3   current_user = request.user
4
5   if request.method == 'POST':
6    form = HomeForm(request.POST)
7    if form.is_valid():
8     barcode = form.cleaned_data.get('barcode')
9     if Barcode.objects.filter(item_id=barcode).exists():
10     bar_product = Barcode.objects.filter(item_id=barcode).first()
11     new_product = Userfridgelist(barcode=bar_product, user=current_user)
12     new_product.save()
13     messages.success(request, f'product with barcode {barcode} inserted !')
14    else:
15     messages.warning(request, f'product with barcode {barcode} does not exist !
           ')
16    return redirect('UI-register')
17   else:
18    form = HomeForm()
19
20
21   context = {'form':form, 'title':'Register'}
22   return  render(request, 'UI/register.html', context)
```

Αν υπάρχει προϊόν στη βάση δεδομένων με το barcode που έδωσε ο χρήστης ως είσοδο (γραμμή 9), τότε παίρνουμε αυτό το barcode αντικείμενο από τον πίνακα Barcode (γραμμή 10) και με βάση



Σχήμα 4.7: Εισαγωγή προϊόντος με barcode 5203278036886.

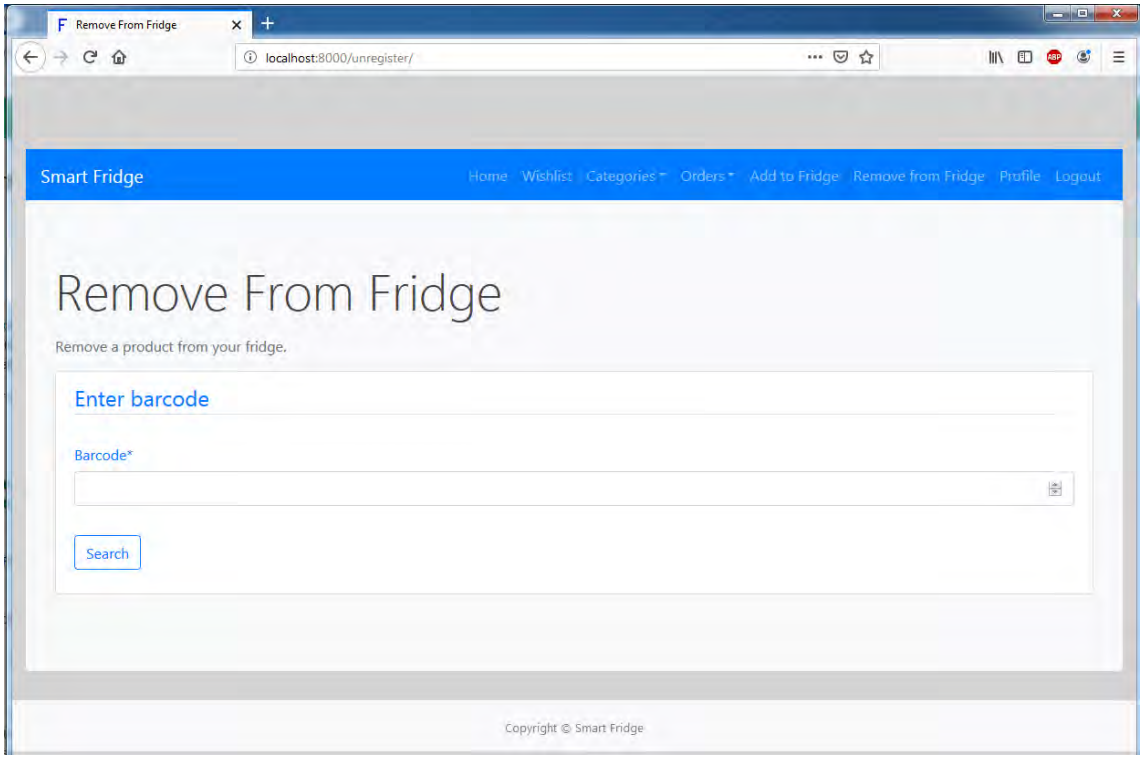
αυτό, δημιουργούμε ένα καινούριο Userfridgelist αντικείμενο, το οποίο περιέχει ως πληροφορία το προϊόν και τον χρήστη με τον οποίο έχει συσχέτιση (γραμμή 11) και το αποθηκεύουμε στον πίνακα Userfridgelist (γραμμή 12). Αν το barcode δεν αντιστοιχεί σε κάποιο προϊόν της βάσης δεδομένων, τότε εμφανίζεται κατάλληλο μήνυμα στον χρήστη (γραμμή 15). Μετά την επιτυχημένη εισαγωγή ενός προϊόντος (εικόνα 4.7), εμφανίζεται κατάλληλο μήνυμα στην οθόνη.

4.4.2 Εξαγωγή προϊόντων

Ο ρόλος της σελίδας **Remove from Fridge** (εικόνα 4.8) είναι η εξαγωγή προϊόντων από το ψυγείο. Η διαδικασία είναι η ίδια με την εισαγωγή προϊόντων στο ψυγείο, με τη διαφορά ότι ο χρήστης σαρώνει το προϊόν εξαγωγής στον barcode σαρωτή που είναι υπεύθυνος για την απομάκρυνση προϊόντων. Επειδή ο αριθμός barcode δεν είναι μοναδικός, στην περίπτωση που ο χρήστης έχει πάνω από ένα τρόφιμα με τον ίδιο αριθμό barcode στο ψυγείο, ο αλγόριθμος εξαγωγής θα αφαιρέσει το πιο παλιό προϊόν βάση ημερομηνίας εισαγωγής (αυτό που εισήχθει πρώτο). Σε περίπτωση που ο χρήστης δεν επιθυμεί να αφαιρέσει το πιο παλιό, μπορεί να κάνει την εξαγωγή προϊόντος από τη σελίδα Home χειροκίνητα, διαλέγοντας ο ίδιος ποιό ακριβώς προϊόν επιθυμεί να απομακρύνει.

View function της σελίδας Remove from Fridge (UI/views.py):

```
1  @login_required
2  def unregister(request):
3
4   current_user = request.user
5   current_userfridgelist = Userfridgelist.objects.filter(user=current_user).all
        ()
6
7   if request.method == 'POST':
8    form = HomeForm(request.POST)
9    if form.is_valid():
10    barcode = form.cleaned_data.get('barcode')
```



Σχήμα 4.8: Σελίδα Remove from Fridge.

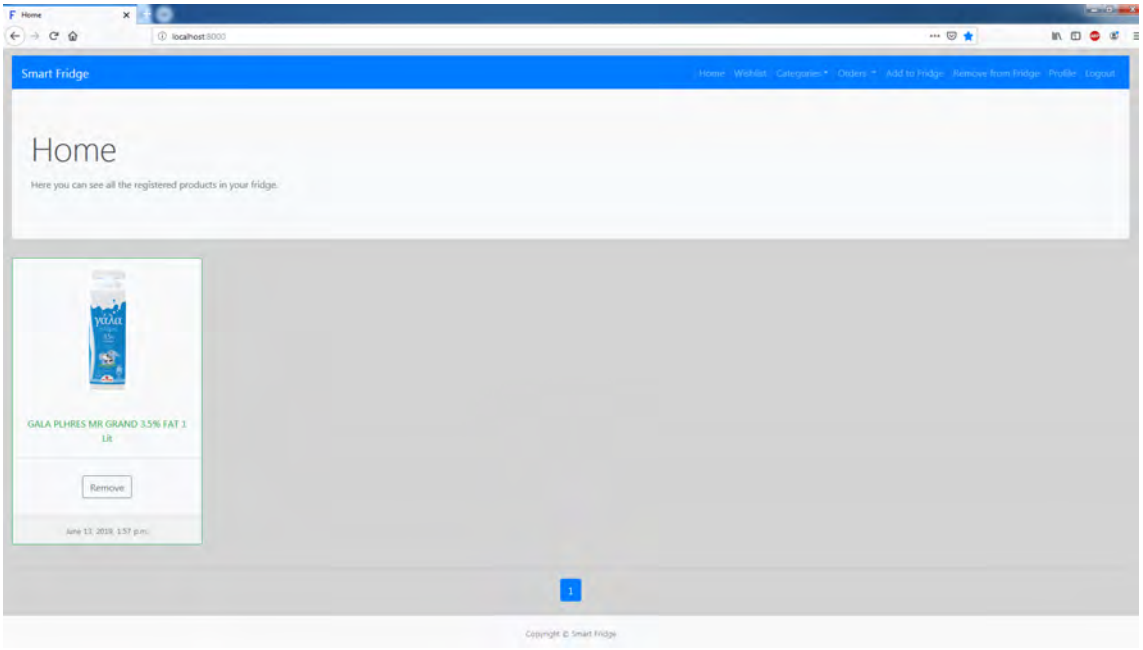
```
11      if Barcode.objects.filter(item_id=barcode).exists():
12          bar_product = Barcode.objects.filter(item_id=barcode).last()
13          current_userfridgelist.filter(barcode=bar_product).first().delete()
14          messages.success(request, f'barcode removed {barcode}!')
15      else:
16          messages.warning(request, f'barcode {barcode} does not exist !')
17      return redirect('UI-unregister')
18  else:
19      form = HomeForm()
20
21
22  context = {'form':form, 'title':'Remove From Fridge'}
23  return  render(request, 'UI/unregister.html', context)
```

4.4.3 Προβολή προϊόντων του ψυγείου - Αρχική σελίδα Home

Η σελίδα **Home** είναι η αρχική σελίδα της εφαρμογής και σκοπός της είναι να εμφανίζει όλα τα προϊόντα που έχει ο χρήστης στο ψυγείο του, μαζί με την ημερομηνία εισαγωγής (εικόνα 4.9). Κάθε φορά που ο χρήστης συνδέεται στην εφαρμογή, ανακατευθύνεται στην αρχική σελίδα Home.

Για να εμφανίσουμε τα προϊόντα στη σελίδα, παίρνουμε όλα τα αντικείμενα του πίνακα User-fridgelist για τον συγκεκριμένο χρήστη και τα τυπώνουμε με σειρά ανάλογη της ημερομηνίας εισαγωγής:

Listing 4.1: display products - home view - UI/views.py



Σχήμα 4.9: Σελίδα Home - Αρχική σελίδα.

```
1       current_user = request.user
2
3       current_userfridgelist = Userfridgelist.objects.filter(user=current_user).
            all()
4   queryset_list = current_userfridgelist.all().order_by("-insert_date")
```

Επίσης, κάθε προϊόν έχει κι ένα έγχρωμο περίγραμμα (εικόνα 4.10)

1. πράσινο: αν το προϊόν είναι φρέσκο
2. πορτοκαλί: αν το προϊόν πλησιάζει την ημερομηνία λήξης
3. κόκκινο: αν το προϊόν έχει περάσει την ημερομηνία λήξης

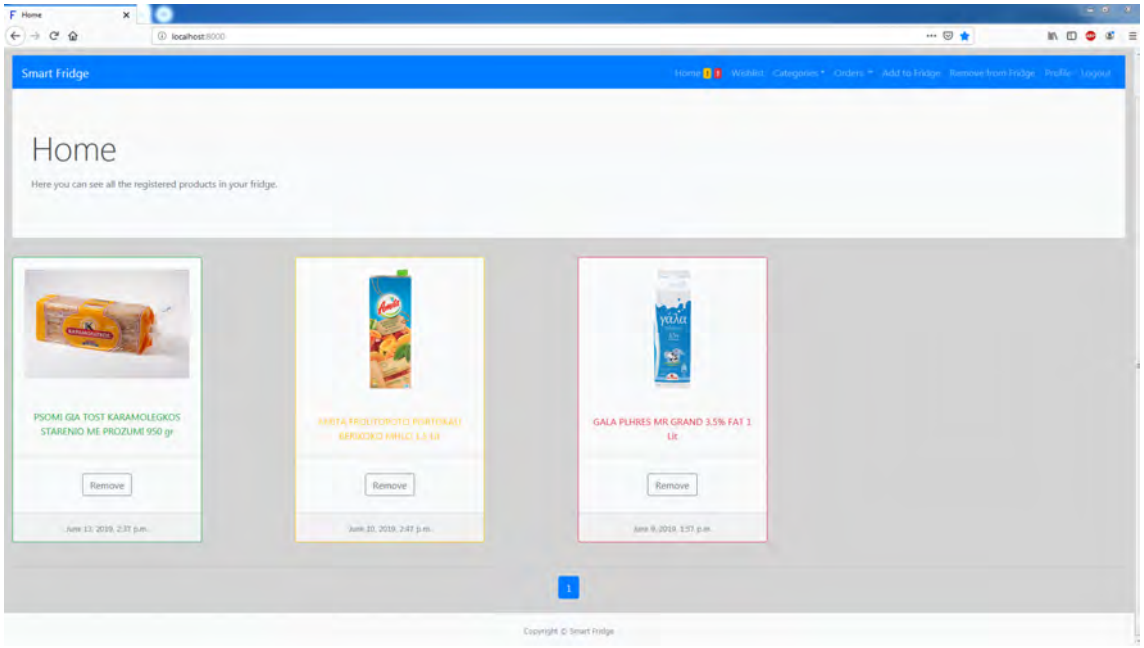
Η εφαρμογή ειδοποιεί τον χρήστη αν στο ψυγείο υπάρχουν προϊόντα που κοντεύουν να λήξουν ή έχουν ήδη λήξει με ένα θαυμαστικό στο μενού, δίπλα στο κουμπί Home (εικόνα 4.10).

Τέλος, ο χρήστης μπορεί να αφαιρέσει χειροκίνητα προϊόντα από το ψυγείο του. Είδαμε στην προηγούμενη ενότητα ότι η αφαίρεση προϊόντων γίνεται στο Remove from Fridge με τον barcode σαρωτή, αφαιρώντας όμως το πιο παλιό προϊόν. Εδώ δίνεται η δυνατότητα στον χρήστη να απομακρύνει όποιο προϊόν θελήσει, με το πάτημα του κουμπιού Remove, χωρίς να χρειάζεται να χρησιμοποιήσει τον barcode σαρωτή.

Για να απομακρύνουμε ένα προϊόν από το ψυγείο μέσω αιτήματος GET, βρίσκουμε το προϊόν (αν υπάρχει) στον πίνακα με τα προϊόντα που βρίσκονται στο ψυγείο του χρήστη, δηλαδή στον πίνακα Userfridgelist, με βάση το πρωτεύον κλειδί `item_id` και το διαγράφουμε.

Listing 4.2: remove product - home view - UI/views.py

```
1       product = request.GET.get("remove_product")
```

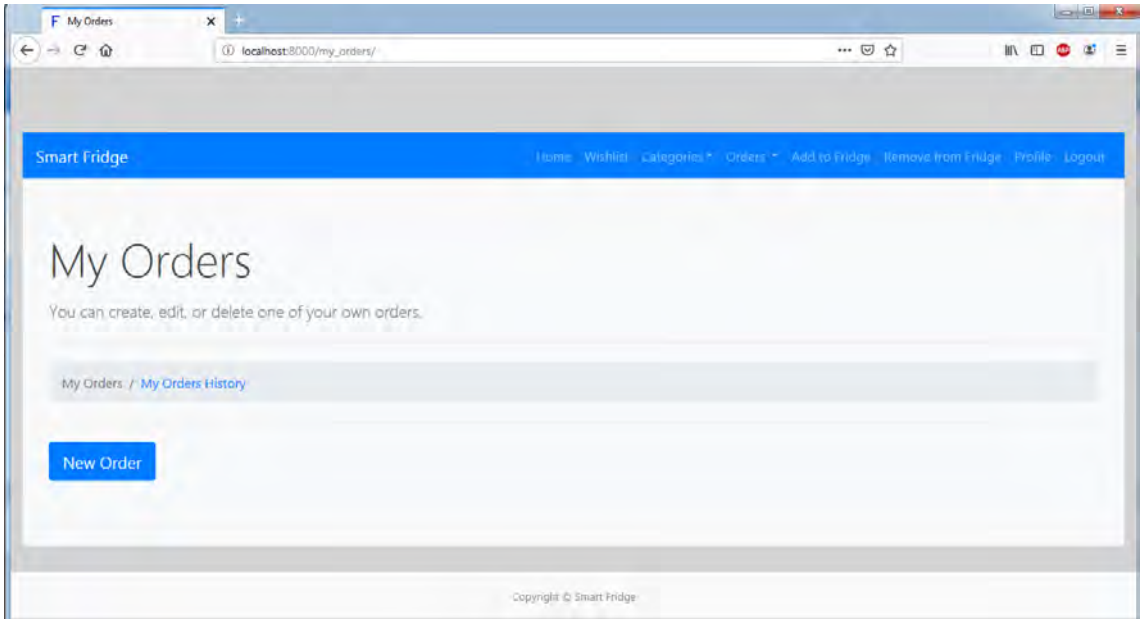


Σχήμα 4.10: Σελίδα Home

```
2    if product:
3     if current_userfridgelist.filter(item_id=product).exists():
4      current_userfridgelist.filter(item_id=product).first().delete()
5      messages.success(request, f'product deleted!')
6      return redirect('UI-home')
7     else:
8      messages.warning(request, f'an unexpected error has occured, please try
           again later ')
```

4.5 Ηλεκτρονικές παραγγελίες σε σουπερμάρκετ

Ο χρήστης έχει τη δυνατότητα να πραγματοποιήσει παραγγελίες που αποστέλονται ηλεκτρονικά σε σουπερμάρκετ μέσω της εφαρμογής. Κάθε παραγγελία είναι ουσιαστικά μια λίστα προϊόντων σε μορφή κειμένου η οποία αποστέλλεται στο συνεργαζόμενο σουπερμάρκετ. Ο χρήστης έχει τη δυνατότητα να δημιουργήσει μόνος του μια ή περισσότερες παραγγελίες, επιλέγοντας ο ίδιος τα προϊόντα που επιθυμεί, διαφορετικά μπορεί να χρησιμοποιήσει την αυτόματη παραγγελία, για την οποία ορίζει ημέρα κι ώρα αποστολής, κι αποστέλλεται αυτόματα (μετά από την έγκρισή του). Στα πλαίσια αυτής της εργασίας καμμία παραγγελία δεν αποστέλλεται σε πραγματικό σουπερμάρκετ, καθώς δεν υπάρχει κάποιο εργαλείο επικοινωνίας (API) μεταξύ της εφαρμογής και της ιστοσελίδας του σουπερμάρκετ, οπότε υποθέτουμε ότι όλες οι παραγγελίες αποστέλλονται σε κάποιο εικονικό κατάστημα.



Σχήμα 4.11: My Orders page

4.5.1 Παραγγελίες χρήστη

Ο χρήστης μπορεί να δημιουργήσει μια ή περισσότερες παραγγελίες στη σελίδα My Orders (εικόνα 4.11) πατώντας το κουμπί `New Order`.

Κατά τη δημιουργία μιας παραγγελίας, δημιουργούμε ένα καινούριο Order αντικείμενο (πίνακας 4.7):

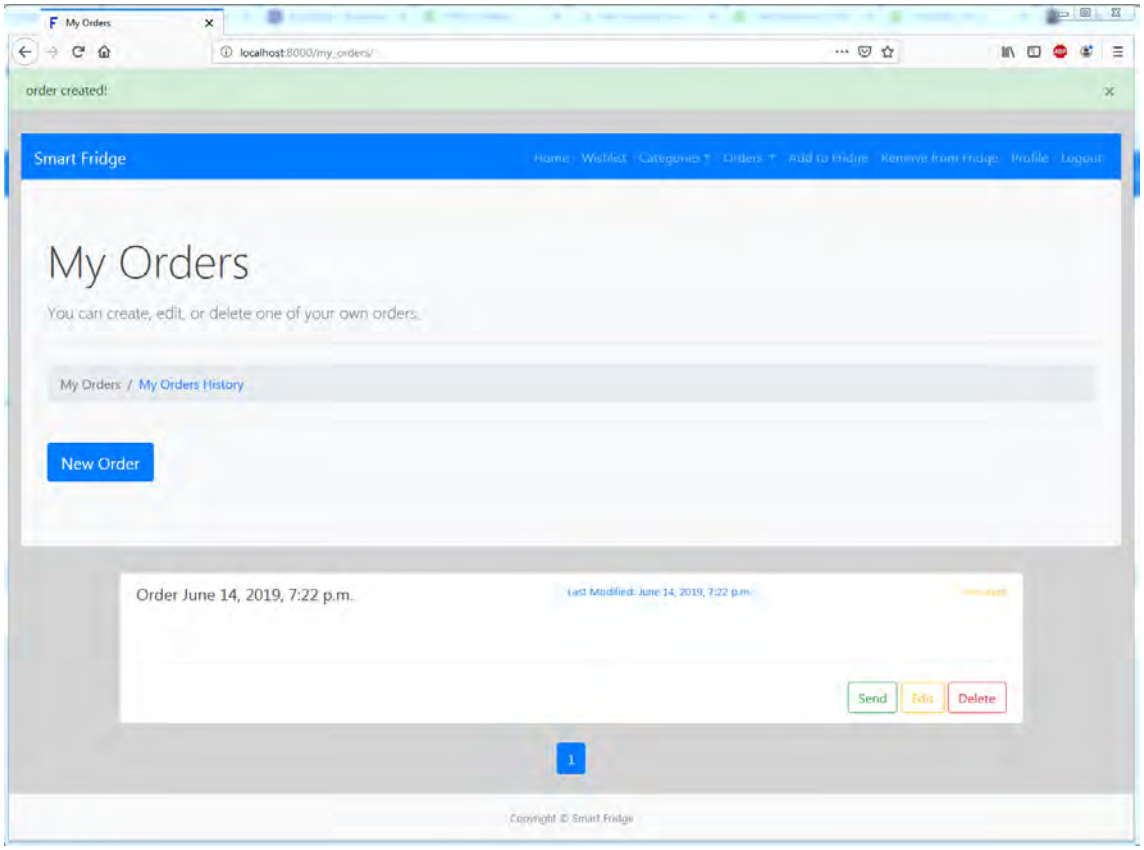
Listing 4.3: create order - MyOrders view - UI/views.py

```
1 q_new_order = request.GET.get("new_order")
2  if q_new_order:
3   new_order = Order.objects.create(user=current_user)
4   messages.success(request, f'order created!')
5   return redirect('UI-my_orders')
```

Κάθε παραγγελία που δημιουργείται αρχικά είναι άδεια και διακρίνεται από την ημερομηνία δημιουργίας της. Ο χρήστης έχει τις επιλογές να επεξεργαστεί (Edit), αποστείλει (Send) ή να διαγράψει μια παραγγελία. Επίσης, για κάθε παραγγελία φαίνονται οι ημερομηνίες τελευταίας τροποποίησης κι αποστολής (εφόσον έχει αποσταλεί, αλλιώς εμφανίζεται `Not sent`) (εικόνα 4.12). Ο χρήστης μπορεί να στείλει πάνω από μια φορά την ίδια παραγγελία εφόσον δεν την έχει διαγράψει. Η δυνατότητα πολλών παραγγελιών βοηθά τον χρήστη να αποθηκεύει πολλαπλές λίστες με προϊόντα τις οποίες θέλει να θυμάται. Πατώντας Edit, ο χρήστης ανακατευθείνεται στη σελίδα με όλες τις κατηγορίες προϊόντων, ώστε να βρει το προϊόν που θέλει βάση της κατηγορίας που ανήκει (εικόνα 4.13).

Ο χρήστης, πέρα από το κουμπί Edit, μπορεί να προσθέσει προϊόντα σε μια παραγγελία της επιλογής του με περιήγηση σε μια από τις κατηγορίες προϊόντων από το μενού (`Categories`).

Για να εισάγουμε ένα προϊόν σε μια παραγγελία δημιουργούμε ένα καινούριο OrderItem, με



Σχήμα 4.12: Σελίδα My Orders - Νέα παραγγελία

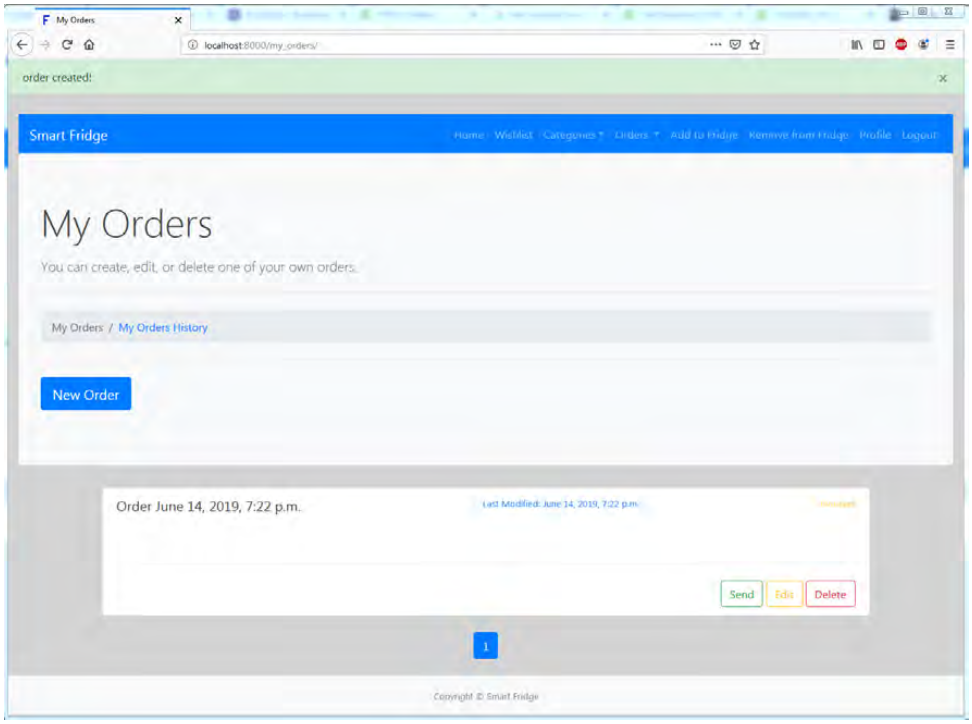
βάση το αναγνωριστικό της παραγγελίας που γίνεται η προσθήκη (`order_id`) και το barcode του προϊόντος (`prod_barcode`):

Listing 4.4: add product

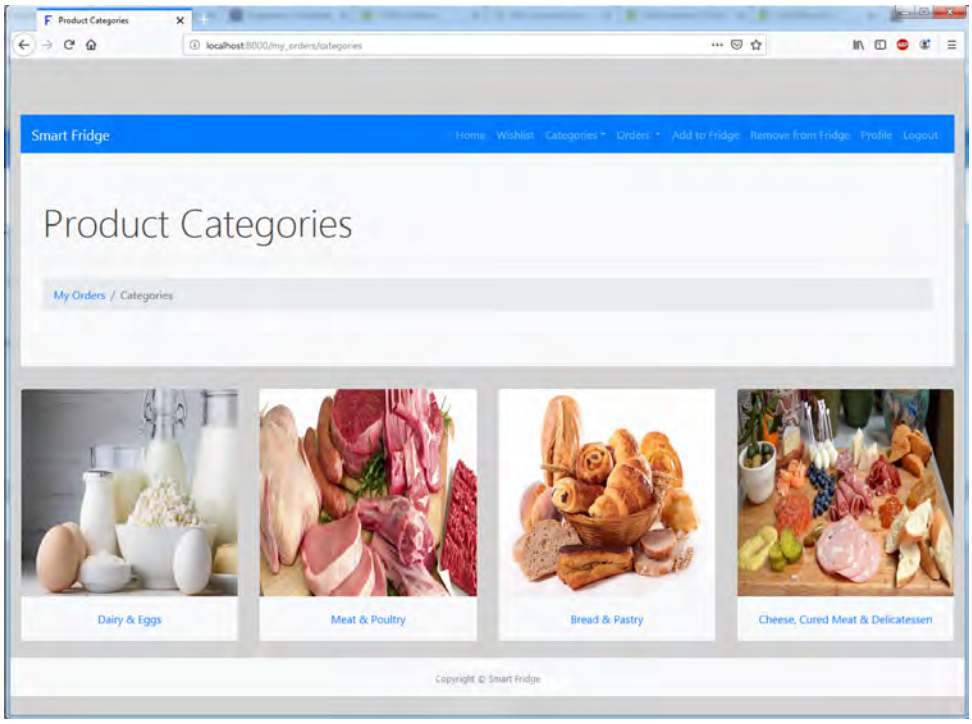
```
1  product = request.GET.get("add_to_order")
2   if product:
3     query_values_list = product.split(" ")
4     prod_barcode = query_values_list[0]
5     order_id = query_values_list[1]
6
7     product_in_barcode = Barcode.objects.filter(item_id=prod_barcode).first()
8     current_order = Order.objects.filter(order_id=order_id).first()
9     item_in_order = OrderItem(order=current_order, product=product_in_barcode)
10    item_in_order.save()
11    now = timezone.localtime(timezone.now())
12    Order.objects.filter(order_id=order_id).update(last_modified=now)
```

Όπως φαίνεται στην εικόνα 4.13δ΄, με τα κουμπιά δεξιά από κάθε προϊόν, ο χρήστης μπορεί να αυξήσει ή να μειώσει την ποσότητα ενός προϊόντος, ή να το διαγράψει.

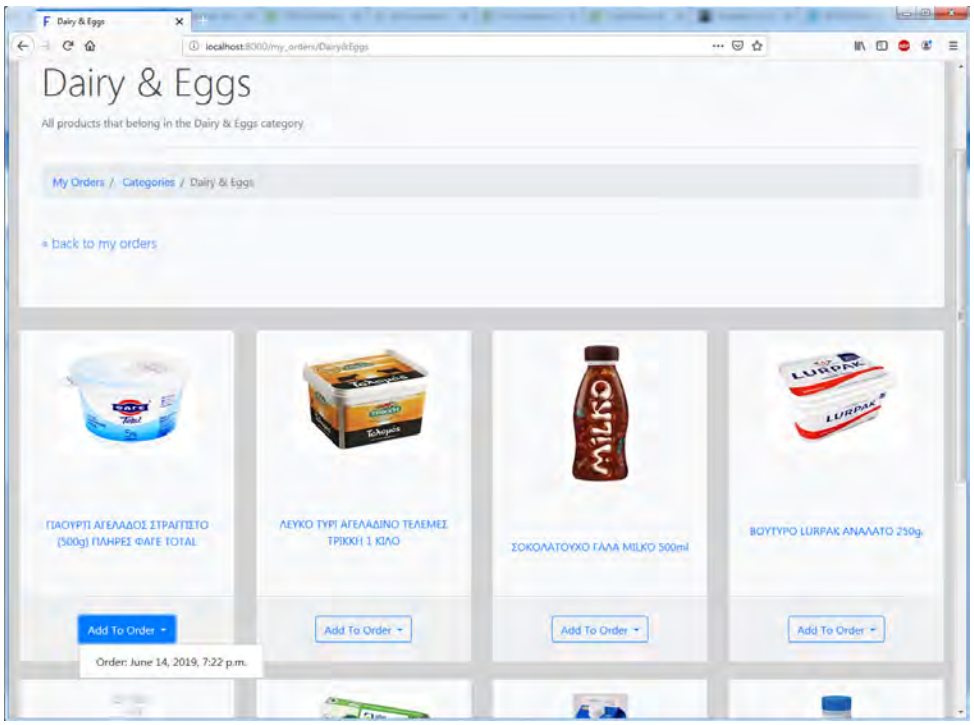
Ο χρήστης μπορεί να δει όλες τις παραγγελίες που έχει στείλει στη σελίδα **My Orders History**, όπου υπάρχει το ιστορικό παραγγελιών. Επίσης, μπορεί να προσθέσει όλα τα προϊόντα μιας λίστας απευθείας στο ψυγείο, χωρίς να χρειαστεί να περάσει ένα-ένα τα προϊόντα της παραγγελίας μέσω



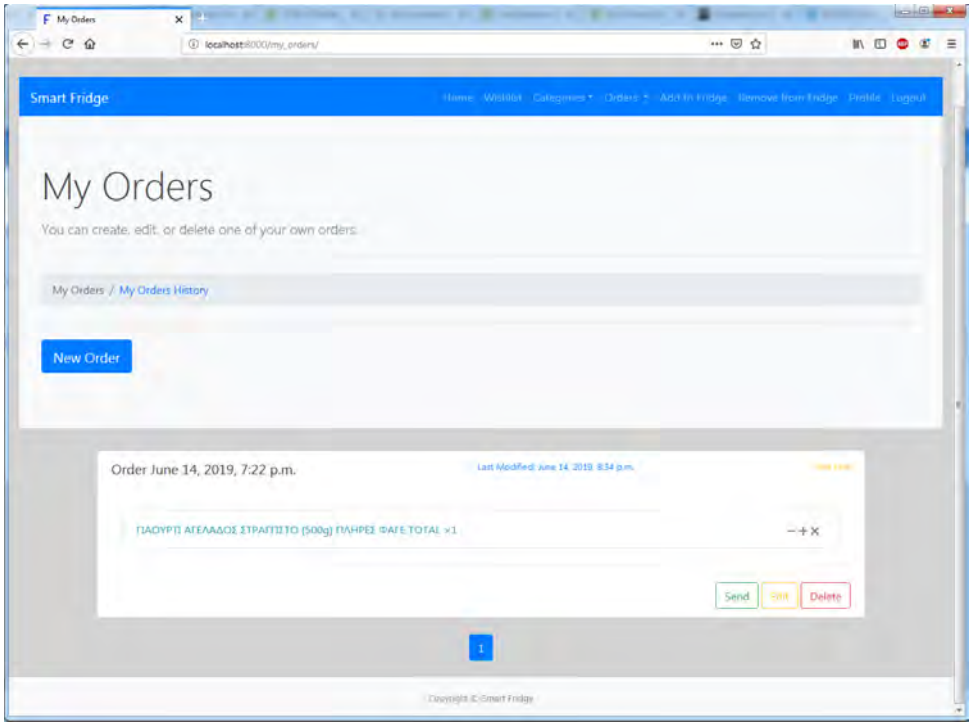
(α΄) Επεξεργασία παραγγελίας



(β΄) Επιλογή κατηγορίας

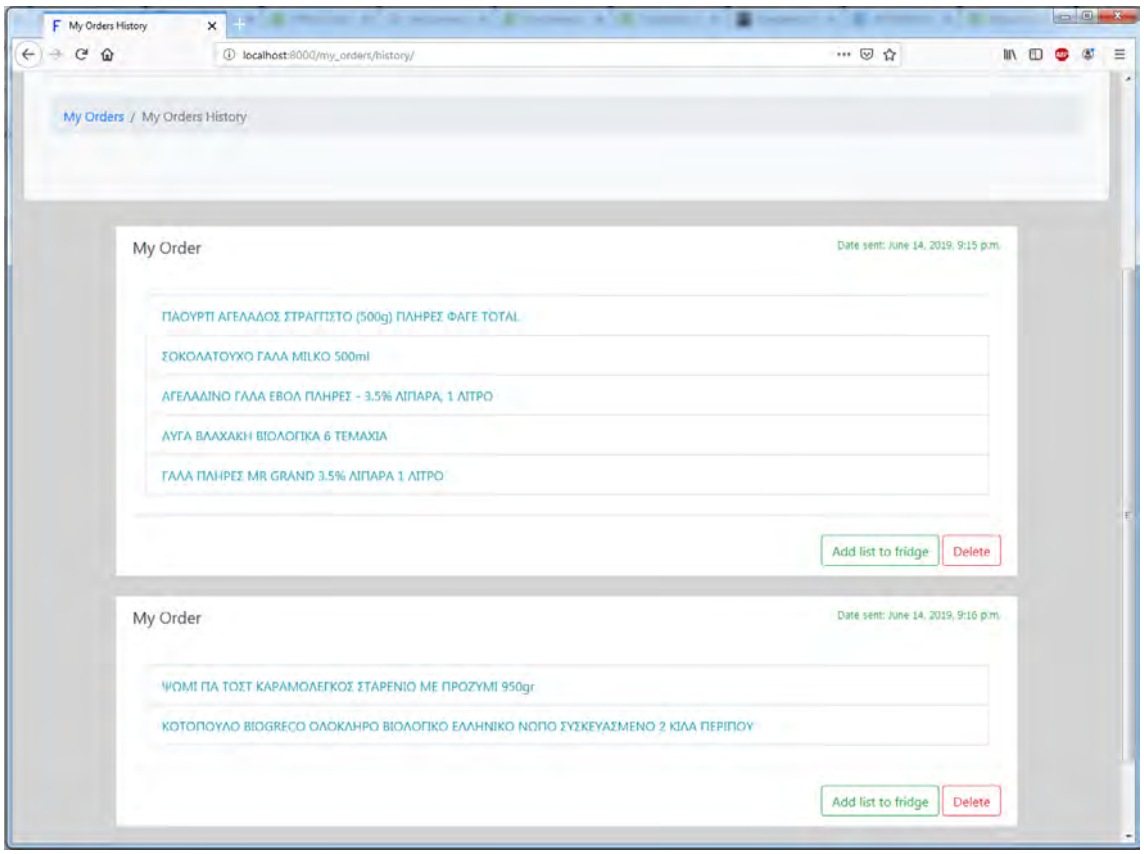


(γ΄) Προσθήκη προϊόντος στην παραγγελία



(δ΄) Πίσω στις παραγγελίες

Σχήμα 4.13: Εισαγωγή προϊόντος σε παραγγελία



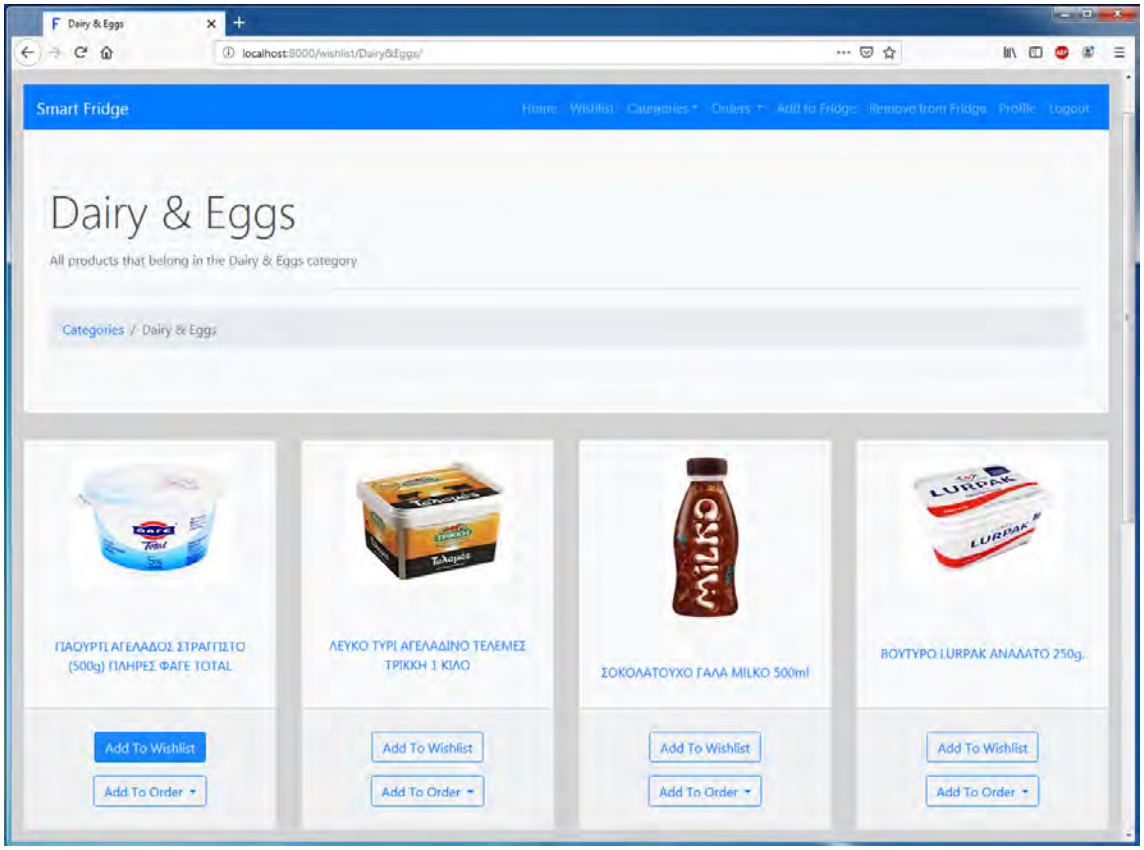
Σχήμα 4.14: Ιστορικό παραγγελιών - My Orders

του barcode σαρωτή (εικόνα 4.14).

Το αντικείμενο OrderHistory, που στην ουσία είναι το ιστορικό των παραγγελιών, δημιουργείται μετά την αποστολή της παραγγελίας. Επίσης, κάθε ένα προϊόν που υπήρχε στη λίστα της παραγγελίας αποθηκεύεται ως ένα OrderItemHistory αντικείμενο.

Listing 4.5: order history

```
1  q = request.GET.get("send_order")
2   if q:
3    now = timezone.localtime(timezone.now())
4
5    Order.objects.filter(user=current_user, order_id=q).update(date_sent=now)
6
7    new_order_history = OrderHistory.objects.create(user=current_user, orderType=
         'my_order', date_sent=now)
8
9    order_items = OrderItem.objects.filter(order_id=q).all()
10   for order_item in order_items:
11    for quantity in range(order_item.quantity):
12     new_product_history = OrderItemHistory(product=order_item.product, order=
           new_order_history)
13     new_product_history.save()
```



Σχήμα 4.15: Προσθήκη προϊόντων στη λίστα επιθυμιών

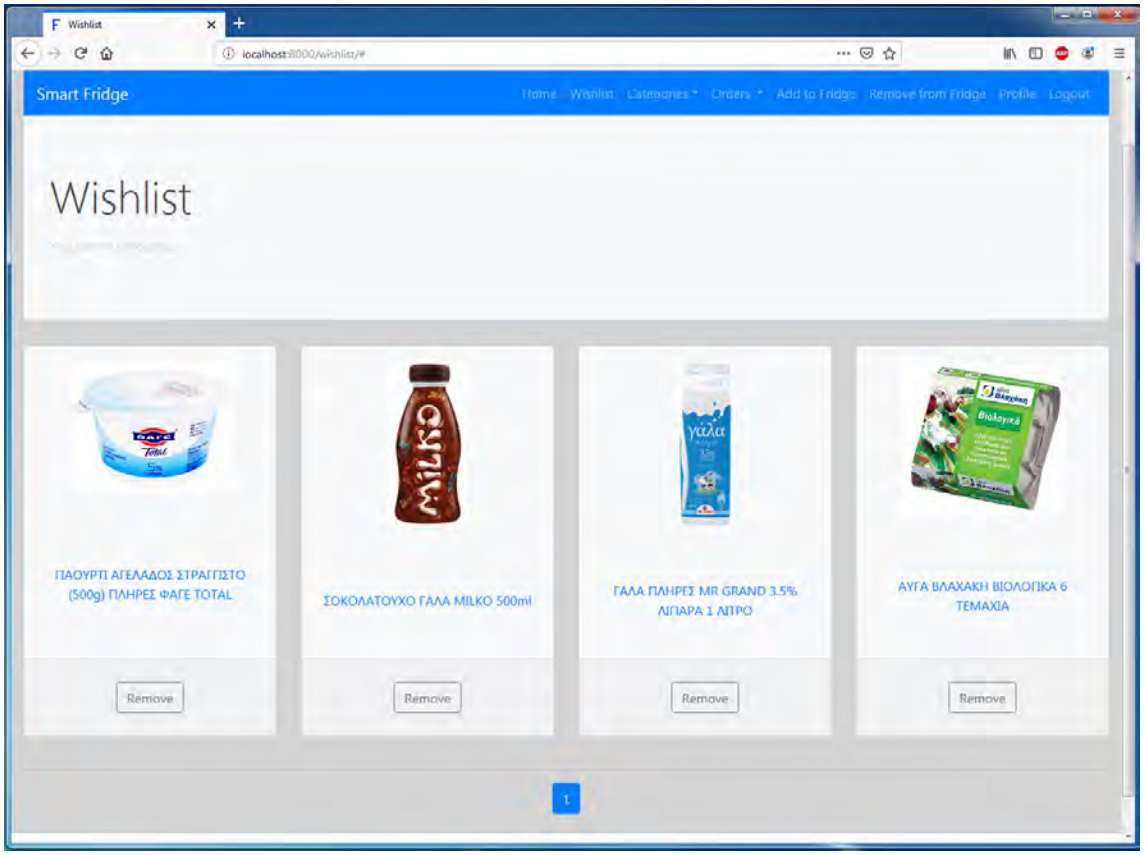
4.5.2 Λίστα επιθυμιών

Ο χρήστης μπορεί να προσθέσει τα προϊόντα της επιλογής του στη λίστα επιθυμιών της εφαρμογής για διάφορους λόγους, είτε για να τα θυμάται, είτε για να έχει γρήγορη πρόσβαση σε αυτά. Από άποψη λειτουργικότητας της εφαρμογής, η φιλοσοφία είναι ότι τα προϊόντα που υπάρχουν στη λίστα επιθυμιών δεν πρέπει να λείπουν από το ψυγείο του χρήστη. Ο χρήστης μπορεί να προσθέσει αντικείμενα στη λίστα επιθυμιών από τις κατηγορίες προϊόντων, πατώντας το κουμπί `Add to wishlist` για το προϊόν της επιθυμίας του (εικόνα 4.15). Στη σελίδα Wishlist ο χρήστης μπορεί να δει όλα τα προϊόντα που έχει προσθέσει στη λίστα επιθυμιών (εικόνα 4.16), από όπου μπορεί να αφαιρέσει όποιο θέλει πατώντας το κουμπί `Remove`.

Η προσθήκη προϊόντος στη λίστα επιθυμιών ισοδυναμεί με προσθήκη προϊόντος στον πίνακα Userproductlist (πίνακας 4.6)

Listing 4.6: Προσθήκη προϊόντος στη λίστα επιθυμιών

```
1 product = request.GET.get("add_product")
2  if product:
3   product_in_barcode = Barcode.objects.filter(item_id=product).first()
4   product_in_wishlist = Userproductlist(product=product_in_barcode, user=
        current_user)
5   product_in_wishlist.save()
6   messages.success(request, f'product added!')
```



Σχήμα 4.16: Σελίδα Wishlist

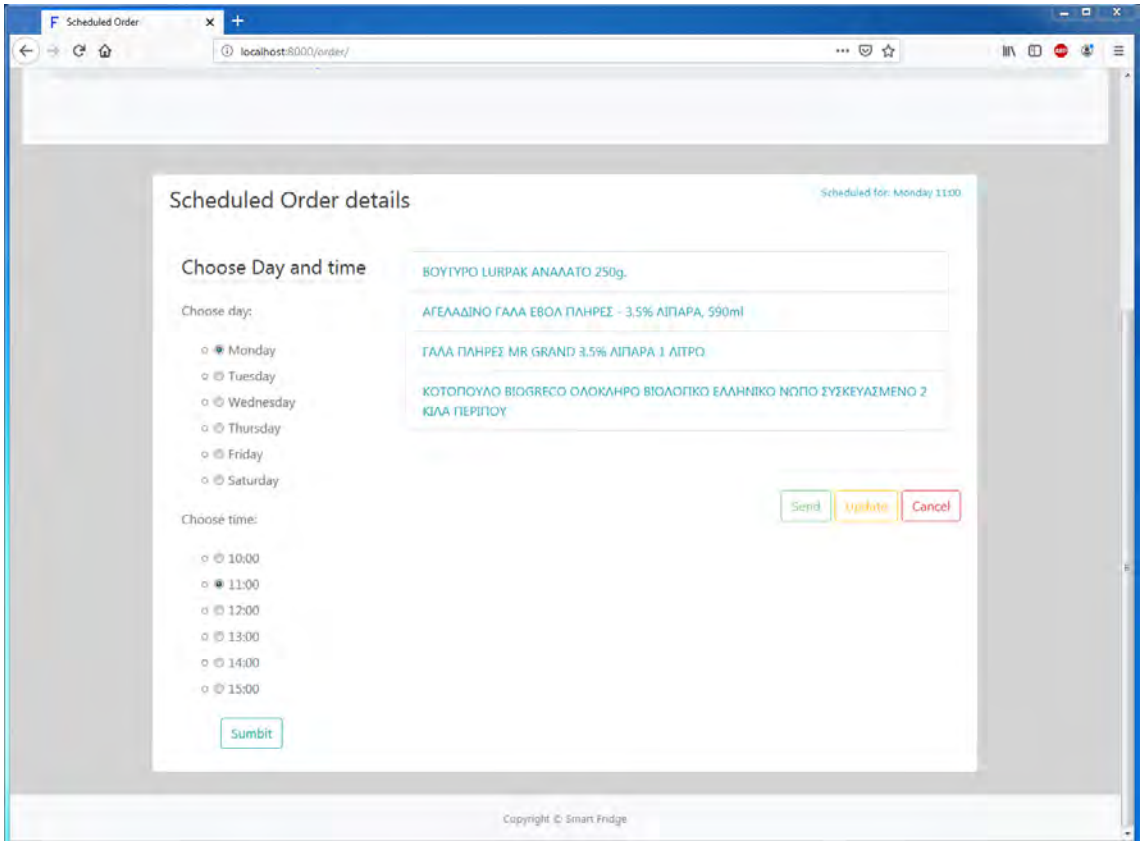
Η αφαίρεση προϊόντος από τη λίστα επιθυμιών ισοδυναμεί με την διαγραφή προϊόντος από τον πίνακα Userproductlist (πίνακας 4.6)

Listing 4.7: Αφαίρεση προϊόντος από τη λίστα επιθυμιών

```
1 product = request.GET.get("remove_product")
2  if product:
3   product_in_barcode = Barcode.objects.filter(item_id=product).first()
4   if current_userproductlist.filter(product=product_in_barcode).exists():
5    current_userproductlist.filter(product=product_in_barcode).first().delete()
6    messages.success(request, f'product removed!')
```

4.5.3 Αυτόματες Παραγγελίες

Η εφαρμογή έχει τη δυνατότητα να ετοιμάσει μια παραγγελία προς ένα σουπερμάρκετ αυτόματα, και να την αποστείλει με την έγκριση του χρήστη. Αυτή η παραγγελία έχει δημιουργηθεί αυτόματα με τη δημιουργία λογαριασμού, όπως συζητήθηκε σε προηγούμενα κεφάλαια, χωρίς όμως να έχει οριστεί ημερομηνία αποστολής. Η ημέρα κι ώρα αποστολής ορίζονται από τον χρήστη στη σελίδα Scheduled Order της εφαρμογής (4.17). Μια μόνο αυτόματη παραγγελία μπορεί να σταλεί την ίδια ημέρα. Αν η παραγγελία σταλεί την ημέρα που επέλεξε ο χρήστης, τότε θα σταλεί πάλι την επόμενη εβδομάδα, την ίδια ημέρα κι ώρα.



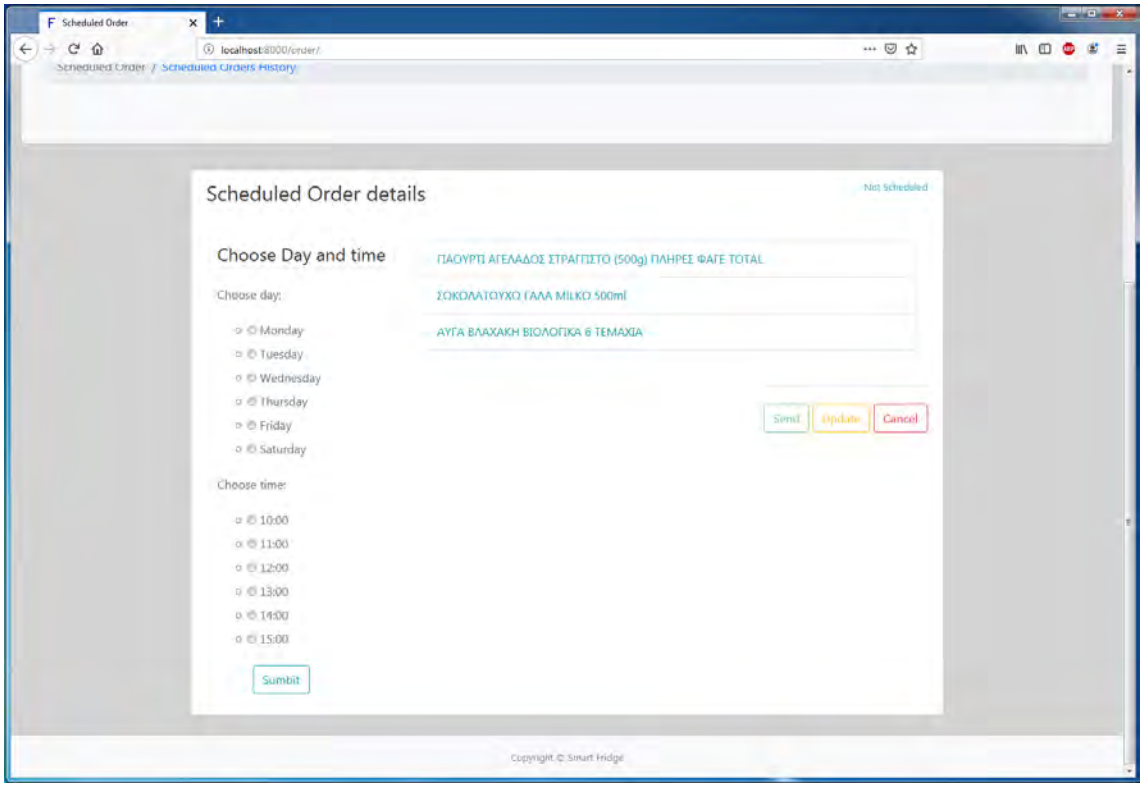
Σχήμα 4.17: Προγραμματισμένη παραγγελία - σελίδα scheduled order

Το περιεχόμενο της αυτόματης παραγγελίας καθορίζεται από τα προϊόντα που υπάρχουν στη λίστα επιθυμιών, αλλά λείπουν από το ψυγείο. Αν υποθέσουμε ότι τα προϊόντα του ψυγείου είναι αυτά που φαίνονται στην εικόνα 4.9 και τα προϊόντα της λίστας επιθυμιών είναι αυτά της εικόνας 4.16, τότε η λίστα της παραγγελίας διαμορφώνεται όπως φαίνεται στην εικόνα 4.18. Η λίστα αυτή με τα προϊόντα ανανεώνεται χειροκίνητα με το κουμπί `Update`, ή αυτόματα όταν η παραγγελία είναι έτοιμη για αποστολή.

Κατά την ανανέωση της λίστας της αυτόματης παραγγελίας, ως προϊόντα της λίστας διαλέγουμε εκείνα τα οποία ανήκουν στην λίστα επιθυμιών (πίνακας Userproductlist) αλλά δεν υπάρχουν στο ψυγείο (πίνακας Userfridgelist).

Listing 4.8: Ανανέωση λίστας αυτόματης παραγγελίας

```
1  current_user_orderitems = AutoOrderItem.objects.filter(order=current_user_order
       ).all()
2  q = request.GET.get("update_order")
3   if q:
4    for auto_order_item in current_user_orderitems:
5     auto_order_item.delete()
6
7    current_user_productlist = Userproductlist.objects.filter(user=current_user).
         values_list('product')
8    current_user_fridgelist = Userfridgelist.objects.filter(user=current_user).
```



Σχήμα 4.18: Προγραμματισμένη παραγγελία

```
        values_list('barcode')
9
10      c1 = Counter(current_user_productlist)
11      c2 = Counter(current_user_fridgelist)
12
13      diff = c1-c2
14      res = list(diff.elements())
15      items_list = [item for t in res for item in t]
16
17      current_user_order = AutoOrder.objects.filter(user=current_user,
            auto_order_id=q).first()
18      for product in items_list:
19       product_in_barcode = Barcode.objects.filter(id=product).first()
20       product_for_order = AutoOrderItem(order=current_user_order, product=
             product_in_barcode)
21       product_for_order.save()
```

Για να αποφασίσει η εφαρμογή αν πρέπει να σταλεί η αυτόματη παραγγελία, πρέπει να κάνει έναν περιοδικό έλεγχο, συγκρίνοντας την τωρινή ημερομηνία, με την ημερομηνία που έδωσε ο χρήστης σαν είσοδο. Αυτή η λειτουργικότητα ξεφεύγει από τα πλαίσια του μοντέλου αίτημα-απάντηση που συζητήσαμε σε προηγούμενο κεφάλαιο, καθώς ο έλεγχος πρέπει να γίνεται χωρίς να στείλει ο χρήστης κάποιο αίτημα και χωρίς να είναι απαραίτητα συνδεδεμένος στην εφαρμογή!

Ο περιοδικός έλεγχος γίνεται μέσω της συνάρτησης `tick()`, η οποία ελέγχει, για όλους τους χρήστες που έχουν προγραμματίσει μια παραγγελία, αν πρέπει να σταλεί ή όχι. Αν μια παραγγελία

κριθεί ότι πρέπει να σταλεί, τότε μπαίνει σε κατάσταση `ready` και περιμένει επιβεβαίωση από τον χρήστη για να σταλεί.

Listing 4.9: Περιοδικός έλεγχος αυτόματης παραγγελίας

```
def tick():
 all_users = User.objects.all()
 for user in all_users:

  now = timezone.localtime(timezone.now())
  current_day = now.strftime('%A')
  current_time = now.strftime('%H')

  current_date = now.strftime('%d/%m/%Y')

  user_order = AutoOrder.objects.filter(user=user).first()

  if user_order.scheduled_date is not None and user_order.scheduled_time is not
      None:

   if user_order.scheduled_date == current_day and user_order.scheduled_time ==
       current_time:

    if user_order.status != 'cancelled' and timezone.localtime(user_order.
       date_sent).strftime('%d/%m/%Y') != current_date and user_order.status
       != "ready":

     AutoOrder.objects.filter(user=user).update(status='ready')
```

Η επιβεβαίωση κι αποστολή της παραγγελίας γίνεται στη σελίδα Scheduled Order, ενώ η εφαρμογή ενημερώνει τον χρήστη με ειδοποιήσεις (εικόνα 4.19). Κατά την επιβεβαίωση της παραγγελίας, αυτή αποθηκεύεται αυτόματα στο ιστορικό. Ο χρήστης μπορεί ανά πάσα στιγμή να ακυρώσει την παραγγελία πατώντας το κουμπί `Cancel`.

Listing 4.10: Επιβεβαίωση - αποστολή - αποθήκευση ιστορικού παραγγελίας

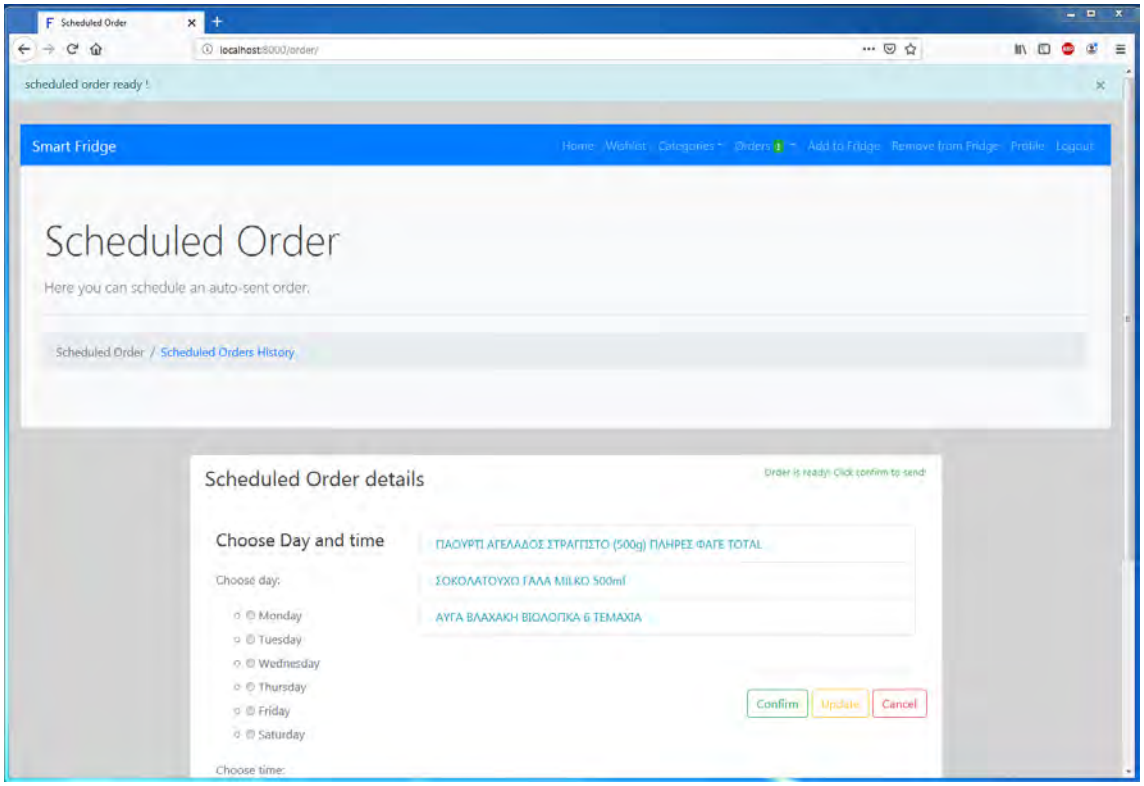
```
q = request.GET.get("confirm_order")
 if q:
  now = timezone.localtime(timezone.now())

  user_order = AutoOrder.objects.filter(user=current_user, auto_order_id=q).
      first()

  AutoOrder.objects.filter(user=current_user, auto_order_id=q).update(status='
      sent')
  AutoOrder.objects.filter(user=current_user, auto_order_id=q).update(date_sent
      =now)

  #add to history
  new_order_history = OrderHistory.objects.create(user=current_user, orderType=
      'auto', date_sent=now)

```



Σχήμα 4.19: Προγραμματισμένη παραγγελία έτοιμη για επιβεβαίωση κι αποστολή

```
13    auto_order_items = AutoOrderItem.objects.filter(order=user_order).all()
14    for auto_order_item in auto_order_items:
15     new_product_history = OrderItemHistory(product=auto_order_item.product,
           order=new_order_history)
16     new_product_history.save()
17    messages.success(request, f'Order sent succesfully! You can view your sent
          orders in the orders history.')
18    return redirect('UI-order')
```

Listing 4.11: Ακύρωση παραγγελίας

```
1  q = request.GET.get("cancel_order")
2   if q:
3    AutoOrder.objects.filter(user=current_user, auto_order_id=q).update(status="
         cancelled")
4    messages.success(request, f'Order cancelled !')
5    return redirect('UI-order')
```

Κεφάλαιο 5

Τεχνικές λεπτομέρειες

Ακολουθούν τεχνικές λεπτομέρειες για την εγκατάσταση των απαραίτητων εργαλείων και το τρέξιμο της διαδικτυακής εφαρμογής.

Ο κώδικας της εφαρμογής είναι διαθέσιμος στην ιστοσελίδα:
`https://github.com/androsos/Smart-Fridge.git`

5.1 Barcode To Pc - Barcode σαρωτής

Πρόκειται για μια εφαρμογή android που μετατρέπει το smartphone σε barcode σαρωτή. Ο χρήστης χρησιμοποιεί την κάμερα για να σαρώσει έναν αριθμό barcode, ο οποίος αποτελεί είσοδο στις σελίδες Add to Fridge και Remove from Fridge που αναλύσαμε σε προηγούμενο κεφάλαιο. Για την επικοινωνία της εφαρμογής android με την εφαρμογή δικτύου μας, είναι απαραίτητος ο barcode to pc διακομιστής. Ο προγραμματιστής μπορεί να βρει τον διακομιστή και να τον κατεβάσει από τη σελίδα της εφαρμογής barcodetopc [4], όπου θα βρει επίσης επιπλέον οδηγίες χρήσης.

5.2 Εγκατάσταση Django

Παρακάτω θα περιγράψουμε πώς μπορούμε να κάνουμε εγκατάσταση το django [10] στον υπολογιστή μας (σε windows OS) και να δημιουργήσουμε ένα νέο project. Απαραίτητη προϋπόθεση είναι να υπάρχει η python [21] προεγκατεστημένη στο μηχάνημά μας. Αρχικά, στο τερματικό μας με την εντολή

```
$ pip install virtualenvwrapper-win
```

κάνουμε εγκατάσταση ένα ειδικό περιβάλλον για κάθε project που δημιουργούμε στο django. Αν και δεν είναι απαραίτητο, προτείνεται αυτό το βήμα καθώς θα βοηθήσει αργότερα στην ανάπτυξη της εφαρμογής μας. Έπειτα δημιουργούμε ένα νέο περιβάλλον με την εντολή

```
$ mkvirtualenv myproject
```

και στη συνέχεια, για να επιλέξουμε αυτο το project, πληκτρολογούμε

```
$ workon myproject
```

Τώρα είμαστε έτοιμοι να **εγκαταστήσουμε το django**, με την εντολή

```
$ pip install django
```

Η δομή καταλόγου του django παρουσιάζεται στο σχήμα 2.4. Δημιουργούμε ένα django project:

```
$ python manage.py startproject project_name
```

Δημιουργούμε μια εφαρμογή για το django project μας:

```
$ python manage.py startapp app_name
```

Δημιουργούμε έναν διαχειριστή της εφαρμογής:

```
$ python manage.py createsuperuser
```

Για να ξεκινήσουμε τον διακομιστή:

```
$ python manage.py runserver
```

Αν θέλουμε να μπορούμε να χρησιμοποιήσουμε την εφαρμογή κι από μια άλλη συσκευή του ίδιου δικτύου, τότε τρέχουμε τον διακομιστή με την εντολή:

```
$ python manage.py runserver 0.0.0.0:8000
```

Ο χρήστης για να συνδεθεί από το κινητό του, θα χρειαστεί να δώσει σαν διεύθυνση στον φυλλομετρητή του ip:8000, όπου ip είναι η ip του μηχανήματος που τρέχει ο διακομιστής.

Κάθε φορά που κάνουμε αλλαγές στα μοντέλα μας, πρέπει να δημιουργήσουμε ένα νέο migration αρχείο στο django, το οποίο αποθηκεύει όλες τις αλλαγές που έγιναν την τελευταία φορά στα μοντέλα. Αυτό γίνεται με την εντολή

```
$ python manage.py makemigrations
```

Αφού δημιουργηθεί το migration αρχείο με τις αλλαγές που έγιναν στα μοντέλα, μπορούμε να εφαρμόσουμε αυτές τις αλλαγές πληκτρολογώντας

```
$ python manage.py migrate
```

Αυτή η λειτουργικότητα του django μας επιτρέπει να μπορούμε να κάνουμε αλλαγές στη βάση δεδομένων μας με απλό κι ασφαλές τρόπο.

5.3 Εγκατάσταση βιβλιοθηκών

Για την επιτυχή λειτουργία του διακομιστή, είναι απαραίτητη η εγκατάσταση ορισμένων βιβλιοθηκών, είτε της python, είτε του django. Αυτές οι βιβλιοθήκες χρησιμοποιούνται για την επέκταση και διευκόλυνση της ήδη υπάρχουσας λειτουργικότητας των εργαλείων που διαθέτουμε. Στο τερματικό εκτελούμε τις εντολές:

```
$ pip install --upgrade django-crispy-forms
```

Πρόκειται για ένα εργαλείο που μας επιτρέπει να ελέγξουμε τη συμπεριφορά απόδοσης των django forms με κομψό τρόπο. Επίσης παρέχει λειτουργικότητα ελέγχου εγκυρότητας των δεδομένων εισόδου για τις φόρμες αυτές.

```
$ pip install schedule
```

Αφορά ένα πακέτο της python που χρησιμοποιήθηκε για τον έλεγχο της προγραμματισμένης παραγγελίας.

```
$ pip install django-apscheduler
```

Αφορά ένα πακέτο του django που επίσης χρησιμοποιήθηκε για τον έλεγχο της προγραμματισμένης παραγγελίας.

```
$ pip install Pillow
```

Εργαλείο εισαγωγής πεδίων εικόνων (ImageField) στα μοντέλα του django.

```
$ pip install numpy
```

Πακέτο για επιπρόσθετη λειτουργικότητα πινάκων, εξοπλισμένο με μεγάλο αριθμό συναρτήσεων για τη διαχείρησή τους.

```
$ pip install pytz
```

Βιβλιοθήκη που επιτρέπει ακριβείς υπολογισμούς ζωνών ώρας.

Κεφάλαιο 6

Επίλογος

6.1 Συμπεράσματα

Η παρούσα εργασία αποτέλεσε την ανάλυση μιας διαδικτυακής εφαρμογής, περιγράφοντας το σενάριο του έξυπνου ψυγείου στη φιλοσοφία του Διαδικτύου των πραγμάτων.

Έγινε λόγος για το πώς η εφαρμογή μπορεί να καταλαβαίνει ποια προϊόντα εισέρχονται και ποια εξέρχονται από το ψυγείο, καθώς και ποια τρόφιμα υπάρχουν κάθε φορά μέσα στο ψυγείο. Για την επίτευξη της εισαγωγής κι εξαγωγής προϊόντων χρησιμοποιήθηκε η τεχνολογία barcode. Μάλιστα, δίνεται η δυνατότητα στον χρήστη να παρακολουθεί τα προϊόντα που βρίσκονται στο ψυγείο χρησιμοποιώντας την εφαρμογή από όπου κι αν βρίσκεται, από οποιαδήποτε συσκευή, αρκεί να είναι συνδεδεμένη στο διαδίκτυο. Έτσι, αξιοποιώντας τον αλγόριθμο του προγράμματος για εντοπισμό κι ενημέρωση για προϊόντα που κοντεύουν ή έχουν λήξει, αυξάνεται η αποδοτικότητα και η ευκολία διαχείρισης των προϊόντων του ψυγείου.

Επίσης, έγινε λόγος για το πώς μπορεί ο χρήστης να δημιουργήσει παραγγελίες, ή πώς δημιουργεί η εφαρμογή αυτόματα παραγγελίες για τον χρήστη, ανάλογα με τις προτιμήσεις του και τα υπάρχοντα προϊόντα στο ψυγείο. Με τον τρόπο αυτόν, ελαχιστοποιείται η πιθανότητα έλλειψης απαραίτητων κι επιθυμητών τροφίμων από το ψυγείο σε περιπτώσεις που είτε ξεχάσθηκαν, είτε δεν υπήρχε η ευκολία ή η δυνατότητα να αγοραστούν.

Για την ανάπτυξη της εφαρμογής είναι αναγκαία η εξοικείωση του προγραμματιστή με το πλαίσιο django και των βασικών του εργαλείων, λειτουργιών και δομικών συστατικών (view, templates, κτλ). Απαραίτητη προϋπόθεση είναι επίσης η βασική γνώση κι εμπειρία με τις προγραμματιστικές γλώσσες python κι html, οι οποίες χρησιμοποιούνται κατά κύριο λόγο στην ανάπτυξη εφαρμογών στο django, καθώς και η κατανόηση της χρήσης της βιβλιοθήκης bootstrap, η οποία χρησιμοποιείται για το αισθητικό κομμάτι της εφαρμογής. Ακόμα, ο προγραμματιστής πρέπει να είναι σε θέση να μπορεί να δημιουργήσει ένα πλήρες σχήμα συσχετίσεων βάσης δεδομένων, να το διαχειριστεί και να εκτελεί πάνω σε αυτό όλα τα ερωτήματα που πιθανώς χρειαστούν για την επίτευξη του σκοπού της εφαρμογής.

6.2 Μελλοντικές Επεκτάσεις

Το σύστημα που αναπτύχθηκε στα πλαίσια αυτής της διπλωματικής εργασίας θα μπορούσε να βελτιωθεί και να επεκταθεί, καθώς έτσι όπως είναι σχεδιασμένη μέχρι στιγμής η εφαρμογή, αξιοποιεί μόνο τα βασικά χαρακτηριστικά των προϊόντων (όνομα, κατασκευαστής, κτλ).

6.2.1 Διατροφική αξία προϊόντων

Αρχικά προτείνεται η ενσωμάτωση συστήματος διατροφικής αξίας των προϊόντων. Αξιοποιώντας τα στοιχεία διατροφικής αξίας, τα οποία καταγράφονται σε κάθε συσκευασία τροφίμου κι είναι εφικτό να τα γνωρίζει η εφαρμογή μέσω μιας επέκτασης της βάσης δεδομένων, δίνεται η δυνατότητα στους χρήστες να βλέπουν τα στοιχεία που ενδεχομένως τους ενδιαφέρουν (θερμίδες, πρωτεΐνες, κτλ), ώστε να ελέγχουν καλύτερα τη διατροφή τους. Επίσης, αποθηκεύοντας ένα αρχείο (ιστορικό) με τη θρεπτική αξία των τροφίμων που καταναλώθηκαν το τελευταίο χρονικό διάστημα θα βοηθούσε την προσπάθεια του χρήστη για μια καλύτερη και πιο υγιεινή διατροφή.

6.2.2 Σύστημα δίαιτας

Με την προϋπόθεση ότι το πρόγραμμα αξιοποιεί το σύστημα διατροφικής αξίας των προϊόντων που αναφέρθηκε προηγουμένως, η εφαρμογή μπορεί να δημιουργεί δίαιτες για τους χρήστες, ανάλογα με τις ανάγκες και τις προτιμήσεις τους. Οι δίαιτες αυτές θα είναι λίστες με περιεχόμενο τα προϊόντα προς κατανάλωση, μαζί με την ποσότητα που απαιτείται από το καθένα και τον προγραμματισμό κατανάλωσης, ανάλογα με το γεύμα (πρωϊνό, μεσημεριανό). Οι λίστες χρειάζεται να δημιουργούνται αυτόματα μέσω αλγορίθμου της εφαρμογής. Ο αλγόριθμος θα αποφασίζει το περιεχόμενο της λίστας με βάση την ανάγκη που έχει δηλώσει ο χρήστης (π.χ να χάσει/πάρει βάρος), τα διατροφικά χαρακτηριστικά των τροφίμων κι ενδεχομένως τη λίστα επιθυμιών. Επίσης, θα ήταν επιθυμητό η εφαρμογή να δίνει τη δυνατότητα στον χρήστη να τροποποιήσει τη λίστα, χωρίς όμως να παραβιάζει τη δίαιτα, όπως για παράδειγμα να μπορεί να ανταλλάξει ένα προϊόν της λίστας με ένα άλλο εκτός λίστας, χωρίς να επηρεάζεται το σύνολο των θερμίδων. Γενικά, αυτό το σύστημα θα μπορούσε να φανεί ιδιαίτερα χρήσιμο και να αξιοποιηθεί από χρήστες που βρίσκονται σε ειδικές διατροφικές καταστάσεις (εγκυμοσύνη, σακχαρώδης διαβήτης, κτλ).

6.2.3 Σύστημα συνταγών

Η εφαρμογή θα μπορούσε να περιέχει ένα σύστημα πρότασης συνταγών για τους χρήστες της. Η αυτόματη επιλογή των συνταγών από το πρόγραμμα θα μπορούσε να γίνει μέσω ενός αλγορίθμου ο οποίος αποφασίζει ποιές θα είναι οι συνταγές αυτές με βάση τα προϊόντα τα οποία προτιμά περισσότερο ο χρήστης, τα οποία είναι αυτά που είτε βρίσκονται στη λίστα επιθυμιών, είτε καταναλώνονται με μεγαλύτερη συχνότητα. Χάριν απλότητας, θεωρούμε ότι τα προϊόντα που καθορίζουν την επιλογή μιας συνταγής αποτελούν κύριο συστατικό της συνταγής αυτής. Επίσης, πρέπει κι ο χρήστης να μπορεί να βλέπει διάφορες συνταγές που σχετίζονται με ορισμένα προϊόντα, χωρίς να απαιτείται αυτόματη πρόταση από την εφαρμογή. Για ανάκτηση των συνταγών από το πρόγραμμα ώστε να τις σερβίρει στον χρήστη, χρειάζεται είτε επέκταση της βάσης δεδομένων, ώστε

να περιλαμβάνει στατικά όλες τις συνταγές που πρόκειται να προταθούν (όχι τόσο παραγωγική προσέγγιση), είτε η δυναμική ανάκτηση συνταγών από το διαδίκτυο με χρήση κάποιου επιπρόσθετου αλγορίθμου. Επίσης, γιατί όχι να μην μπορεί ο ίδιος ο χρήστης να δημιουργεί συνταγές με διάφορα προϊόντα της επιλογής του, ώστε να έχει ταχύτερη κι άμεση πρόσβαση σε αυτές.

Βιβλιογραφία

[1] Bushra Alolayan. Do I Really Have to Accept Smart Fridges? *An empirical study. In the Proceedings of The Seventh International Conference on Advances in Computer-Human Interactions*, σελίδες 186–191, 2014.

[2] www.amazon.com. `https://www.amazon.com/`.

[3] Ralf Jung Antonio Krüger, Lübomira Spassova. Innovative Retail Laboratory – Investigating Future Shopping Technologies. Στο *it - Information Technology Methoden und innovative Anwendungen der Informatik und Informationstechnik*, σελίδες 114–118. ISSN (Print) 1611-2776, 2010, Volume 52, Issue 2. doi: https://doi.org/10.1524/itit.2010.0580.

[4] εφαρμογή android σαρωτή barcode. `https://barcodetopc.com/`.

[5] Judy Chuan Chuan Lin Chin-Lung Hsu. An empirical examination of consumer adoption of Internet of Things services: Network externalities and concern for information privacy perspectives. Στο *Computers in Human Behavior*, σελίδες 516–527. ISSN 0747-5632, 2016. doi:10.1016/j.chb.2016.04.023.

[6] Client-Server wikipedia. `https://en.wikipedia.org/wiki/Client-server_model`.

[7] django book. `https://djangobook.com/`.

[8] django documentation. `https://docs.djangoproject.com/en/`.

[9] django presentation. `https://www.slideshare.net/ssuserbf9115/django-introduction-amp-tutorial-pdf`.

[10] django windows download page. `https://docs.djangoproject.com/en/2.2/howto/windows/`.

[11] THE WONDERFUL POSSIBILITIES OF CONNECTING YOUR FRIDGE TO THE INTERNET. `https://www.wired.com/2014/10/is-your-refrigerator-running/`.

[12] Shade Kuyoro Folasade Osisanwo και Oludele Awodele. Internet Refrigerator –A typical Internet of Things (IoT). Στο *3rd International Conference on Advances in Engineering Sci-*

ences & Applied Mathematics (ICAESAM'2015) March 23-24, 2015 London (UK), σελίδες 59–63. ICAESAM, 2015.

[13] Fong A.C.M. Fong B. και Li C.K. *Telemedicine Technologies: Information Technology in Medicine and Telehealth*. Wiley, 2011.

[14] Evolution of the Fridge Computer. `http://www.orangecone.com/archives/2008/01/the_fridge_comp.html`.

[15] IoT wikipedia. `https://en.wikipedia.org/wiki/Internet_of_things`.

[16] The advantages and disadvantages of Internet Of Things. `https://e27.co/advantages-disadvantages-internet-things-20160615/`.

[17] Understanding IoT Management for Smart Refrigerator. `https://www.academia.edu/20821849/Understanding_IoT_Management_for_Smart_Refrigerator`.

[18] Internet fridges: the zombie idea that will never, ever happen. `https://www.theguardian.com/technology/2014/jan/07/internet-fridge-lg-ces-2014`.

[19] LG launches fridges, washers and cookers that chat. `https://www.bbc.com/news/technology-27308869`.

[20] python wikipedia. `https://el.wikipedia.org/wiki/Python`.

[21] python download page. `https://www.python.org/downloads/`.

[22] José Rouillard. The Pervasive Fridge. A smart computer system against uneaten food loss. *Seventh International Conference on Systems*, σελίδες 135–140, Feb 2012, Saint-Gilles, Réunion.

[23] smart device. `https://en.wikipedia.org/wiki/Smart_device`.

[24] Smart refrigerator. `https://el.wikipedia.org/wiki/Διαδίκτυο_των_πραγμάτων`.

[25] Internet Refrigerator (Smart Refrigerator). `https://en.wikipedia.org/wiki/Smart_refrigerator`.

[26] Yuan Gao Jesse S. Jin Rukshan Athauda Suhuai Luo, Hongfeng Xia. Smart Fridges with Multimedia Capability for Better Nutrition and Health. *2008 International Symposium on Ubiquitous Multimedia Computing*, σελίδες 39–44, 2008. doi:10.1109/UMC.2008.17.

[27] web application wikipedia. `https://en.wikipedia.org/wiki/Web_application`.

[28] what is a web framework. https://jeffknupp.com/blog/2014/03/03/what-is-a-web-framework/.

[29] Βάση δεδομένων wikipedia. https://el.wikipedia.org/wiki/Βάση_δεδομένων.

[30] διαδικτυακή εφαρμογή wikipedia. https://el.wikipedia.org/wiki/Διαδικτυακή_εφαρμογή.

Συντομογραφίες

βλπ	βλέπε
κτλ.	και τα λοιπά
κ.ο.κ	και ούτω καθεξής
IoT	Internet of Things
app	application
σχ.	σχήμα
OS	operating system
UI	User Interface
SQL	Structured Query Language
CES	International Consumer Electronics Show
URL	Uniform Resource Locator
API	Application Programming Interface
RFID	Radio Frequency Identification
ΣΔΒΔ	Σύστημα Διαχείρισης Βάσης Δεδομένων
DBMS	Database Management System

Ορολογία - Γλωσσάρι

Ελληνικός όρος	Αγγλικός όρος
προφίλ	Profile
barcode σαρωτής	barcode scanner
δομή δεδομένων	data base
έξυπνο ψυγείο	smart fridge
ροή εργασίας	workflow
προτόκολλο	protocol
αποσύνδεση	log out
σύνδεση	log in
χρήστης	user
παραγγελία	Order
διαδίκτυο των πραγμάτων	internet of things
παράγοντες μορφής	form factors
ενέσεις SQL	SQL injections
διαπιστευτήρια	credentials
ηλεκτρονικές επιθέσεις	cyber attacks
φυλλομετρητής/περιηγητής	browser
διακομιστής/εξυπηρετητής	server
πελάτης	client
πλαίσιο	framework
διαδίκτυο	web
μοντέλο	model
πρότυπο	template
όψη	view
ελεγκτής	controller
εγκατάσταση	install
εφαρμογή ιστού	web application
ηλεκτρονική διεύθυνση	URL
αίτηση	request
απάντηση	response
επιχειρηματική λογική	business logic
κλάση	class

μοντέλο	model
ερώτημα	query
λίστα επιθυμιών	wishlist
πρωτεύον κλειδί	primary key
όνομα χρήστη	username
κωδικός	password
αντικείμενο	object
πιστοποίηση	authentication
προκαθορισμένος	default
διαχειριστής	superuser
περιβάλλον χρήστη	user interface
ταυτοποίηση μέσω ραδιοσυχνοτήτων	Radio Frequency Identification