ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Σύστημα Παγκόσμιου Ιστού για την Παραγγελία Προϊόντων, αξιοποιώντας Τεχνολογίες NoSQL Βάσεων Δεδομένων

Διπλωματική Εργασία

Κατερίνα Μεμούσι

Επιβλέπων: Μιχαήλ Βασιλακόπουλος

Βόλος 2020

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Σύστημα Παγκόσμιου Ιστού για την Παραγγελία Προϊόντων, αξιοποιώντας Τεχνολογίες NoSQL Βάσεων Δεδομένων

Διπλωματική Εργασία

Κατερίνα Μεμούσι

Επιβλέπων: Μιχαήλ Βασιλακόπουλος

Βόλος 2020

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

A Web-based System for Ordering Products, exploiting NoSQL Database Technologies

Diploma Thesis

Katerina Memushi

Supervisor: Michail Vassilakopoulos

Volos 2020

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπων **Μιχαήλ Βασιλακόπουλος**
Αναπληρωτής Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Μέλος **Παναγιώτα Τσομπανοπούλου**
Αναπληρώτρια Καθηγήτρια, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Μέλος **Ελένη Τουσίδου**
Μέλος Ε.ΔΙ.Π, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Ημερομηνία έγκρισης: 2-10-2020

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή κ. Βασιλακόπουλο για την επίβλεψη αυτής της διπλωματικής εργασίας και για την ευκαιρία που μου έδωσε να την εκπονήσω. Επίσης, θα ήθελα να ευχαριστήσω τους γονείς μου για τη συνεχή στήριξη και συμπαράσταση που μου προσέφεραν όλα αυτά τα χρόνια.

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

«Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής».

Ο/Η Δηλών/ούσα

Κατερίνα Μεμούσι

2-10-2020

Περίληψη

Η ηλεκτρονική παραγγελία φαγητού με κατ' οίκον παράδοση βρίσκεται τα τελευταία χρόνια σε αυξητική τάση, ενώ έχει μειωθεί το ποσοστό των παραγγελιών μέσω τηλεφώνου. Υπάρχουν αρκετές πλατφόρμες ηλεκτρονικής παραγγελίας φαγητού, όμως οι περισσότερες δεν προσφέρουν τη δυνατότητα παραλαβής της παραγγελίας από το κατάστημα. Ο σκοπός της παρούσας διπλωματικής εργασίας είναι να λύσει το πρόβλημα αυτό με τη δημιουργία μιας ηλεκτρονικής εφαρμογής παραγγελίας φαγητού που περιλαμβάνει πολλά εστιατόρια και δίνει την επιλογή στο χρήστη να παραλάβει την παραγγελία του από το κατάστημα σε επιλεγμένη ημέρα και ώρα. Τα κύρια εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής είναι το Django και η NoSQL βάση δεδομένων, MongoDB. Αρχικά, έγινε σχεδιασμός της βάσης και στη συνέχεια γράφτηκε σταδιακά ο κώδικας σε γλώσσα Python, με τρόπο που να πληροί τις προδιαγραφές της εφαρμογής. Χρησιμοποιήθηκαν αρκετά πακέτα κατά την υλοποίηση της εφαρμογής, τα οποία περιλαμβάνονται στο φάκελο με τα αρχεία του κώδικα. Μέσω της εργασίας αυτής έγινε κατανοητή η διαδικασία κατασκευής μιας web εφαρμογής με το Django και ο τρόπος χρήσης και λειτουγίας μιας βάσης δεδομένων και πιο συγκεκριμένα, της MongoDB.

Λέξεις Κλειδιά

web εφαρμογή, Django, MongoDB, βάσεις δεδομένων, ηλεκτρονική παραγγελία φαγητού

Abstract

Online food ordering offering home delivery has been rising over the last years, while the percentage of phone orders has decreased. There are plenty of online food ordering platforms, but most do not offer the ability to pick up an order at store. The purpose of this thesis is to solve this problem by developing a web application for online food ordering that involves many restaurants and offers users the takeaway option at their chosen date and time. The main tools used to develop this application were Django and the NoSQL database, MongoDB. The database was the first one to be designed and then the code was gradually written in Python in a way that satisfies the requirements of the application. During the implementation of the application, several packages were used that are included in the folder containing the code files. Through this project, the process of building a web application with Django was understood as well as the way databases and more specifically MongoDB works.

Keywords

web application, Django, MongoDB, databases, online food ordering

Πίνακας περιεχομένων

Κατάλογος σχημάτων

Κεφάλαιο 1

Εισαγωγή

Ένα από τα βασικότερα προβλήματα που μοιράζονται τα άτομα της σύγχρονης κοινωνίας είναι η έλλειψη χρόνου που προκαλείται λόγω αυξημένων απαιτήσεων και υποχρεώσεων. Οι εργαζόμενοι είναι μια μερίδα πληθυσμού που γνωρίζει πολύ καλά τι σημαίνουν οι γρήγοροι ρυθμοί ζωής, καθώς παράλληλα με τη δουλειά τους πρέπει να φροντίζουν και για τη διεκπεραίωση των οικιακών τους υποχρεώσεων, στις οποίες περιλαμβάνεται και η μαγειρική. Λόγω της έλλειψης χρόνου, πολλές φορές δεν προλαβαίνουν να μαγειρέψουν και καταφεύγουν στη λύση του έτοιμου φαγητού. Την ίδια λύση βρίσκουν και οι φοιτητές, οι οποίοι είτε δεν ξέρουν να μαγειρεύουν είτε δεν έχουν χρόνο λόγω διαβάσματος. Ευτυχώς, υπάρχει πληθώρα ηλεκτρονικών εφαρμογών που δίνει τη δυνατότητα στα άτομα αυτά να παραγγείλουν ό,τι επιθυμούν, με επιλογή παράδοσης του φαγητού στο χώρο τους ή παραλαβή από το κατάστημα.

Ο ηλεκτρονικός τρόπος παραγγελίας βοηθάει τα καταστήματα στην αύξηση των πελατών τους, καθώς και στην αποδοτικότερη οργάνωση της επιχείρισής τους. Όσον αφορά στους πελάτες, μπορούν να παραγγείλουν εύκολα και γρήγορα, χωρίς να χρειάζεται να περιμένουν στην ουρά του καταστήματος. Υπάρχει όμως μια πολύ σημαντική ανάγκη για τα άτομα με περιορισμένο χρόνο: η πραγματοποίηση μιας παραγγελίας με επιλογή παραλαβής από το κατάστημα σε ώρα καθορισμένη από τον πελάτη. Αυτό το χαρακτηριστικό είναι πολύ χρήσιμο σε περιπτώσεις που κάποιος θέλει να είναι έτοιμο το φαγητό του σε μια συγκεκριμένη ώρα, για να το πάρει για παράδειγμα αφού σχολάσει από τη δουλειά του ή στο κενό από τη σχολή του.

1.1 Αντικείμενο της διπλωματικής

Σκοπός της διπλωματικής αυτής είναι να δοθεί λύση στο πρόβλημα της αναμονής στην ουρά των καταστημάτων για την παραλαβή μιας παραγγελίας. Όπως έχει ήδη αναφερθεί, τα άτομα που έχουν αναγκη αυτή τη λύση είναι κυρίως οι φοιτητές και οι εργαζόμενοι, οι οποίοι πολλές φορές χρειάζεται να πάρουν φαγητό από έξω σε κάποιο μικρό διάλειμμα από τη δουλειά ή τη σχολή τους και δε θέλουν να περιμένουν στην ουρά.

Για τη λύση του προβλήματος χρειάστηκε η ανάπτυξη μιας ηλεκτρονικής εφαρμογής, στην οποία θα συμμετέχουν καταστήματα και οι πελάτες-χρήστες θα μπορούν να κάνουν παραγγελία σε ένα από τα διαθέσιμα καταστήματα, επιλέγοντας την ημέρα και ώρα που επιθυμούν να παραλάβουν την παραγγελία τους από το κατάστημα. Πρόκειται δηλαδή για μια web εφαρμογή παραγγελίας φαγητού για takeaway σε συγκεκριμένο χρόνο.

1.1.1 Συνεισφορά

Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

1. Μελετήθηκαν παρόμοιες εφαρμογές ηλεκτρονικής παραγγελίας φαγητού
2. Σχεδιάστηκε η βάση δεδομένων
3. Αναπτύχθηκε η εφαρμογή

1.2 Οργάνωση του τόμου

Στο κεφάλαιο 2 αναφέρονται σχετικές εργασίες και εφαρμογές. Τα εργαλεία και οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής παρουσιάζονται στο Κεφάλαιο 3. Στο Κεφάλαιο 4 γίνεται σύγκριση μεταξύ των frameworks Django και Flask, ενώ αναφέρονται και τα βασικότερα κριτήρια επιλογής μιας βάσης δεδομένων. Το Κεφάλαιο 5 συζητά τη σύνδεση με τη βάση και τη διάρθρωση των αρχείων του κώδικα. Στο Κεφάλαιο 6 παρουσιάζονται εικόνες από το περιβάλλον της εφαρμογής, καθώς και ο τρόπος χρήσης της. Τα βήματα εγκατάστασης παρουσιάζονται στο Κεφάλαιο 7 και τα συμπεράσματα στο Κεφάλαιο 8.

Κεφάλαιο 2

Σχετικές Εργασίες και Εφαρμογές

Η ανάπτυξη μιας εφαμογής για ηλεκτρονική παραγγελία φαγητού σίγουρα δεν είναι μια πρωτότυπη ιδέα. Υπάρχουν ήδη εφαρμογές που χρησιμοποιούνται για αυτό το σκοπό, ενώ πολλά είναι και τα καταστήματα εστίασης που διαθέτουν τη δική τους ιστοσελίδα με δυνατότητα ηλεκτρονικής παραγγελίας. Ο πιο συνηθισμένος τρόπος λήψης μιας παραγγελίας είναι στο κατάστημα από τους σερβιτόρους. Σύμφωνα με το άρθρο [3], το βασικότερο μειονέκτημα αυτής της διαδικασίας είναι ότι το χαρτί στο οποίο σημειώνεται η παραγγελία μπορεί να χαθεί ή να καταστραφεί. Επίσης, γίνεται σπατάλη χρόνου και χαρτιού, ενώ για μια μικρή αλλαγή στο μενού του καταστήματος πρέπει να ξανά τυπωθεί ο κατάλογος από την αρχή. Στο ίδιο άρθρο αναφέρονται ακόμη, κάποιοι εναλλακτικοί τρόποι παραγγελίας, όπως το self-service και τα συστήματα PDA, τα οποία είναι ασύρματα τερματικά, όμως δεν είναι ελκυστικά για το χρήστη, καθώς δεν περιλαμβάνουν εικόνες στο μενού. Η δημιουργία μιας εφαρμογής για τη λήψη παραγγελιών ξεπερνάει τους περιορισμούς που δημιουργούνται από τους προαναφερόμενους τρόπους παραγγελίας. Τα καθήκοντα του καταστήματος μειώνονται, η διαδικασία παραγγελίας αυτοματοποιείται, το περιβάλλον διεπαφής γίνεται πιο φιλικό προς το χρήστη και ο κατάλογος μπορεί να τροποποιηθεί πολύ πιο εύκολα.

Το σύστημα που δημιουργήθηκε και αναλύεται στο παραπάνω άρθρο [3] περιλαμβάνει τα τρία βασικά μέρη ενός εστιατορίου, τα οποία είναι η κουζίνα, το ταμείο και η εξυπηρέτηση. Έχει αναπτυχθεί σε Android και χρησιμοποιείται τόσο από τους πελάτες του καταστήματος που βρίσκονται ήδη εκεί, όσο και από όσους βρίσκονται αλλού και θέλουν να κάνουν κράτηση τραπεζιού. Η διαδικασία που ακολουθούν όσοι βρίσκονται στο κατάστημα είναι να ανοίξουν το tablet, το οποίο υπάρχει πάνω σε κάθε τραπέζι και να επιλέξουν όποια από τα διαθέσιμα φαγητά επιθυμούν. Όσοι βρίσκονται εκτός του καταστήματος πρέπει να κάνουν

την παραγγελία τους από το κινητό τους. Η πληρωμή μπορεί να γίνει είτε με μετρητά είτε με κάρτα, ενώ το μενού μπορεί να τροποποιηθεί από τον ιδιοκτήτη του καταστήματος ή τον μάνατζερ. Η βάση που χρησιμοποιήθηκε σε αυτή την εφαρμογή είναι η SQL και η πλατφόρμα ανάπτυξής της είναι η Android Jelly Bean.

Μια άλλη δημοφιλής πλατφόρμα για την ανάπτυξη web εφαρμογών είναι η NodeJS, η οποία είναι χτισμένη σε περιβάλλον JavaScript. Ενώ η JavaScript είναι κυρίως γνωστή για την ανάπτυξη της πλευράς του πελάτη (client-side), η NodeJS χρησιμοποιείται στην πλευρά του διακομιστή (server-side) και συναντάται συχνά σε εφαρμογές σε συνδυασμό με τη MongoDB και το framework «Express.js». Στη διπλωματική εργασία στα πλαίσια της οποίας αναπτύχθηκε μια εφαρμογή ηλεκτρονικού εμπορίου [4], αναλύονται οι τρεις παραπάνω τεχνολογίες από τις οποίες αποτελείται η στοίβα MERN με την προσθήκη της βιβλιοθήκης «React.js». Αρχικά, το Express με την NodeJS δημιουργούν ένα Application Programming Interface (API) που χρησιμοποιείται για τη λογική και την επικοινωνία με τη MongoDB. Όταν η React στέλνει ένα αίτημα HTTP στο διακομιστή, ο διακομιστής αναλύει το αίτημα, ανακτά τα δεδομένα από τη βάση και τα στέλνει στον πελάτη. Η React στη συνέχεια ενημερώνεται με βάση αυτά τα δεδομένα.

Η παραπάνω εφαρμογή χρησιμοποιείται για την ηλεκτρονική αγορά βιβλίων από ένα βιβλιοπωλείο και έχει σαν στόχο να κάνει περισσότερο δημοφιλές το βιβλιοπωλείο, ώστε να αυξηθούν οι πωλήσεις. Περιλαμβάνει δύο ειδών χρήστες: τους διαχειριστές και τους πελάτες. Οι διαχειριστές, όπως είναι φυσικό, μπορούν να δημιουργούν, να ανανεώνουν και να διαγράφουν τα προϊόντα από τη βάση, καθώς και να επεξεργάζονται τις παραγγελίες των χρηστών. Οι χρήστες μπορούν να βλέπουν τα προϊόντα και τις πληροφορίες τους και στη συνέχεια να προσθέτουν στο καλάθι όσα βιβλία τους ενδιαφέρουν και να πληρώνουν το συνολικό ποσό.

Εκτός από τη στοίβα MERN υπάρχει και η στοίβα MEAN που αποτελείται από τη MongoDB, την Express.js, το Angular και τη NodeJS. Πάνω σε αυτή έχει αναπτυχθεί μια εφαρμογή, η οποία επίσης χρησιμοποιείται για την παραγγελία φαγητού. Στην ιστοσελίδα στην οποία περιγράφεται η εφαρμογή [5], αναφέρονται τα χαρακτηριστικά της, τα οποία περιλαμβάνουν το σύστημα ελέγχου ταυτότητας ενός χρήστη, τη λίστα των προϊόντων και την παραγγελία και πληρωμή. Αρχικά, η βάση σχεδιάστηκε σε μορφή σχεσιακών βάσεων δεδομένων. Ο χρήστης μπορεί να συνδεθεί και να δει όλα τα διαθέσιμα φαγητά του καταστήματος και στη συνέχεια να πληρώσει για την παραγγελία που δημιουργήθηκε. Στη συνέχεια, το σχήμα της βάσης μετατράπηκε σε σχήμα της MongoDB με τις εξής συλλογές: τη συλλογή

χρηστών, τη συλλογή αντικειμένων και τη συλλογή παραγγελιών. Η συλλογή των χρηστών περιλαμβάνει το όνομα, τον κωδικό, το email, το username, καθώς και το id του χρήστη. Στη συλλογή των αντικειμένων περιλαμβάνονται τα πεδία: τίτλος, τιμή, φωτογραφία, τύπος και id του προϊόντος. Η συλλογή των παραγγελιών περιέχει ένα πεδίο με τα προϊόντα της παραγγελίας που είναι ένας πίνακας με προϊόντα από τη δεύτερη συλλογή. Επίσης, περιλαμβάνονται ένα πεδίο που δείχνει ποιος χρήστης έκανε την παραγγελία, καθώς και η ποσότητα του κάθε προϊόντος που βρίσκεται στο καλάθι. Το άρθρο παραθέτει επίσης τα βήματα που ακολουθήθηκαν προκειμένου να στηθεί το περιβάλλον ανάπτυξης της εφαρμογής, ενώ δίνεται και ο κώδικας που χρησιμοποιήθηκε για τη σύνδεση με τη βάση δεδομένων.

Μία ακόμα εφαρμογή [6] που προσφέρει παρόμοιες δυνατότητες με την εφαρμογή που αναπτύχθηκε στα πλαίσια αυτής της διπλωματικής, κατασκευάστηκε με χρήση του Django και της βάσης δεδομένων «Sqlite». Για το σχεδιασμό του front-end αξιοποιήθηκαν οι τεχνολογίες HTML, CSS και JavaScript. Όπως και στις προηγούμενες εφαρμογές, στους χρήστες της περιλαμβάνονται τόσο οι διαχειριστές που μπορεί να είναι οι ιδιοκτήκτες των καταστημάτων ή οι managers, όσο και οι πελάτες. Οι πελάτες μπορούν να δουν διαφορετικά καταστήματα εστίασης και να διαλέξουν αυτό από το οποίο επιθυμούν να παραγγείλουν. Για κάθε κατάστημα εμφανίζεται η κατηγορία του, το ελάχιστο ποσό παραγγελίας και αν είναι ανοιχτό ή κλειστό. Αφού ο χρήστης επιλέξει κατάστημα, μπορεί να δει το μενού του και για κάθε προϊόν που εμφανίζεται, δίπλα αναγράφεται η κατηγορία του, η τιμή του και δύο κουμπιά για αύξηση και μείωση ποσότητας κατά την προσθήκη στο καλάθι. Ο διαχειριστής μπορεί να προσθέσει ένα καινούριο προϊόν, καθώς και να επεξεργαστεί ή να διαγράψει ένα υπάρχον. Επίσης, μέσω της επιλογής «Orders» έχει τη δυνατότητα να δει το ιστορικό παραγγελιών για το κατάστημά του, το οποίο περιλαμβάνει τη διεύθυνση παράδοσης, τη λίστα των προϊόντων, την ποσότητα και το συνολικό ποσό προς πληρωμή. Τέλος, μπορεί να επιλέξει μια κατάσταση για την παραγγελία, η οποία δείχνει την εξέλιξή της.

Όπως προκύπτει από τις εργασίες που αναλύθηκαν παραπάνω, η συγκεκριμένη εφαρμογή έχει κοινά χαρακτηριστικά και λειτουργίες με αυτές, όμως παρουσιάζει και αρκετές διαφορές. Καταρχάς, όλες οι εργασίες με εξαίρεση την τελευταία χρησιμοποιούνται για την παραγγελία από ένα μόνο κατάστημα και δε δίνουν τη δυνατότητα επιλογής καταστήματος. Η παρούσα εφαρμογή πρόκειται για μία πλατφόρμα καταστημάτων εστίασης που περιλαμβάνει διάφορα καταστήματα από τα οποία μπορεί να επιλέξει ο χρήστης. Η τελευταία εφαρμογή, ενώ αφορά και εκείνη μια πλατφόρμα καταστημάτων, προσφέρει περιορισμένες λειτουργίες. Δε δίνεται

η δυνατότητα επιλογής υλικών κατά την προσθήκη ενός προϊόντος στο καλάθι, ενώ ο χρήστης δεν μπορεί να δει το ιστορικό των παραγγελιών του. Επίσης, το περιβάλλον της εφαρμογής είναι πιο απλό και τα χαρακτηριστικά του κάθε προϊόντος που μπορεί να δει ο χρήστης είναι λιγότερα. Μια βασική διαφορά της παρούσας εφαρμογής με τις παραπάνω είναι και ο τρόπος παραλαβής των προϊόντων. Στο άρθρο που αναλύθηκε, η εφαρμογή χρησιμοποιείται μόνο για παραγγελίες που πρόκειται να καταναλωθούν εντός του καταστήματος. Όμως, σε αυτή την εφαρμογή ο χρήστης επιλέγει την ώρα και ημέρα παραλαβής της παραγγελίας, με σκοπό να την καταναλώσει εκτός του καταστήματος. Τέλος, οι σχετικές εφαρμογές έχουν αναπτυχθεί με διαφορετικά frameworks, γεγονός που σημαίνει ότι η γλώσσα προγραμματισμού είναι άλλη. Όσες εφαρμογές έχουν αναπτυχθεί με το Django χρησιμοποιούν σχεσιακές βάσεις δεδομένων σε αντίθεση με την εφαρμογή αυτή που χρησιμοποιεί τη NoSQL βάση, MongoDB.

Κεφάλαιο 3

Τεχνολογίες και εργαλεία αναπτυξης

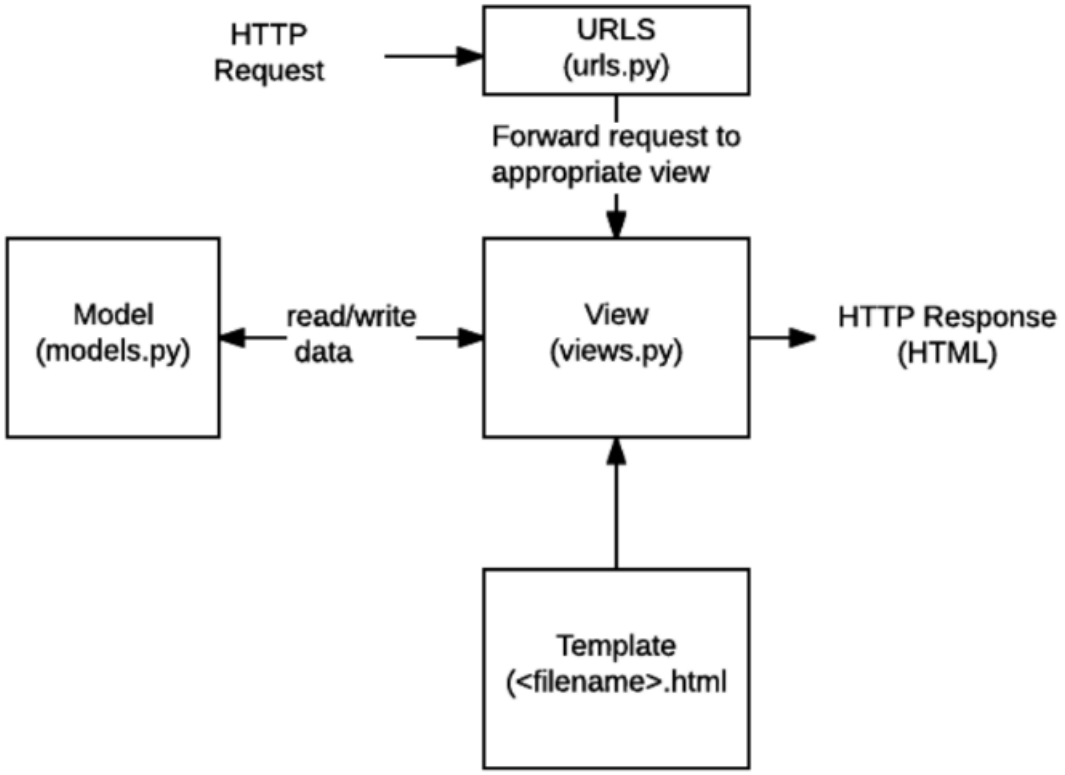
3.1 Εισαγωγή

Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε ένας συνδυασμός εργαλείων και τεχνολογιών, με σκοπό να καλυφθούν οι απαιτήσεις τις εφαρμογής στο front-end, το back-end και τη βάση δεδομένων. Τα εργαλεία και οι εκδόσεις τους επιλέχθηκαν έτσι ώστε να συνδυάζονται μεταξύ τους για να μην παρουσιαστούν προβλήματα κατά την ανάπτυξη της εφαρμογής. Παρακάτω περιγράφονται αναλυτικά οι τεχνολογίες που χρησιμοποιήθηκαν.

3.2 Django

Το web framework που χρησιμοποιήθηκε είναι το Django, ένα από τα πιο δημοφιλή γραμμένα σε Python [7]. Το Django διευκολύνει τη δημιουργία των web εφαρμογών, καθώς περιλαμβάνει πολλές βιβλιοθήκες και δίνει έμφαση στην αποδοτικότητα και την επαναχρησιμοποίηση των στοιχείων. Μερικές από τις δυνατότητες που προσφέρει είναι ο έλεγχος της ταυτότητας ενός χρήστη, οι φόρμες και η προσθήκη και επεξεργασία δεδομένων μέσω του πάνελ διαχείρισης στο οποίο έχει πρόσβαση μόνο ο διαχειριστής.

Όταν ένας χρήστης ζητάει τη διεύθυνση μια σελίδας, γίνεται ένα αίτημα προς τον server, το οποίο διαχειρίζεται το Django. Όπως φαίνεται στο σχήμα 3.1, το Django ψάχνει τη ζητούμενη διεύθυνση στο αρχείο «urls.py» στο οποίο κάθε διεύθυνση της web εφαρμογής αντιστοιχίζεται σε μια συνάρτηση που ονομάζεται view. Εάν η διεύθυνση βρεθεί, γίνεται μεταβίβαση του αιτήματος στο αντίστοιχο view, το οποίο θα πρέπει να επιστρέψει μια απάντηση στο αίτημα. Η απάντηση μπορεί να είναι μια HTML σελίδα, μια ανακατεύθυνση σε διαφορε-



Σχήμα 3.1: Διάγραμμα ροής αιτημάτων και απαντήσεων στο Django (Πηγή: [1])

τική σελίδα, ένα error ή οτιδήποτε μπορεί να εμφανιστεί σε μια ιστοσελίδα. Ένα view μπορεί να διαβάσει και να γράψει δεδομένα από και προς τη βάση, αντίστοιχα. Για παράδειγμα, μπορεί να διαβάσει από τη βάση το τηλέφωνο του χρήστη που έκανε το αίτημα και να επιτρέψει στον χρήστη να το επεξεργαστεί, αλλάζοντας την τιμή που υπάρχει στη βάση δεδομένων.

Κάθε πίνακας της βάσης δεδομένων ορίζεται από ένα μοντέλο του Django. Κάθε μοντέλο είναι μια κλάση της Python και κάθε χαρακτηριστικό του είναι ένα πεδίο του πίνακα. Με λίγα λόγια, ένα μοντέλο είναι η πηγή της πληροφορίας για τα δεδομένα της εφαρμογής. Περιλαμβάνει όλα τα απαραίτητα πεδία και τις συμπεριφορές των δεδομένων που αποθηκεύονται. Μετά από μια αλλαγή στην βάση δεδομένων, ο χρήστης μπορεί να ενημερωθεί για την επιτυχία ή την αποτυχία της αλλαγής, μέσω του εργαλείου «messages».

3.3 MongoDB

3.3.1 Τι είναι η MongoDB

Η MongoDB ανήκει στις NoSQL βάσεις δεδομένων και είναι προσανατολισμένη σε έγγραφα. Αυτό σημαίνει ότι για την αποθήκευση πληροφοριών χρησιμοποιούνται έγγραφα που μοιάζουν με τη μορφή JSON.

Το JSON (JavaScript Object Notation) είναι το πιο δημοφιλές πρότυπο ανταλλαγής δεδομένων στο Διαδίκτυο και αποτελείται από ζευγάρια ονομάτων και τιμών, οι οποίες μπορεί να είναι αριθμοί, strings, συναρτήσεις, κ.λπ. Ένα έγγραφο JSON μπορεί να διαβαστεί το ίδιο εύκολα από έναν άνθρωπο και μια μηχανή, όμως το γεγονός ότι υποστηρίζει έναν περιορισμένο αριθμό τύπων δεδομένων και η ανάλυσή του είναι χρονοβόρα, το καθιστά λιγότερο κατάλληλο για χρήση μέσα σε μια βάση. Αντίθετα, τα έγγραφα της μορφής BSON (Binary JSON), η οποία μπορεί να θεωρηθεί ως δυαδική ή αριθμητική αναπαράσταση των εγγράφων JSON, μπορούν να αναλυθούν πολύ πιο γρήγορα. Αυτό οφείλεται στη δυαδική δομή της BSON που κωδικοποιεί τον τύπο και το μήκος της πληροφορίας. Η μορφή BSON υποστηρίζει τύπους δεδομένων που δεν υποστηρίζονται από τη JSON, όπως η ημερομηνία και τα δυαδικά δεδομένα. Τα έγγραφα που χρησιμοποιεί η MongoDB είναι σε μορφή BSON και μπορούν να αποθηκεύσουν οτιδήποτε μπορεί να αναπαρασταθεί και από τη μορφή JSON.

3.3.2 Διαφορά MongoDB και RDBMS

Η MongoDB αποτελείται από συλλογές και έγγραφα. Οι συλλογές είναι ανάλογες με τους πίνακες των σχεσιακών βάσεων δεδομένων, ενώ κάθε έγγραφο αντιστοιχεί σε μια εγγραφή και αποτελείται από ζεύγη κλειδιού-τιμής. Τα έγγραφα ομαδοποιούνται σε συλλογές και έχουν δυναμικό σχήμα. Πιο συγκεκριμένα, τα έγγραφα μιας συλλογής δε χρειάζεται να έχουν ίδια δομή ή αριθμό πεδίων, και τα κοινά πεδία των εγγράφων μιας συλλογής μπορούν να αποθηκεύουν διαφορετικούς τύπους δεδομένων. Επίσης, για την αναζήτηση δεδομένων με σχέση πινάκων «Ένα προς πολλά» δε χρειάζεται να γίνει συνένωση μεταξύ των πινάκων όπως στις σχεσιακές βάσεις δεδομένων, καθώς τα έγγραφα μπορούν να ενσωματώνουν άλλα έγγραφα. Για παράδειγμα, σε ένα σύστημα σχεσιακών βάσεων δεδομένων αν ένα άτομο έχει πολλές διευθύνσεις καταχωρημένες στη βάση, τότε για την εύρεση όλων των διευθύσεων θα πρέπει να γίνει join μεταξύ του πίνακα των διευθύνσεων και του πίνακα των ατόμων. Όμως, στη MongoDB το μόνο που χρειάζεται είναι να βρεθεί το έγγραφο που αντιστοιχεί στο συγ-

κεκριμένο άτομο, στο οποίο θα περιλαμβάνονται όλες οι διευθύνσεις. Στον πίνακα 3.1 φαίνονται οι σημαντικότερες διαφορές μεταξύ της MongoDB και ενός συστήματος σχεσιακών βάσεων δεδομένων.

Σύστημα Σχεσιακών Βάσεων Δεδομένων (RDBMS)	MongoDB
Βάση	Βάση
Πίνακας	Συλλογή
Εγγραφή	Έγγραφο
Στήλη	Πεδίο
Συνένωση πινάκων (Join)	Ενσωματωμένα έγγραφα

Πίνακας 3.1: Διαφορές μεταξύ συστημάτων σχεσιακών βάσεων δεδομένων και MongoDB

3.3.3 Χαρακτηριστικά της MongoDB

Ένα χαρακτηριστικό της MongoDB είναι τα ευρετήρια (indexes). Τα ευρετήρια είναι ειδικές δομές δεδομένων που αποθηκεύουν ένα μικρό ποσό των δεδομένων μιας συλλογής. Ένα ευρετήριο αποθηκεύει την τιμή ενός ή παραπάνω πεδίων, ταξινομημένων με βάση την τιμή του πεδίου. Τα ευρετήρια αυξάνουν σημαντικά την ταχύτητα των ερωτημάτων (queries) και μπορούν να δημιουργηθούν και για πεδία που είναι ενσωματωμένα. Εάν δεν έχει οριστεί κάποιο ευρετήριο, η MongoDB θα πρέπει να σκανάρει όλα τα έγγραφα μιας συλλογής για να βρει όσα ικανοποιούν τις συνθήκες ενός ερωτήματος.

Στη MongoDB είναι διαθέσιμη η αναπαραγωγή δεδομένων (replication), δηλαδή ο συγχρονισμός δεδομένων που βρίσκονται σε διαφορετικούς διακομιστές. Η αναπαραγωγή αυξάνει τη διαθεσιμότητα των δεδομένων, ενώ δε χρειάζεται η διακοπή λειτουργίας όλων των διακομιστών για τη δημιουργία αντιγράφων ασφαλείας. Τα σύνολα αντιγράφων είναι ομάδες εγγραφών που περιλαμβάνουν το ίδιο σύνολο δεδομένων. Η διατήρηση πολλαπλών αντιγράφων των δεδομένων σε διαφορετικούς διακομιστές είναι χρήσιμη σε περίπτωση που κάποιος διακομιστής χάσει τα δεδομένα του. Ένα σύνολο αντιγράφων περιλαμβάνει διάφορους κόμβους που φέρουν δεδομένα και προαιρετικά έναν κόμβο ρυθμιστή, ο οποίος δεν μπορεί να αποθηκεύσει δεδομένα. Από τους κόμβους που φέρουν δεδομένα, μόνο ένας θεωρείται κύριος κόμβος, ενώ οι υπόλοιποι θεωρούνται δευτερεύοντες. Ο κύριος κόμβος λαμβάνει όλες τις λειτουργίες εγγραφής και ανάγνωσης και καταγράφει όλες τις αλλαγές στο σύνολο δεδο-

μένων του. Οι δευτερεύοντες κόμβοι εφαρμόζουν όλες τις λειτουργίες στα δικά τους σύνολα δεδομένων, έτσι ώστε τα σύνολα δεδομένων τους να αντανακλούν το σύνολο δεδομένων του κύριου κόμβου. Αν ο κύριος κόμβος δεν είναι διαθέσιμος, ένας κατάλληλος δευτερεύον θα επιλεχθεί ως κύριος κόμβος. Ένας κόμβος ρυθμιστής δεν μπορεί να επιλεχθεί ως κύριος κόμβος, όμως μπορεί να ψηφίσει ένα δευτερεύοντα.

Οι λειτουργίες συναθροίσεων (aggregations) επεξεργάζονται εγγραφές δεδομένων και επιστρέφουν υπολογιζόμενα αποτελέσματα. Ομαδοποιούν τιμές από πολλαπλά έγγραφα και μπορούν να εκτελέσουν ποικίλες λειτουργίες στα ομαδοποιημένα δεδομένα προκειμένου να επιστρέψουν ένα αποτέλεσμα. Τα αντίστοιχα των συναθροίσεων στην SQL είναι τα «group by» και «count(*)» Η MongoDB παρέχει τρεις τρόπους για την εκτέλεση της συνάθροισης: τον αγωγό συνάθροισης, τη συνάρτηση map-reduce και τις μεθόδους συνάθροισης μοναδικού σκοπού. Στον πρώτο τρόπο, τα έγγραφα εισέρχονται σε έναν αγωγό πολλαπλών σταδίων που τα μετατρέπει σε ένα συγκεντρωμένο αποτέλεσμα. Η τεχνική map-reduce περιλαμβάνει δύο φάσεις: το στάδιο map που επεξεργάζεται κάθε έγγραφο και εξάγει ένα ή περισσότερα αντικείμενα για κάθε έγγραφο και τη φάση reduce η οποία συνδυάζει τα αποτελέσματα της φάσης map. Προαιρετικά μπορεί να υπάρξει και ένα τελικό στάδιο στο οποίο εφαρμόζονται αλλαγές στο αποτέλεσμα. Η συγεκριμένη τεχνική χρησιμοποιεί συναρτήσεις σε JavaScript, γεγονός που προσφέρει ευελιξία σε σχέση με τον αγωγό συνάθροισης, όμως είναι λιγότερο αποδοτική και πιο πολύπλοκη. Τέλος, οι μέθοδοι συνάθροισης μοναδικού σκοπού παρέχουν εύκολη πρόσβαση σε γνωστές μεθόδους συνάθροισης, όμως είναι λιγότερο ευέλικτες από τις άλλες δύο τεχνικές.

Ο διαμοιρασμός (sharding) είναι μια μέθοδος που κατανέμει δεδομένα σε πολλαπλές μηχανές και χρησιμοποιείται από τη MongoDB για την υποστήριξη εφαρμογών με πολύ μεγάλα σύνολα δεδομένων. Υπάρχουν δύο μέθοδοι για την ανάπτυξη του συστήματος: η κάθετη και η οριζόντια κλιμάκωση. Η κάθετη κλιμάκωση αυξάνει τη χωρητικότητα ενός διακομιστή χρησιμοποιώντας μια πιο ισχυρή CPU, προσθέτοντας RAM ή αυξάνοντας τον αποθηκευτικό χώρο. Αυτού του είδους η κλιμάκωση χρησιμοποιείται από τις σχεσιακές βάσεις δεδομένων. Φυσικά υπάρχουν περιορισμοί στη διαθέσιμη τεχνολογία που εμποδίζουν το μηχάνημα να είναι επαρκώς αποδοτικό. Η οριζόντια κλιμάκωση χρησιμοποιείται από τη MongoDB και περιλαμβάνει το διαχωρισμό των δεδομένων του συστήματος σε πολλαπλούς διακομιστές, προσθέτοντας επιπλέον διακομιστές ανάλογα με την απαιτούμενη χωρητικότητα. Κάθε μηχάνημα αναλαμβάνει ένα τμήμα του συνολικού φόρτου και παρέχει πιθανώς μεγαλύτερη

αποδοτικότητα από ένα διακομιστή υψηλής ταχύτητας και χωρητικότητας. Η προσθήκη διακομιστών για την αύξηση της χωρητικότητας μπορεί να έχει χημηλότερο κόστος από ένα μηχάνημα υψηλού επιπέδου harware, όμως αυξάνει την πολυπλοκότητα της οργάνωσης και της συντήρησης μιας εφαρμογής.

3.4 HTML, CSS, JavaScript και jQuery

Για το front-end της εφαρμογής χρησιμοποιήθηκαν οι HTML, CSS και JavaScript [8]. Η HTML είναι η κύρια γλώσσα σήμανσης για τις ιστοσελίδες, η οποία κατασκευάζει τη δομή του περιεχομένου τους. Η CSS χρησιμοποιείται για την μορφοποίηση των σελίδων, κάτι το οποίο δεν προσφέρει η HTML. Η JavaScript, μία από τις πιο δημοφιλείς γλώσσες προγραμματισμού, υποστηρίζει αντικειμενοστραφές και λειτουργικό στυλ προγραμματισμού. Η jQuery [9] είναι μια βιβλιοθήκη JavaScript που μπορεί να διαμορφώσει HTML και CSS στοιχεία, εφέ JavaScript και animations με χρήση μικρού κώδικα. Στη συγκεκριμένη εφαρμογή χρησιμοποιήθηκε για την εμφάνιση δεδομένων σε πραγματικό χρόνο, χωρίς την ανάγκη ανανέωσης της σελίδας. Καθώς η εφαρμογή περιλαμβάνει πολλές σελίδες τόσο για την πλευρά του πελάτη-χρήστη όσο και για τα καταστήματα που μπορούν να εγγραφούν, χρησιμοποιήθηκε ένα έτοιμο template [10], κατάλληλο για ηλεκτρονική παραγγελία φαγητού. Το template αυτό προσαρμόστηκε στις απαιτήσεις της εφαρμογής και έγιναν αλλαγές σε κάθε HTML σελίδα που περιλάμβανε.

Κεφάλαιο 4

Σύγκριση συστημάτων

4.1 Εισαγωγή

Στο κεφάλαιο αυτό γίνεται σύγκριση κάποιων frameworks και βάσεων δεδομένων. Πιο συγκεκριμένα, αναλύονται οι διαφορές των frameworks Django και Flask, καθώς και της MongoDB με τις σχεσιακές βάσεις δεδομένων.

4.2 Σύγκριση Django και Flask

Το Django και το Flask είναι τα δύο πιο δημοφιλή frameworks που χρησιμοποιούνται για την ανάπτυξη εφαρμογών. Προσφέρουν περίπου τις ίδιες δυνατότητες, όμως παρουσιάζουν αρκετές διαφορές. Το γεγονός ότι και τα δύο frameworks είναι γραμμένα σε Python καθιστά και τα δύο συστήματα αρκετά πιο εύκολα για το χρήστη σε σχέση με τα υπόλοιπα, καθώς η Python θεωρείται από τις πιο απλές γλώσσες προγραμματισμού και με τη χρήση της μπορούν να γραφούν περισσότερες συναρτήσεις με λιγότερες γραμμές κώδικα.

Το βασικά πελονεκτήματα του Django είναι ότι διαθέτει ORM(Object Relational Mapping). Η τεχνική ORM προσφέρει τη δυνατότητα στο χρήστη να αλληλεπιδρά με τη βάση δεδομένων χρησιμοποιώντας τη γλώσσα προγραμματισμού με την οποία αναπτύσσει τον κώδικα, και όχι την SQL, για παράδειγμα. Αυτό γίνεται με τη χρήση των μοντέλων δεδομένων, τα οποία επιτρέπουν στον προγραμματιστή να συνδέσει τους πίνακες της βάσης με κλάσεις, ώστε να μπορεί να διαχειρίζεται τα μοντέλα με τον ίδιο τρόπο όπως τις σχέσεις μεταξύ των πινάκων στη βάση. Με τη χρήση μιας βιβλιοθήκης ORM, ο προγραμματιστής μπορεί να εναλλάσει βάσεις δεδομένων χωρίς να χρειάζεται να γράφει κάθε φορά στη γλώσσα της

αντίστοιχης βάσης. Με τη χρήση της ίδιας γλώσσας για την πρόσβαση σε διαφορετικές βάσεις, η ανάπτυξη του κώδικα ολοκληρώνεται πολύ γρηγορότερα και αποφεύγεται η χρήση πολύπλοκων ερωτημάτων. Με αυτό τον τρόπο, ο κώδικας είναι πιο «καθαρός» και μπορεί να διαβαστεί και να τροποποιηθεί αρκετά εύκολα. Παρόλα αυτά, κάποιες φορές η σωστή ρύθμιση μιας βιβλιοθήκης ORM μπορεί να είναι περίπλοκη διαδικασία.

Με την εγκατάσταση του Django ο χρήστης μπορεί να χρησιμοποιήσει έτοιμα πακέτα στην εφαρμογή του, τα οποία είναι φτιαγμένα για το Django. Όλα τα πακέτα προσφέρονται δωρεάν και ο προγραμματιστής μπορεί να διαλέξει αυτό που ταιριάζει καλύτερα στις απαιτήσεις της εφαρμογής χωρίς να χρειαστεί να γράψει ξανά κώδικα για εφαρμογές που ήδη υπάρχουν. Όσον αφορά την ασφάλεια, το Django είναι αρκετά ασφαλές, καθώς προσφέρει προστασία έναντι σε επιθέσεις XSS, CSRF και ενθέσεις SQL. Οι επιθέσεις XSS επιτρέπουν στον εισβολέα να εισάγει «μολυσμένο» κώδικα στον browser του χρήστη, όμως τα templates του Django προστατεύουν τις εφαρμογές από τις περισσότερες επιθέσεις αυτού του τύπου. Μέσω των CSRF επιθέσεων, ο εισβολέας μπορεί να εκτελέσει ενέργειες χρησιμοποιώντας τα στοιχεία ενός άλλου χρήστη, ενώ με τις ενθέσεις SQL μπορεί να εκτελέσει κώδικα SQL στη βάση δεδομένων.

Το Django προσφέρει το πακέτο του Django Admin, μέσω του οποίου ο διαχειριστής μιας εφαρμογής μπορεί να τροποποιήσει δεδομένα που αφορούν τους χρήστες και γενικότερα την εφαρμογή. Μπορεί να δει όλα τα μοντέλα της εφαρμογής, να προσθέσει νέες εγγραφές και να τροποποιήσει ή να διαγράψει τις ήδη υπάρχουσες. Για παράδειγμα, μπορεί να προσθέσει ένα νέο χρήστη ή να επεξεργαστεί τα στοιχεία μιας παραγγελίας που έχει πραγματοποιηθεί. Στο Django Admin ο προγραμματιστής μπορεί να προσθέσει ή να αφαιρέσει διάφορες δυνατότητες και για τη χρήση του χρειάζονται ελάχιστες τεχνικές δεξιότητες. Τελευταίο, αλλά εξίσου σημαντικό πλεονέκτημα για τη χρήση του Django είναι το γεγονός ότι προσφέρει πολύ καλό documentation και η κοινότητά του είναι ενεργή και προσπαθεί να το κάνει όσο το δυνατό πιο φιλικό προς τους αρχάριους προγραμματιστές.

Όμως, παρόλο που προσφέρει αρκετές δυνατότητες, θεωρείται από πολλούς παλαιομοδίτικο μιας και ο προγραμματιστής χρειάζεται να φτιάξει κάποια μέρη της ιστοσελίδας που ίσως να μη χρειαζόταν αν χρησιμοποιούσε ένα πιο μοντέρνο framework. Ακόμα, θεωρείται υπερβολικό για χρήση σε μικρά projects και λίγο χαοτικό για τα μεγάλα, καθώς τα μοντέλα περιλαμβάνονται όλα σε ένα αρχείο. Τέλος, η χρήση των κανονικών εκφράσεων στον ορισμό των URLs δεν είναι εύκολη, ειδικά για τους αρχάριους.

Το Flask θεωρείται πιο μοντέρνο framework και προσφέρει μεγάλη ευελιξία, καθώς μπορεί να τροποποιηθεί εύκολα και με ασφάλεια λόγω της απλότητάς του. Έχει καλύτερες επιδόσεις, γιατί υπάρχουν λιγότερα επίπεδα μεταξύ του χρήστη και της βάσης δεδομένων και έτσι είναι πιο γρήγορο. Επίσης, δίνει τη δυνατότητα να δημιουργηθούν πολλές εφαρμογές Flask, κατανεμημένες σε ένα μεγάλο δίκτυο από διακομιστές, όπου η καθεμιά έχει συγκεκριμένους σκοπούς. Αυτό προσφέρει μεγαλύτερη αποδοτικότητα και καλύτερη επίδοση.

Μία από τις διαφορές του με το Django, που συγκαταλλέγεται στα αρνητικά του στοιχεία, είναι ότι ενώ είναι απλούστερο δεν προσφέρει τις ίδιες γνώσεις που προσφέρει η ανάπτυξη μιας εφαρμογής με το Django. Ένας προγραμματιστής με γνώση της Python μπορεί να προσαρμοστεί γρηγορότερα στο Flask σε αντίθεση με το Django. Όμως, η κοινότητα του Django αποτελείται από πιο έμπειρους προγραμματιστές που γνωρίζουν καλά το framework. Έτσι, ένας προγραμματιστής με εμπειρία στη χρήση του Django μπορεί να προσαρμοστεί πιο γρήγορα σε ένα νέο project του Django από ότι θα προσαρμοστεί ένας προγραμματιστής με εμπειρία στο Flask σε ένα καινούριο project του Flask. Ακόμα, το Flask προσφέρει λιγότερα εργαλεία σε σχέση με το Django, επομένως κατά την ανάπτυξη μιας εφαρμογής μπορεί να χρειαστεί η ανάπτυξη επιπλέον κώδικα από τον προγραμματιστή. Για παράδειγμα, στο Flask δεν υπάρχει πακέτο που να χρησιμοποιείται για την εγγραφή και τη σύνδεση των χρηστών. Ο προγραμματιστής θα πρέπει να γράψει από την αρχή τον κώδικα ή να τον βρει από άλλη πηγή. Επίσης, δεν υπάρχει σελίδα για τον διαχειριστή της εφαρμογής, όμως το πρόβλημα αυτό μπορεί να επιδιορθωθεί με την εγκατάσταση της βιβλιοθήκης «flask-admin». Μία ακόμα βασική διαφορά μεταξύ των δύο frameworks είναι πως το Flask δε διαθέτει βιβλιοθήκη ORM και έτσι όλες οι εντολές που αφορούν τη βάση δεδομένων πρέπει να γράφονται στην αντίστοιχη γλώσσα. Όσον αφορά την ασφάλεια, η εγκατάσταση της βιβλιοθήκης «Flask-Security» προσφέρει σχεδόν τους ίδιους μηχανισμούς με το Django για την αποτροπή διαρροών δεδομένων και άλλων επιθέσεων. Τέλος, όπως ήδη αναφέρθηκε, η κοινότητα του Flask αποτελείται από λιγότερο έμπειρους προγραμματιστές. Έτσι, η αναζήτηση βοήθειας στην κοινότητα μπορεί να είναι δύσκολη και η εύρεση λύσης σε κάποιο πιθανό πρόβλημα να είναι περισσότερο χρονοβόρα.

Μέσω της σύγκρισης των δύο frameworks γίνεται κατανοητό ότι για την ανάπτυξη αυτής της εφαρμογής, το Django είναι πιο κατάλληλο, καθώς δίνει μεγαλύτερη έμφαση στη λειτουργικότητα. Παρέχει περισσότερες δυνατότητες, μεγαλύτερη κοινότητα, ενώ παράλληλα προσφέρει ασφάλεια χωρίς την εγκατάσταση επιπρόσθετων πακέτων.

4.3 Κριτήρια επιλογής βάσης δεδομένων

Για την ανάπτυξη μιας εφαρμογής, απαραίτητο βήμα είναι η επιλογή της κατάλληλης βάσης. Κατά την επιλογή πρέπει να είναι ξεκάθαρες οι απαιτήσεις της εφαρμογής και να ληφθούν υπόψιν διαφορετικές παράμετροι. Υπάρχουν πολλές βάσεις δεδομένων που μπορεί να χρησιμοποιηθούν σε μια εφαρμογή, όμως παρακάτω θα αναλυθούν οι περιπτώσεις στις οποίες είναι κατάλληλη η χρήση της MongoDB ή της MySQL [11].

Αρχικά, βασικό κριτήριο επιλογής μιας βάσης δεδομένων είναι η δομή του σχήματος της βάσης. Ανάλογα με τις λειτουργίες της εφαρμογής, το σχήμα της βάσης μπορεί να είναι είτε αυστηρά δομημένο είτε αδόμητο. Για παράδειγμα, στην παρούσα εφαρμογή μέσω των φορμών που υπάρχουν και των συναλλαγών που πραγματοποιούνται, τα πεδία των πινάκων που συμπληρώνονται είναι κοινά. Όταν ένα κατάστημα καταχωρεί τα προϊόντα του στην ιστοσελίδα, καλείται να συμπληρώσει τα ίδια στοιχεία με οποιοδήποτε άλλο κατάστημα κάνει την ίδια ενέργεια. Έτσι και όταν ένας χρήστης επιλέξει να προσθέσει στο καλάθι του προϊόντα, θα αποθηκευτούν στη βάση οι τιμές όλων των πεδίων του αντίστοιχου πίνακα. Επομένως, η χρήση μιας βάσης με δομημένο σχήμα είναι η κατάλληλη επιλογή. Οι βάσεις που διαθέτουν αυτό το χαρακτηριστικό είναι οι σχεσιακές, μεταξύ των οποίων και η MySQL. Έχουν αυστηρά δομημένο σχήμα και τα πεδία των πινάκων τους είναι τα ίδια για όλες τις εγγραφές [12]. Η MongoDB, αντίθετα, προσφέρει μεγάλη ευελιξία στο σχήμα της βάσης και τα έγγραφα μιας συλλογής μπορούν να περιλαμβάνουν τελείως διαφορετικά δεδομένα. Σε περίπτωση που μελλοντικά υπάρχει πιθανότητα να προστεθούν και άλλες λειτουργίες σε μια εφαρμογή, ίσως θα ήταν καλύτερη επιλογή η MongoDB, καθώς η MySQL είναι δύσκολο να αλλάξει το σχήμα της, αφού θα χρειαστεί να αλλάξουν και οι σχέσεις μεταξύ των πινάκων της βάσης.

Ένας άλλος παράγοντας που καθορίζει την επιλογή της βάσης είναι η απόδοση και η ταχύτητα που απαιτείται από την εφαρμογή. Η MongoDB μπορεί να διαχειριστεί τεράστιο όγκο μη-δομημένων δεδομένων σε μικρό χρόνο. Αντίθετα, η MySQL είναι πιο αργή όταν επεξεργάζεται πολλά δεδομένα, επομένως είναι καλύτερη για εφαρμογές με μικρό όγκο δεδομένων. Στη συγκεκριμένη εφαρμογή καθώς θα υπάρχουν πολλές συναλλαγές μελλοντικά, η MongoDB είναι καλύτερη επιλογή όσον αφορά τον όγκο των δεδομένων. Μια ακόμα παράμετρος που σχετίζεται με το πλήθος των δεδομένων είναι η «θραύση» ή αλλιώς «sharding». Όπως έχει αναφερθεί στο Κεφάλαιο 3 υπάρχουν δύο ειδών sharding: το οριζόντιο και το κάθετο. Το κάθετο μπορεί να γίνει τόσο από τις σχεσιακές βάσεις δεδομένων όσο και από τις

μη-σχεσιακές, ενώ το οριζόντιο sharding μπορεί να γίνει μόνο από τις NoSQL βάσεις και περιλαμβάνει την προσθήκη επιπλέον διακομιστών για τον διαμοιρασμό των δεδομένων. Έχει πολύ χαμηλότερο κόστος από το οριζόντιο, αφού δεν περιλαμβάνει κόστος για το hardware. Η ικανότητα των NoSQL βάσεων να διαμοιράζονται τα δεδομένα οριζόντια οφείλεται στο αδόμητο σχήμα τους, καθώς τα δεδομένα λειτουργούν σχεδόν ανεξάρτητα.

Η MySQL και γενικότερα οι σχεσιακές βάσεις δεδομένων είναι περισσότερο κατάλληλες από τη MongoDB όταν χρειάζονται οι ιδιότητες ACID [13]. Το ACID είναι ένα σύνολο ιδιοτήτων το οποίο εγγυάται ότι οι συναλλαγές στη βάση δεδομένων λειτουργούν αξιόπιστα. Περιλαμβάνει την ατομικότητα, τη συνέπεια, την απομόνωση και τη μονιμότητα. Η ατομικότητα φροντίζει για την απόρριψη μιας ενέργειας όταν ένα μέρος μιας συναλλαγής ή και ολόκληρη η συναλλαγή αποτύχει. Σε αυτή την περίπτωση, η βάση παραμένει όπως πριν τη συναλλαγή. Η συνέπεια διασφαλίζει ότι η βάση δεδομένων διατηρείται σε μια συνεπή κατάσταση, δηλαδή ότι οδηγείται από τη μια συνεπή κατάσταση στην άλλη. Αν μια συναλλαγή παραβιάζει κάποιο κανόνα της συνέπειας, ανακαλείται προκειμένου η βάση να έχει μόνο έγκυρα δεδομένα. Η απομόνωση αναφέρεται στην απαίτηση ότι όλες οι ενέργειες δεν μπορούν να έχουν πρόσβαση ή να δουν δεδομένα τα οποία τροποποιούνται εκείνη την στιγμή από μια συναλλαγή η οποία δεν έχει ακόμα ολοκληρωθεί. Τέλος, η μονιμότητα εγγυάται στον χρήστη ότι αν μια συναλλαγή ολοκληρωθεί επιτυχώς, τότε τα αποτελέσματα της δε θα χαθούν. Οι αλλαγές που έχει κάνει η συναλλαγή δε θα χαθούν σε περίπτωση απώλειας ρεύματος ή άλλης καταστροφής. Μιας και η εφαρμογή που αναπτύχθηκε περιλαμβάνει ηλεκτρονική αγορά προϊόντων είναι πολύ βασικό να διασφαλίζεται η σωστή λειτουργία της αγοράς. Επομένως, μια σχεσιακή βάση δεδομένων θα ήταν περισσότερο κατάλληλη.

Τέλος, η επιλογή μιας βάσης πρέπει να γίνεται λαμβάνοντας υπόψιν και το framework με το οποίο θα αναπτυχθεί η εφαρμογή. Το Django είναι περισσότερο συμβατό με τις σχεσιακές βάσεις δεδομένων, καθώς αυτές υποστηρίζονται από την επίσημη έκδοσή του. Για τη σύνδεσή του με τη MongoDB, όπως έχει αναφερθεί, χρησιμοποιείται ένα πακέτο, όμως περιορίζεται η λειτουργικότητά του και παρουσιάζονται μερικά προβλήματα κατά την εκτέλεση κάποιων ερωτημάτων προς τη βάση.

Κεφάλαιο 5

Υλοποίηση εφαρμογής

5.1 Εισαγωγή

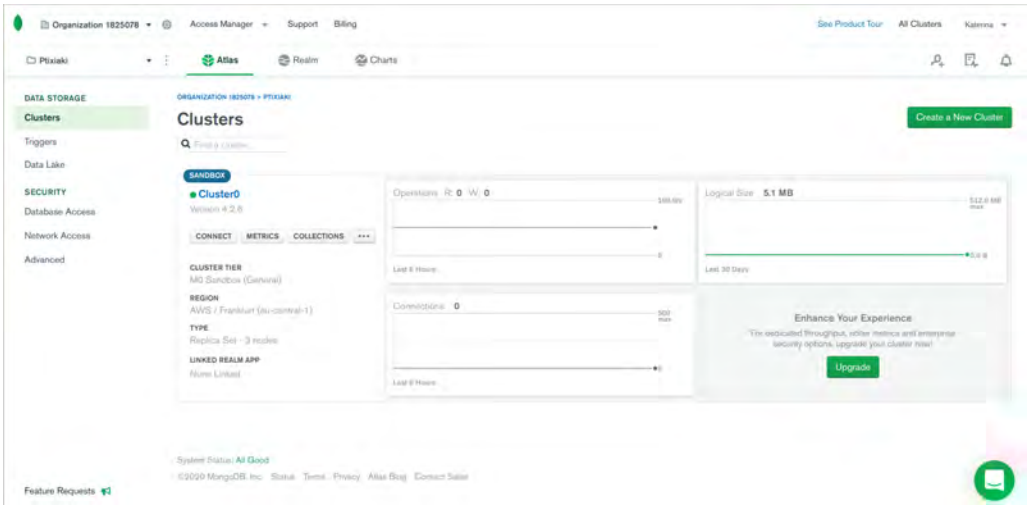
Στο κεφάλαιο αυτό παρουσιάζονται τα στάδια που ακολουθήθηκαν για την υλοποίηση της εφαρμογής, καθώς και οι πιο σημαντικές λειτουργίες που εκτελεί. Για την ανάλυση των λειτουργιών γίνεται αναφορά στα αντίστοιχα κομμάτια κώδικα.

5.2 Εγκατάσταση του Django και σύνδεση με τη MongoDB

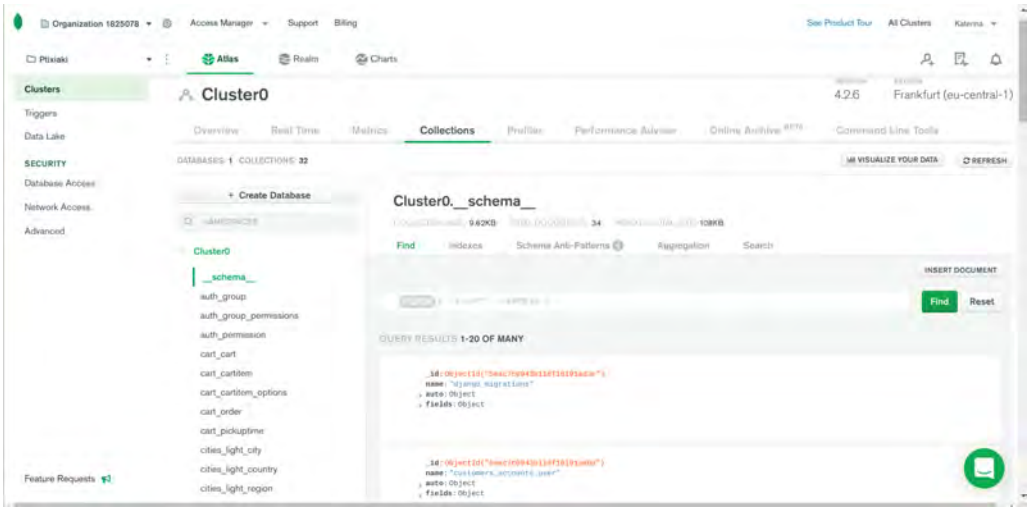
Για την ανάπτυξη της εφαρμογής χρειάστηκε αρχικά η δημιουργία ενός εικονικού, απομονωμένου περιβάλλοντος με τη χρήση του εργαλείου «virtualenv» της Python. Το εργαλείο αυτό δημιουργεί ένα φάκελο που περιλαμβάνει όλα τα απαραίτητα εκτελέσιμα αρχεία που μπορεί να χρειαστεί ένα project της Python. Η δημιουργία του εικονικού περιβάλλοντος είναι ένα απαραίτητο βήμα που διευκολύνει την ανάπτυξη μελλοντικών εφαρμογών στον ίδιο υπολογιστή, καθώς κάθε project θα μπορεί να χρησιμοποιεί διαφορετικά πακέτα και οι αλλαγές που γίνονται σε μία εφαρμογή δε θα επηρεάζουν καμία άλλη.

Μετά τη δημιουργία και ενεργοποίηση του εικονικού περιβάλλοντος έγινε εγκατάσταση του Django 2.1.5 με βάση τις οδηγίες της επίσημης σελίδας του [14]. Στη συνέχεια, δημιουργήθηκε ένα project με το όνομα «ptixiaki» και ένας διαχειριστής.

Η MongoDB παρέχει την υπηρεσία MongoDB Atlas [2], στην οποία ο χρήστης μπορεί να ενοικιάσει ένα διακομιστή αντί να χρησιμοποιήσει τη βάση τοπικά, επιλέγοντας μεταξύ τριών πακέτων. Το πακέτο που χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής είναι το δωρεάν πακέτο, το οποίο θέτει τον περιορισμό των 512 MB για το μέγεθος των δεδομένων



Σχήμα 5.1: Αρχική σελίδα της MongoDB Atlas. (Πηγή: [2])



Σχήμα 5.2: Σελίδα της MongoDB Atlas στην οποία ο διαχειριστής μπορεί να δει τις συλλογές και τα έγγραφα της βάσης της εφαρμογής του. (Πηγή: [2])

που θα περιλαμβάνονται στη βάση. Μετά τη δημιουργία ενός λογαριασμού Atlas, χρειάστηκε η κατασκευή ενός cluster, που έγινε με την επιλογή ενός παρόχου cloud και ενός ονόματος για το cluster. Στη συνέχεια, έπρεπε να οριστεί ένας χρήστης για τη βάση στον οποίο δόθηκαν δικαιώματα διαχειριστή. Η αρχική σελίδα της MongoDB Atlas φαίνεται στο σχήμα 5.1.

Εάν ο διαχειριστής επιλέξει το κουμπί «Collections» τότε θα μεταφερθεί στη σελίδα που περιλαμβάνονται οι συλλογές και τα έγγραφα της καθεμιάς. Το στιγμιότυπο της σελίδας παρουσιάζεται στο σχήμα 5.2.

Επειδή το Django δεν υποστηρίζει επίσημα τη MongoDB, για τη σύνδεσή τους ήταν απαραίτητη η εγκατάσταση ενός εργαλείου που ονομάζεται «djongo», το οποίο μετατρέπει τα ερωτήματα SQL σε ερωτήματα της MongoDB. Με την εγκατάσταση του djongo, εγκατα-

στάθηκε και το πακέτο «pymongo» που περιλαμβάνει διάφορα αρχεία, μεταξύ των οποίων και το «mongoclient.py». Το cluster που δημιουργήθηκε στη MongoDB Atlas περιλάμβανε ένα string σύνδεσης, το οποίο αντικατέστησε την παράμετρο «HOST» στο αρχείο «mongoclient.py», ώστε να γίνει η σύνδεση με τη βάση.

5.3 Django εφαρμογές

Κάθε project του Django μπορεί να περιλαμβάνει μία ή παραπάνω εφαρμογές (apps). Για να μη συγχέεται ο όρος «εφαρμογές» του Django με τις web εφαρμογές, δηλαδή τις ιστοσελίδες, θα γίνεται αναφορά στις πρώτες ως «Django εφαρμογές». Οι Django εφαρμογές είναι κομμάτια κώδικα που μπορούν να επαναχρησιμοποιηθούν σε πολλά projects και αναφέρονται σε διαφορετικές λειτουργίες μιας ιστοσελίδας. Για παράδειγμα, μία Django εφαρμογή δημιουργίας και ελέγχου ταυτότητας χρήστη μπορεί να χρησιμοποιηθεί σε κάθε ιστοσελίδα που απαιτεί σύνδεση και εγγραφή. Στην παρούσα εργασία δημιουργήθηκαν οι πέντε Django εφαρμογές: home, customers_accounts, stores_accounts, products και cart. Καθεμιά περιλαμβάνει μοντέλα, φόρμες και views.

5.3.1 Home

Στη Django εφαρμογή «home» γίνεται διαχείριση των λειτουργιών που έχουν σχέση με την αρχική σελίδα της εφαρμογής. Για να μπορεί ο χρήστης να βρίσκει καταστήματα στην πόλη που τον ενδιαφέρει είναι απαραίτητο να εισάγει το όνομα της πόλης. Για το λόγο αυτό χρησιμοποιήθηκε το πακέτο «django-cities-light» [15], το οποίο περιλαμβάνει μοντέλα με πόλεις από όλο τον κόσμο. Τα δεδομένα εξάγονται από τη γεωγραφική βάση «GeoNames» [16] και ο προγραμματιστής της εφαρμογής μπορεί να ορίσει τη χώρα από την οποία θέλει να εισαχθούν οι πόλεις. Σε συνδυασμό με το πακέτο αυτό, χρησιμοποιήθηκε και το πακέτο «django-autocomplete-light» [17], το οποίο εμφανίζει προτεινόμενες πόλεις ανάλογα με τα γράμματα που πληκτρολογεί ο χρήστης. Στη συγκεκριμένη Django εφαρμογή είναι δηλωμένο ένα μοντέλο που ονομάζεται «City», το οποίο περιλαμβάνει ένα ξένο κλειδί προς τον πίνακα City του πακέτου «django-cities-light». Υπάρχει μία φόρμα, η «CityForm», η οποία έχει ως πεδίο το πεδίο του πίνακα «City». Όταν γίνει αίτημα προς τον server για την αρχική σελίδα της εφαρμογής, το αίτημα θα περαστεί στο «home_view», το οποίο θα εμφανίσει το αντίστοιχο HTML template και τη φόρμα για την εισαγωγή πόλης από το χρήστη.

5.3.2 Customers_accounts

Η Django εφαρμογή «customers_accounts» είναι υπεύθυνη για τις ενέργειες που αφορούν το προφίλ ενός χρήστη. Περιλαμβάνει τα μοντέλα «User» και «CustomerProfile». Το πρώτο μοντέλο κληρονομεί ιδιότητες από το ήδη υπάρχον μοντέλο του Django, «AbstractBaseUser», το οποίο παρέχει την κύρια υλοποίηση ενός μοντέλου χρηστών, περιλαμβάνοντας κατακερματισμένους κωδικούς πρόσβασης. Έχει προστεθεί ένα πεδίο, το «business», ώστε να διαχωρίζονται οι λογαριασμοί χρηστών από αυτούς των καταστημάτων. Επίσης, για τη δημιουργία λογαριασμού δε χρειάζεται η επιλογή ονόματος χρήστη, καθώς το email χρησιμοποιείται σαν username. Το μοντέλο «CustomerProfile» περιλαμβάνει το ονοματεπώνυμο ενός χρήστη, το τηλέφωνο και το email του.

Για τη δημιουργία ενός προφίλ χρησιμοποιούνται δύο φόρμες. Η πρώτη ονομάζεται «RegisterForm» και περιλαμβάνει ένα πεδίο για το email και ένα για τον κωδικό πρόσβασης. Η φόρμα «CustomerProfileForm» περιέχει τα πεδία του πίνακα «CustomerProfile», δηλαδή όνομα, επώνυμο, τηλέφωνο και email. Για την ενημέρωση των στοιχείων του προφίλ ενός χρήστη χρησιμοποιείται η φόρμα «UpdateProfileForm», η οποία αρχικοποιείται με τα στοιχεία του συνδεδεμένου χρήστη. Εάν ο χρήστης επιθυμεί να αλλάξει το email του, τότε γίνεται έλεγχος στη βάση για την εύρεση χρήστη με το ίδιο email. Αν βρεθεί κάποια εγγραφή, θα εμφανιστεί μήνυμα λάθους στο χρήστη που θα τον ενημερώνει ότι το email που έχει εισάγει χρησιμοποιείται ήδη.

Συνολικά για την Django εφαρμογή «customers_accounts» έχουν δημιουργηθεί τα εξής views.

- **register_customer_view**

  Εμφανίζει τη σελίδα στην οποία ο χρήστης μπορεί να κάνει εγγραφή και περιλαμβάνει τις φόρμες «RegisterForm» και «CustomerProfileForm». Αν τα στοιχεία του χρήστη είναι σωστά, γίνεται σύνδεσή του στην εφαρμογή.

- **login_view**

  Το «login_view» χρησιμοποιεί τις συναρτήσεις «authenticate» και «login» που προσφέρονται από το Django. Η πρώτη συνάρτηση επαληθεύει την ταυτότητα ενός χρήστη, ελέγχοντας αν το email και ο κωδικός πρόσβασης ταιριάζουν με κάποιο λογαριασμό της βάσης. Εάν τα στοιχεία αυτά επαληθευτούν, τότε γίνεται σύνδεση μέσω της συνάρτησης «login».

- **logout_view**

  Αυτό το view είναι υπεύθυνο για την αποσύνδεση του χρήστη από την εφαρμογή, η οποία γίνεται με τη συνάρτηση «logout» του Django. Μετά την αποσύνδεση, ο χρήστης ανακατευθύνεται στην αρχική σελίδα.

- **change_password_view**

  Χρησιμοποιεί τη φόρμα του Django, «PasswordChangeForm», με την οποία μπορεί να γίνει αλλαγή του κωδικού πρόσβασης και εμφανίζει την αντίστοιχη σελίδα.

- **update_profile_view**

  Εμφανίζει τη φόρμα «UpdateForm» στο αντίστοιχο HTML template και την αρχικοποιεί με τα στοιχεία του συνδεδεμένου χρήστη. Εάν ο χρήστης αλλάξει κάποιο στοιχείο, γίνεται ενημέρωση της βάσης.

- **my_orders_view**

  Στο συγκεκριμένο view γίνεται ένα query για την αναζήτηση των παραγγελιών που έχει εκτελέσει ο συνδεδεμένος χρήστης. Στην HTML σελίδα εμφανίζεται ένας πίνακας με πληροφορίες για καθεμιά από αυτές τις παραγγελίες.

5.3.3 Stores_accounts

Η Django εφαρμογή «stores_accounts» περιλαμβάνει όσες λειτουργίες έχουν σχέση με το προφίλ ενός καταστήματος. Έχουν κατασκευαστεί τρία μοντέλα, δηλαδή πίνακες, τα οποία είναι τα: StoreCategory, OpeningTime και Store. Κάθε μοντέλο αποθηκεύεται στη βάση σαν συλλογή. Το μοντέλο «StoreCategory» περιέχει κατηγορίες εστιατορίων, όπως για παράδειγμα πιτσαρίες, κρεπερί, ψητοπωλεία, κ.λπ. Το μοντέλο «OpeningTime» περιλαμβάνει το ωράριο των καταστημάτων για κάθε ημέρα της εβδομάδας, ενώ στο μοντέλο «Store» αποθηκεύονται πληροφορίες σχετικά με ένα κατάστημα, όπως το όνομα, το τηλέφωνο, η διεύθυνση, κ.λπ. Στην εικόνα 5.3 φαίνεται ένα έγγραφο της συλλογής ”stores_accounts_store”, η οποία δημιουργείται από το μοντέλο «Store».

Όσον αφορά στις φόρμες, έχουν κατασκευαστεί οι StoreForm και OpeningTimeFormset. Η πρώτη περιλαμβάνει τα πεδία του μοντέλου «Store», ενώ η δεύτερη αποτελείται από 14 υπό-φόρμες, δύο για κάθε ημέρα της εβδομάδας. Για παράδειγμα, αν ένα κατάστημα είναι

Σχήμα 5.3: Έγγραφο της συλλογής που δημιουργείται από το μοντέλο «Store».

ανοιχτό εφτά ημέρες την εβδομάδα και ανοίγει δύο φορές τη μέρα θα χρειαστεί να συμπληρώσει όλες τις υπό-φόρμες με το ωράριό του.

Όπως έχει προαναφερθεί, όλες οι συναρτήσεις που ορίζουν τι θα συμβεί όταν ζητηθεί μια σελίδα από τον server, ονομάζονται views. Για τη συγκεκριμένη Django εφαρμογή έχουν υλοποιηθεί έντεκα views.

- **home_business_view**

  Η συνάρτηση αυτή εμφανίζει την αρχική σελίδα της εφαρμογής, όπως αυτή φαίνεται όταν είναι συνδεδεμένο ένα κατάστημα, στην οποία περιλαμβάνεται και ένας πίνακας με όσες παραγγελίες έχουν πραγματοποιηθεί από τους χρήστες και δεν έχουν γίνει ακόμα αποδεκτές. Το κατάστημα μπορεί είτε να αποδεχτεί είτε να απορρίψει την παραγγελία και αμέσως μετά θα σταλεί στο χρήστη ένα email επιβεβαίωσης ή απόρριψης, αντίστοιχα.

- **shops_view**

  Το συγκεκριμένο view εμφανίζει τη σελίδα που περιλαμβάνει όλα τα καταστήματα της πόλης, η οποία έχει επιλεχθεί σε προηγούμενο βήμα από το χρήστη. Επίσης, ελέγχει εάν υπάρχει κλειδί για το session του χρήστη, και αν όχι το δημιουργεί. Ακόμα, γίνεται εύρεση όλων των καλαθιών που αντιστοιχούν στο συγκεκριμένο session και στη συνέχεια διαγράφονται. Αυτό συμβαίνει γιατί όσο ο χρήστης είναι στην εφαρμογή δεν μπορεί να παραγγείλει ταυτόχρονα από δύο καταστήματα, οπότε χρειάζεται μόνο ένα καλάθι. Εάν ολοκληρώσει μια παραγγελία, μπορεί φυσικά να παραγγείλει ξανά.

- **register_store_view**

  Όπως υποδηλώνει και το όνομα της συγκεκριμένης συνάρτησης, σκοπός της είναι η διαχείριση των ενεργειών που αφορούν στη δημιουργία λογαριασμού ενός καταστήματος. Για την εγγραφή χρειάζεται μόνο το email του καταστήματος και ο κωδικός πρόσβασης, ενώ χρησιμοποιείται η φόρμα «RegisterForm» που είναι δηλωμένη στη Django εφαρμογή «customers_accounts».

- **register_store_info_view**

  Το παραπάνω view εμφανίζει τη σελίδα στην οποία το κατάστημα μπορεί να εισάγει πληροφορίες για την επιχείριση. Γίνεται χρήση της φόρμας «StoreForm», επομένως τα πεδία που μπορεί να εισάγει είναι το όνομα, το τηλέφωνο, το email, η περιγραφή του, οι κατηγορίες εστιατορίου στις οποίες ανήκει, η διεύθυνση και μια φωτογραφία του. Σε αυτή τη σελίδα εμφανίζεται επίσης η φόρμα «OpeningTimeFormset» στην οποία το κατάστημα μπορεί να εισάγει τις μέρες και ώρες λειτουργίας του. Εάν το κατάστημα έχει ήδη εισάγει αυτές τις πληροφορίες, όταν ανοίξει τη συγκεκριμένη σελίδα εμφανίζονται όλα τα πεδία συμπληρωμένα με τα στοιχεία του, τα οποία μπορεί να επεξεργαστεί.

- **login_view**

  Το «login_view» χρησιμοποιεί τις συναρτήσεις «authenticate» και «login» που προσφέρονται από το Django. Λειτουργεί ακριβώς όπως το «login_view» της Django εφαρμογής «customers_accounts».

- **logout_view**

  Αυτό το view είναι υπεύθυνο για την αποσύνδεση του καταστήματος από την εφαρμογή, η οποία γίνεται με τη συνάρτηση «logout» του Django. Μετά την αποσύνδεση, το κατάστημα ανακατευθύνεται στην αρχική σελίδα.

- **orders_by_date_view**

  Εδώ εμφανίζεται η σελίδα στην οποία το κατάστημα μπορεί να επιλέξει να δει ποιες παραγγελίες έχουν ημερομηνία παραλαβής εκείνη την ημέρα, ποιες έχουν ήδη παραληφθεί και ποιες πρέπει να παραληφθούν μελλοντικά.

- **todays_confirmed_orders_view**

Εάν το κατάστημα μεταβεί στη σελίδα που εμφανίζονται οι παραγγελίες εκείνης της ημέρας, θα γίνει κλήση του συγκεκριμένου view. Αρχικά, υπολογίζεται η ημερομηνία και η ώρα που γίνεται το αίτημα. Στη συνέχεια, γίνεται ένα query προς στη βάση στο οποίο ζητούνται όσες παραγγελίες έχουν ημερομηνία παραλαβής την ημερομηνία του αιτήματος και ώρα παραλαβής μεγαλύτερη ή ίση από την ώρα του αιτήματος. Εάν βρεθούν παραγγελίες που πληρούν τα παραπάνω κριτήρια, αυτές εμφανίζονται στη σελίδα και δίπλα στην καθεμιά εμφανίζεται ένα checkbox, το οποίο μπορεί να επιλεχθεί από το κατάστημα μόλις η αντίστοιχη παραγγελία είναι έτοιμη προς παραλαβή.

- **picked_up_orders_view**

  Σε αυτό το view γίνεται ένα query για να βρεθούν οι παραγγελίες που έχουν παραληφθεί μέχρι την ώρα που γίνεται το αίτημα από το κατάστημα.

- **future_to_pick_up_orders_view**

  Εδώ γίνεται ένα query για να βρεθούν οι παραγγελίες που έχουν ημερομηνία παραλαβής μεγαλύτερη της ημερομηνίας που γίνεται το αίτημα από το κατάστημα.

- **store_details_view**

  Το «store_details_view» εμφανίζει τη σελίδα που βλέπει ο χρήστης με τις πληροφορίες ενός καταστήματος.

5.3.4 Products

Όσες ενέργειες αφορούν στη διαχείριση των προϊόντων ενός καταστήματος περιλαμβάνονται στη Django εφαρμογή «products». Τα μοντέλα που έχουν οριστεί είναι τα εξής: ProductCategory, IngredientsCategory, Ingredients, IngredientPrice και Product.

Στο μοντέλο «ProductCategory» ορίζονται οι δυνατές κατηγορίες στις οποίες μπορεί να ανήκει ένα προϊόν. Για παράδειγμα, τα ορεκτικά, τα μαγειρευτά, οι αλμυρές και γλυκές κρέπες και οι πίτσες είναι κατηγορίες προϊόντων. Το μοντέλο «IngredientsCategory» περιλαμβάνει τις κατηγορίες στις οποίες ανήκουν τα πρόσθετα υλικά που μπορεί να περιέχει ένα προϊόν. Εάν ένας χρήστης επιλέξει να παραγγείλει μια αλμυρή κρέπα, θα πρέπει να προσθέσει τα υλικά που επιθυμεί, όπως τα αλλαντικά και τα τυροκομικά. Τα αλλαντικά και τα τυροκομικά ορίζονται ως αντικείμενα του μοντέλου «IngredientsCategory» και όπως τα υπόλοιπα αντικείμενα έχουν μια σχέση «Πολλά προς Πολλά» με το μοντέλο «ProductCategory».

Στο μοντέλο «Ingredients» περιλαμβάνονται τα υλικά που μπορούν να προστεθούν σε ένα προϊόν, όπως οι πατάτες, που μπορούν να περιέχονται σε μια αλμυρή κρέπα ή ένα σάντουιτς. Όσες πληροφορίες χαρακτηρίζουν ένα προϊόν περιλαμβάνονται στο μοντέλο «Product». Πιο συγκεκριμένα, μεταξύ των πεδίων του δηλώνονται το όνομα, η περιγραφή, η τιμή, η ποσότητα, η κατηγορία του προϊόντος, η φωτογραφία του, το κατάστημα στο οποίο υπάρχει το προϊόν και τα έξτρα υλικά που μπορούν να προστεθούν σε αυτό. Τέλος, στο μοντέλο «IngredientPrice», για κάθε προϊόν που περιλαμβάνει έξτρα υλικά ορίζεται η τιμή για καθένα από αυτά.

Η φόρμα «ProductForm» χρησιμοποιείται για την καταχώρηση νέων προϊόντων και περιλαμβάνει πεδία που χαρακτηρίζουν ένα προϊόν, όπως το όνομα, η τιμή, η περιγραφή και η κατηγορία στην οποία ανήκει. Σε περίπτωση που για την αγορά του προϊόντος χρειάζεται η επιλογή έξτρα υλικών, χρησιμοποιείται η φόρμα «OptionsForm», στην οποία εισάγονται τα έξτρα υλικά από τα οποία μπορεί να επιλέξει ο πελάτης.

Τα views που περιλαμβάνει η Django εφαρμογή «products» είναι τα εξής:

- **detailpage_view**

  Είναι υπεύθυνο για την εμφάνιση των προϊόντων που περιλαμβάνονται στο επιλεγμένο κατάστημα. Μόλις ο χρήστης ζητήσει τη σελίδα που αντιστοιχεί στο «detailpage_view», δημιουργείται ένα άδειο καλάθι, το οποίο αφορά το κατάστημα στο οποίο έχει εισέλθει.

- **register_store_menu_view**

  Το view αυτό διαχειρίζεται τις λειτουργίες που είναι σχετικές με την εγγραφή και την ανανέωση του μενού από τους καταστηματάρχες. Αφού το κατάστημα ολοκληρώσει την εγγραφή του, μπορεί να εισάγει τα φαγητά που επιθυμεί, συμπληρώνοντας τη φόρμα «ProductForm» με το όνομα του φαγητού, τη φωτογραφία, την τιμή, την περιγραφή και την κατηγορία στην οποία ανήκει. Όποτε θελήσει μπορεί να επεξεργαστεί τις πληροφορίες για το προϊόν ή ακόμα και να το διαγράψει.

- **register_store_options_view**

  Κατά την εισαγωγή ενός προϊόντος στο μενού ενός καταστήματος, είναι δυνατή η προσθήκη επιπλέον προαιρετικών υλικών, τα οποία ο χρήστης θα μπορεί να προσθέτει στο συγκεκριμένο φαγητό όταν το παραγγέλνει. Η εισαγωγή αυτών των υλικών στο μενού

γίνεται μέσω του register_store_options_view, στο οποίο χρησιμοποιείται η φόρμα OptionsForm, η οποία έχει αναλυθεί παραπάνω.

- **price_view**

  Χρησιμοποιείται για την καταχώρηση των τιμών των έξτρα υλικών. Κάθε φορά που το κατάστημα προσθέτει ένα υλικό, το οποίο δεν υπάρχει ήδη ως επιλογή σε κάποιο προϊόν του καταστήματος, καλείται να εισάγει και την τιμή του. Στην περίπτωση που το υλικό που προστίθεται σε ένα προϊόν διατίθεται και σε κάποιο διαφορετικό προϊόν του καταστήματος, δε χρειάζεται να γίνει εισαγωγή της τιμής του, καθώς εισάγεται αυτόματα η τιμή που είχε επιλεχθεί αρχικά. Για παράδειγμα, αν σε ένα κατάστημα που προσφέρει καφέ, δίνεται η δυνατότητα επιλογής μαύρης ζάχαρης στον cappuccino, τότε αν προστεθεί και σαν επιλογή στον espresso, δε θα χρειαστεί να γίνει ξανά εισαγωγή της τιμής.

  Για την καταχώρηση των τιμών δεν ήταν δυνατό να χρησιμοποιηθεί μία μόνο Django φόρμα, καθώς τα υλικά που εισάγονται μπορεί να είναι περισσότερα από ένα. Έτσι χρησιμοποιήθηκε ένα formset, το οποίο είναι μία ομάδα φορμών που μπορούν να εμφανιστούν στην ίδια σελίδα και να αποθηκευτούν με ένα μόνο post request. Οι υποφόρμες που δημιουργούνται είναι όσες και το πλήθος των υλικών για τα οποία πρέπει να καταχωρηθεί η τιμή. Προκειμένου να αποθηκευτεί στη βάση η τιμή για κάθε υλικό, γίνεται ένα for loop το οποίο ελέγχει την τιμή που εισήχθη σε κάθε υπο-φόρμα του formset.

- **edit_price_view**

  Στο συγκεκριμένο view περιλαμβάνεται ο κώδικας που χρησιμοποιείται για την αλλαγή της τιμής ενός ή περισσότερων πρόσθετων υλικών. Αρχικά, γίνεται εύρεση όλων των υλικών που διατίθενται στα προϊόντα του καταστήματος. Στη συνέχεια, όπως και στο price_view, δημιουργείται ένα formset, το οποίο περιλαμβάνει υπο-φόρμες για κάθε διαθέσιμο υλικό. Αφού γίνουν οι επιθυμητές αλλαγές στις τιμές των υλικών, με τη χρήση ενός for loop οι νέες τιμές αποθηκεύονται στη βάση δεδομένων και γίνεται ανακατεύθυνση του χρήστη στην αρχική σελίδα της εφαρμογής.

- **edit_menu_view**

  Είναι υπεύθυνο για την προβολή της σελίδας στην οποία μπορεί να γίνει τροποποί-

ηση του μενού του καταστήματος. Για κάθε προϊόν του μενού υπάρχει η επιλογή επεξεργασίας ή διαγραφής του. Εάν επιλεγεί η επεξεργασία του προϊόντος, γίνεται ανακατεύθυνση στη σελίδα που αντιστοιχεί στο register_store_menu_view, όπου μπορεί να οριστούν εκ νέου τα ακόλουθα πεδία: φωτογραφία, όνομα, περιγραφή, κατηγορία, τιμή και διαθέσιμη ποσότητα του προϊόντος. Σε περίπτωση διαγραφής του προϊόντος γίνεται ανακατεύθυνση στη σελίδα που αντιστοιχεί στο delete_product_view, που αναλύεται παρακάτω.

- **delete_product_view**

  Η σελίδα διαγραφής ενός προϊόντος, η οποία γίνεται render από το delete_product_view, περιλαμβάνει τις λεπτομέρειες του προϊόντος που έχει επιλεγεί για διαγραφή και δύο κουμπιά για επιβεβαίωση ή ακύρωση διαγραφής. Και στις δύο περιπτώσεις γίνεται ανακατεύθυνση στη σελίδα της επεξεργασίας του μενού.

5.3.5 Cart

Στη Django εφαρμογή «cart» περιλαμβάνονται όσες λειτουργίες σχετίζονται με την επιλογή και αγορά προϊόντων από τους χρήστες. Τα μοντέλα είναι τα: Cart, CartItem, PickUpTime και Order.

Το μοντέλο «Cart» περιλαμβάνει τα έξι πεδία: user, session, store, total, date_added, και date_updated. Το πεδίο «user» όπως είναι φυσικό, αποτελεί ξένο κλειδί του πίνακα «User» της Django εφαρμογής «customers_accounts». Το πεδίο «session» είναι ξένο κλειδί του μοντέλου «Session» που περιλαμβάνει το Django. Ένα session είναι ο μηχανισμός που αποθηκεύει πληροφορία σχετικά με την αλληλεπίδραση μεταξύ ενός web browser και μιας ιστοσελίδας. Τα sessions επιτρέπουν την αποθήκευση δεδομένων σε κάθε browser και όποτε ο browser συνδεθεί, τα δεδομένα είναι διαθέσιμα. Το πεδίο «store» είναι ξένο κλειδί του μοντέλου «Store» της Django εφαρμογής «stores_accounts» και χρησιμοποιείται για να αντιστοιχίζει κάθε καλάθι με ένα κατάστημα. Στο πεδίο «total» αποθηκεύεται το συνολικό ποσό προς πληρωμή, το οποίο ισούται με το άθροισμα των τιμών των προϊόντων που περιλαμβάνονται στο καλάθι. Μέσω της συνάρτησης «update_total», κάθε φορά που προστίθεται ένα προϊόν στο καλάθι, το συνολικό ποσό ενημερώνεται. Η ημερομηνία δημιουργίας του καλαθιού αποθηκεύεται στο πεδίο «date_added», ενώ η ημερομηνία τελευταίας τροποποίησης στο πεδίο «date_updated».

Η σχέση των προϊόντων που περιλαμβάνονται σε ένα καλάθι με το καλάθι ορίζεται μέσω του μοντέλου «CartItem». Το πεδίο «cart» είναι ξένο κλειδί του μοντέλου «Cart» που αναλύθηκε παραπάνω και το πεδίο «product» είναι ξένο κλειδί του μοντέλου «Product» της Django εφαρμογής «products». Για την αποθήκευση της επιθυμητής ποσότητας παραγγελίας ενός προϊόντος χρησιμοποιείται το πεδίο «quantity», ενώ για την ημερομηνία προσθήκης στο καλάθι και τροποποίησης χρησιμοποιούνται τα πεδία «date_added» και «date_updated», αντίστοιχα. Καθώς υπάρχουν προϊόντα που για την παραγγελία τους χρειάζεται να προστεθούν κάποια επιπλέον υλικά, έχει δημιουργηθεί το πεδίο «options», το οποίο έχει μια σχέση «Πολλά προς Πολλά» με το μοντέλο «IngredientPrice». Στο πεδίο «total_price» αποθηκεύεται η τελική τιμή του προϊόντος που καθορίζεται από το άθροισμα της κανονικής του τιμής και των τιμών των επιπλέον υλικών που έχουν επιλεχθεί για το προϊόν.

Στο μοντέλο «PickUpTime» περιλαμβάνονται τα δύο πεδία «date» και «hour». Το πρώτο πεδίο χρησιμεύει για την αποθήκευση της ημερομηνίας στην οποία ο χρήστης θέλει να είναι έτοιμη η παραγγελία του, ώστε να περάσει από το κατάστημα για να την παραλάβει. Το πεδίο «hour» χρησιμοποιείται αντίστοιχα για την αποθήκευση της ώρας παραλαβής.

Τέλος, το μοντέλο «Order» είναι αυτό που αποθηκεύει πληροφορίες για τις παραγγελίες που εκτελούνται. Αρχικά, ορίζεται το πεδίο «cart» που είναι ξένο κλειδί του μοντέλου «Cart» και αναφέρεται στο καλάθι, τα προϊόντα του οποίου ο χρήστης επιθυμεί να παραγγείλει. Το πεδίο «pickup» είναι ξένο κλειδί του μοντέλου «PickUpTime», στο οποίο όπως έχει αναφερθεί, αποθηκεύονται η ημερομηνία και η ώρα παραλαβής της παραγγελίας. Αν η παραγγελία έχει πληρωθεί, η τιμή του πεδίου «paid» γίνεται true, διαφορετικά παραμένει false. Αφού εκτελεστεί η παραγγελία, το κατάστημα ειδοποιείται και πρέπει να απαντήσει εάν την αποδέχεται ή όχι. Το πεδίο «confirmed» γίνεται true όταν το κατάστημα απαντήσει στην παραγγελία και σε περίπτωση που την αποδεχτεί θα γίνει true και το πεδίο «accepted». Το πεδίο «ordered_by» είναι ξένο κλειδί στο μοντέλο «CustomerProfile» της Django εφαρμογής «customers_accounts» και δείχνει ποιος είναι ο χρήστης που εκτέλεσε την παραγγελία. Τέλος, στο πεδίο «prepared» εμφανίζεται εάν το κατάστημα έχει ετοιμάσει την παραγγελία ή όχι.

Οι φόρμες που έχουν δημιουργηθεί στη Django εφαρμογή «cart» είναι οι: CustomerProfileForm, PickupDateForm, PickupTimeForm και PaymentForm. Η «CustomerProfileForm» χρησιμοποιείται για την εισαγωγή των προσωπικών στοιχείων του χρήστη που κάνει μια παραγγελία και συνδέεται με το μοντέλο «CustomerProfile». Περιλαμβάνει τα πεδία του ονό-

ματος, του επιθέτου, του τηλεφώνου επικοινωνίας και του email. Εάν ο χρήστης είναι συνδεδεμένος στην εφαρμογή, τότε τα πεδία της φόρμας συμπληρώνονται αυτόματα με βάση τα στοιχεία του προφίλ του χρήστη. Σε περίπτωση που ο χρήστης δεν είναι εγγεγραμμένος θα πρέπει να εισάγει τα απαιτούμενα στοιχεία. Στην ίδια σελίδα εμφανίζεται και η φόρμα «PickupDateForm», η οποία περιλαμβάνει ένα πεδίο ημερομηνίας και μέσω αυτής γίνεται επιλογή της ημέρας παραλαβής της παραγγελίας. Ο τρόπος που επιλέγονται οι ημερομηνίες που εμφανίζονται στη φόρμα προς επιλογή θα αναλυθεί αργότερα στα views. Η φόρμα «PickupTimeForm» περιλαμβάνει το πεδίο της ώρας παραλαβής, ενώ η «PaymentForm» περιλαμβάνει τα πεδία για την εισαγωγή των στοιχείων της χρεωστικής ή πιστωτικής κάρτας με την οποία θα γίνει η πληρωμή της παραγγελίας. Τα πεδία της φόρμας αυτής προέρχονται από το πακέτο «django-credit-cards» [18] που προσφέρει πεδία φορμών για πιστωτικές κάρτες.

Τα views που διαχειρίζονται τις ενέργειες των παραγγελιών και των καλαθιών είναι έξι, ενώ μέσα σε αυτά καλούνται και τρεις διαφορετικές συναρτήσεις.

- **add_to_cart_view**

  Όταν ο χρήστης προσθέσει ένα προϊόν στο καλάθι, καλείται το συγκεκριμένο view. Αρχικά, γίνεται έλεγχος για το αν το προϊόν που προστίθεται στο καλάθι, υπάρχει ήδη. Σε αυτή την περίπτωση, αν τα έξτρα υλικά είναι τα ίδια αυξάνεται η ποσότητα του προϊόντος στο καλάθι, αλλιώς εισάγεται σαν διαφορετικό προϊόν. Στη συνέχεια, υπολογίζεται η τιμή του προϊόντος, που περιλαμβάνει και την τιμή κάθε επιπλέον υλικού και καλείται η συνάρτηση «update_total», η οποία ανανεώνει την τιμή του πεδίου «total».

- **update_product_view**

  Το view αυτό καλείται για την τροποποίηση των προϊόντων που περιλαμβάνονται στο καλάθι. Ο χρήστης μπορεί να αλλάξει την ποσότητα ενός προϊόντος, καθώς και τα επιπλέον υλικά που έχουν επιλεχθεί για το προϊόν. Μετά την επεξεργασία, η τιμή του προϊόντος υπολογίζεται ξανά και αλλάζει στη βάση δεδομένων.

- **delete_product_view**

  Στο «delete_product_view», αφού διαγραφεί το επιλεγμένο προϊόν από το καλάθι, ανανεώνεται το συνολικό ποσό του καλαθιού και γίνεται ανακατεύθυνση στη σελίδα με τα προϊόντα του καταστήματος.

- **order_view**

  Αφού ο χρήστης έχει ολοκληρώσει την προσθήκη προϊόντων στο καλάθι του, μπορεί να συνεχίσει στη σελίδα που θα εισάγει τα στοιχεία του μέσω της φόρμας «CustomerProfileForm». Για να επιλέξει την ημέρα κατά την οποία επιθυμεί να παραλάβει την παραγγελία του από το κατάστημα χρησιμοποιείται η φόρμα «PickupDateForm». Αρχικά, στο «order_view» γίνεται αναζήτηση του ωραρίου του καταστήματος στη βάση. Στη συνέχεια, σαν πρώτη δυνατή ημερομηνία παραλαβής τίθεται η ημερομηνία παραγγελίας αν το κατάστημα είναι ανοιχτό τη συγκεκριμένη μέρα, διαφορετικά ορίζεται η κοντινότερη ημερομηνία στην οποία το κατάστημα λειτουργεί. Το εύρος ημερομηνιών που δίνεται στο χρήστη περιλαμβάνει τις εφτά κοντινότερες ημέρες λειτουργίας του καταστήματος. Ο χρήστης επιλέγει την επιθυμητή ημερομηνία και στη συνέχεια ανακατευθύνεται στη σελίδα επιλογής ώρας.

- **order_time_view**

  Χρησιμοποιείται για την επιλογή της ώρας παραλαβής μέσω της φόρμας «PickupTimeForm». Όταν η επιλεγμένη ημέρα παραλαβής είναι ίδια με την ημέρα παραγγελίας, τότε η πρώτη διαθέσιμη ώρα που εμφανίζεται στο χρήστη είναι μετά από κάποιο χρονικό διάστημα, ώστε να υπάρχει ένα περιθώριο προετοιμασίας της παραγγελίας από το κατάστημα. Εάν ένα κατάστημα είναι ανοιχτό και μετά τα μεσάνυχτα, τότε οι ώρες αυτές θα εμφανιστούν όταν ο χρήστης επιλέξει ως ημέρα παραλαβής την επόμενη ημέρα. Μετά την επιλογή της ώρας παραλαβής, ο χρήστης ανακατευθύνεται στη σελίδα πληρωμής.

- **payment_view**

  Η φόρμα «PaymentForm» εμφανίζεται στη σελίδα που αντιστοιχεί στο «payment_view» και περιλαμβάνει τα τρία πεδία στα οποία εισάγονται τα στοιχεία της χρεωστικής ή πιστωτικής κάρτας του χρήστη. Εάν τα στοιχεία είναι σωστά, τότε στο χρήστη εμφανίζεται ένα μήνυμα επιτυχούς πληρωμής και το αντίστοιχο κατάστημα ενημερώνεται για την καινούργια παραγγελία.

Κεφάλαιο 6

Διεπαφή χρήστη (User interface)

6.1 Εισαγωγή

Για την κατασκευή του user interface χρησιμοποιήθηκε το έτοιμο HTML template [10], ώστε η εφαρμογή να είναι όσο το δυνατόν πιο εύκολη για τους χρήστες χωρίς όμως να υστερεί σε λειτουργικότητα. Στις περισσότερες σελίδες του template έγιναν αλλαγές για να προσαρμοστούν στις ακριβείς απαιτήσεις της εφαρμογής. Η εφαρμογή, όπως είναι γνωστό, περιλαμβάνει δύο κατηγορίες χρηστών: όσους είναι απλοί χρήστες που επιθυμούν να παραγγείλουν φαγητό και όσους είναι χρήστες που θέλουν να εισάγουν το κατάστημά τους στην εφαρμογή. Για το λόγο αυτό, η παρουσίαση της εφαρμογής θα χωριστεί σε δύο υποενότητες, μία για κάθε κατηγορία χρηστών.

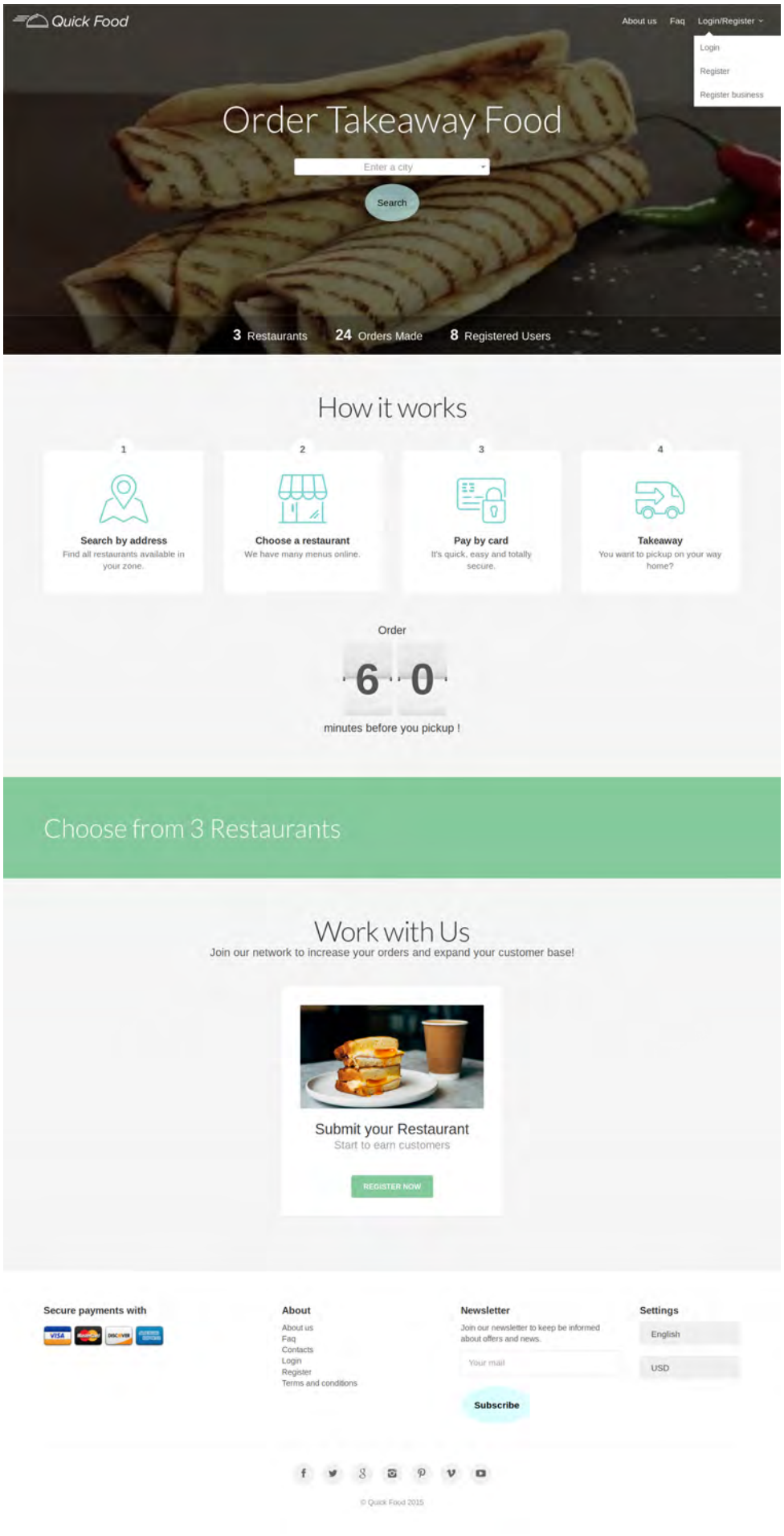
6.2 Παρουσίαση της εφαρμογής

6.2.1 Σελίδες με πρόσβαση από απλούς χρήστες

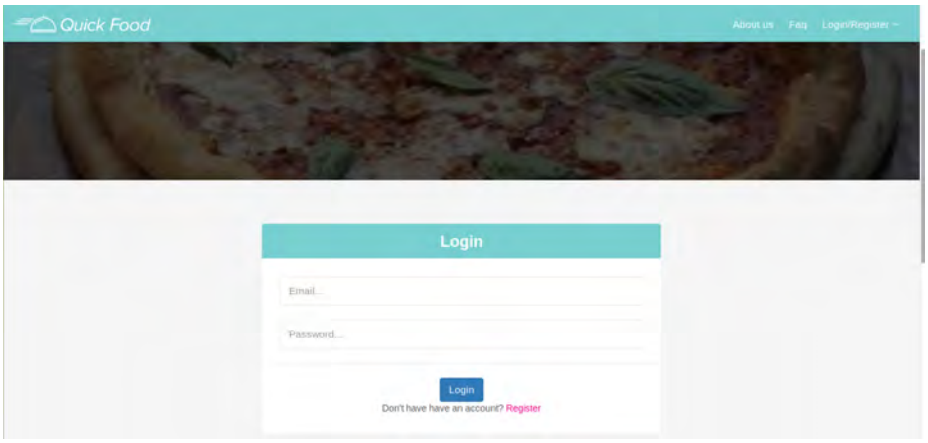
Στο σχήμα 6.1 της επόμενης σελίδας φαίνεται η αρχική σελίδα της εφαρμογής, η οποία είναι ίδια για όποιον χρήστη δε διαθέτει λογαριασμό. Πάνω δεξιά στη σελίδα υπάρχει η επιλογή «Login/Register» που περιλαμβάνει ένα dropdown μενού που διαθέτει τις επιλογές «Login», «Register» και «Register business». Όπως είναι λογικό, η επιλογή «Register business» αφορά τους χρήστες που διαθέτουν κάποιο εστιατόριο, το οποίο θέλουν να εισάγουν στην εφαρμογή. Στην αρχική σελίδα φαίνεται επίσης και ο αριθμός των εστιατορίων που έχουν εγγραφεί στην εφαρμογή, το σύνολο των παραγγελιών που έχουν εκτελεστεί και ο

αριθμός των εγγεγραμμένων χρηστών. Ακόμα, αναγράφεται η διαδικασία την οποία πρέπει να ακολουθήσει κανείς για να ολοκληρώσει μια παραγγελία. Για την εκτέλεση μιας παραγγελίας δεν είναι απαραίτητη η ύπαρξη λογαριασμού, οπότε ο χρήστης μπορεί απευθείας να εισάγει την περιοχή για την οποία ενδιαφέρεται.

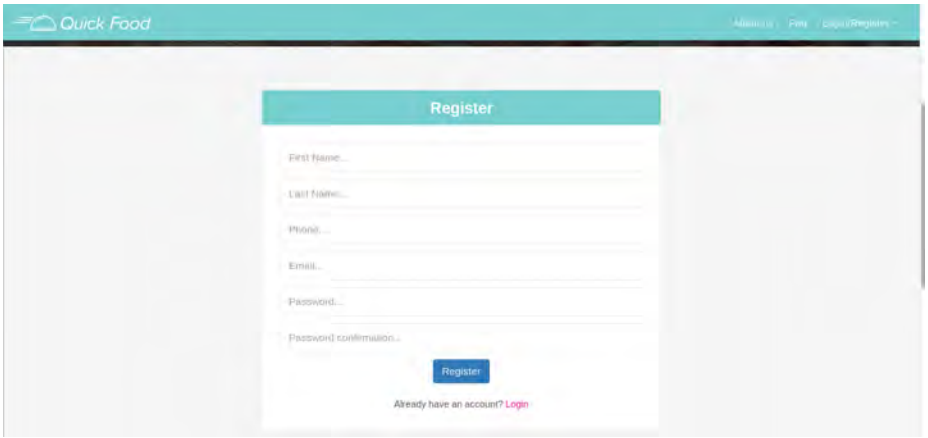
Στην εικόνα 6.2 εμφανίζεται η φόρμα σύνδεσης ενός χρήστη που περιλαμβάνει τα πεδία email και password. Εάν ο χρήστης συνδεθεί επιτυχώς θα ανακατευθυνθεί ξανά στην αρχική σελίδα. Για την εγγραφή ενός νέου απλού χρήστη, πρέπει να συμπληρωθούν τα πεδία First Name, Last Name, Phone, Email, Password και Password confirmation, όπως φαίνεται στην εικόνα 6.3.



Σχήμα 6.1: Αρχική σελίδα της εφαρμογής

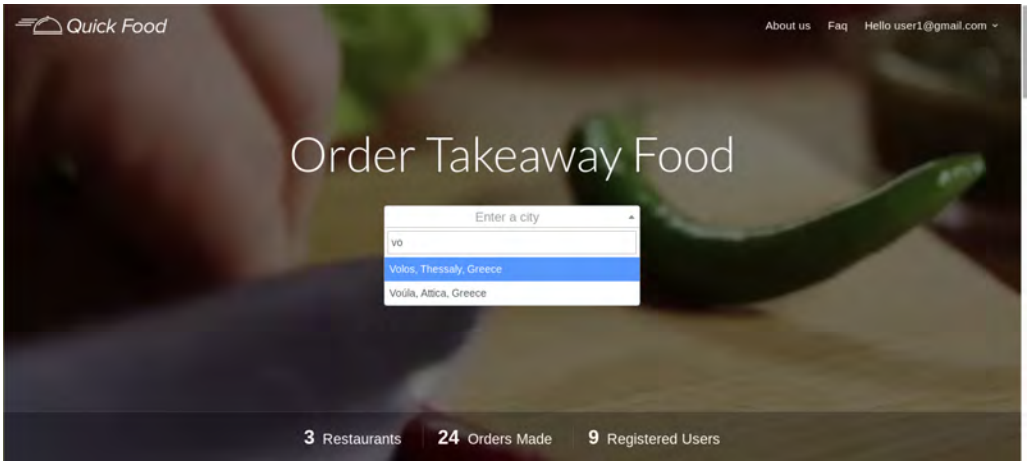


Σχήμα 6.2: Σύνδεση χρήστη



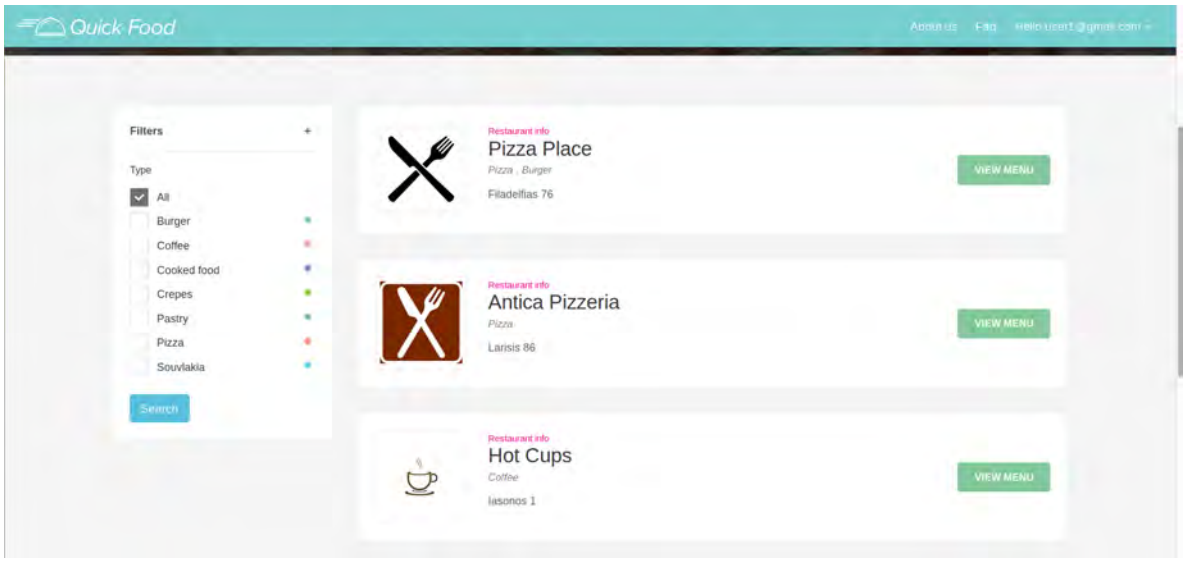
Σχήμα 6.3: Εγγραφή χρήστη

Καθώς ο χρήστης ξεκινά να πληκτρολογεί την περιοχή για την οποία θέλει να αναζητήσει καταστήματα, εμφανίζονται προτεινόμενες περιοχές ανάλογα με τα γράμματα τα οποία έχει πληκτρολογήσει. Όταν επιλέξει την επιθυμητή περιοχή μπορεί να πατήσει το κουμπί «Search» για να γίνει αναζήτηση και να ανακατευθυνθεί στη σελίδα με τα καταστήματα.

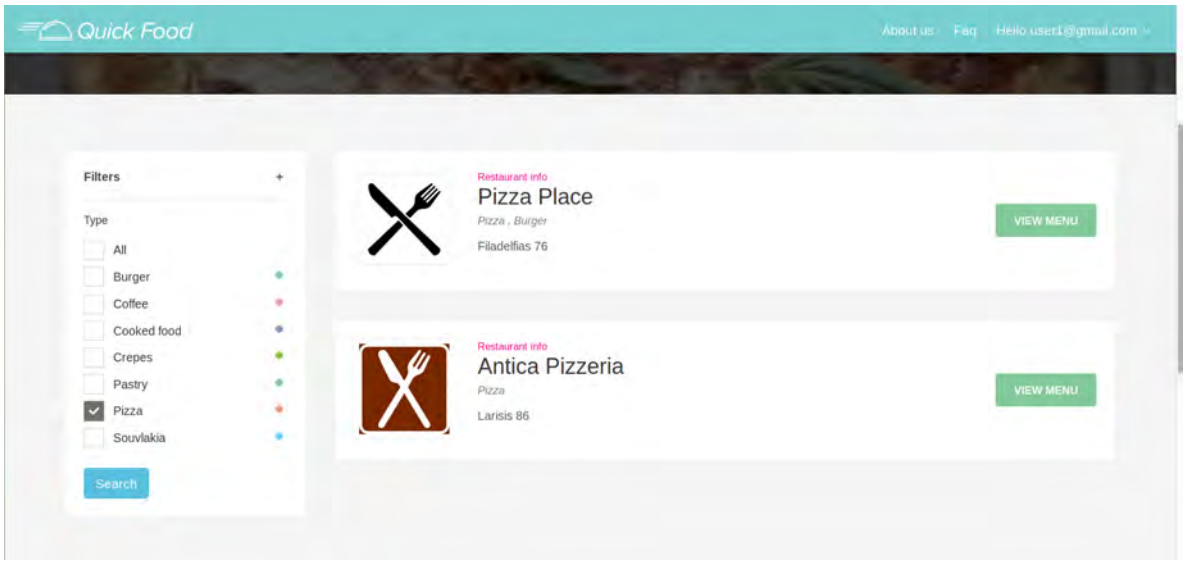


Σχήμα 6.4: Αναζήτηση καταστημάτων με βάση την περιοχή

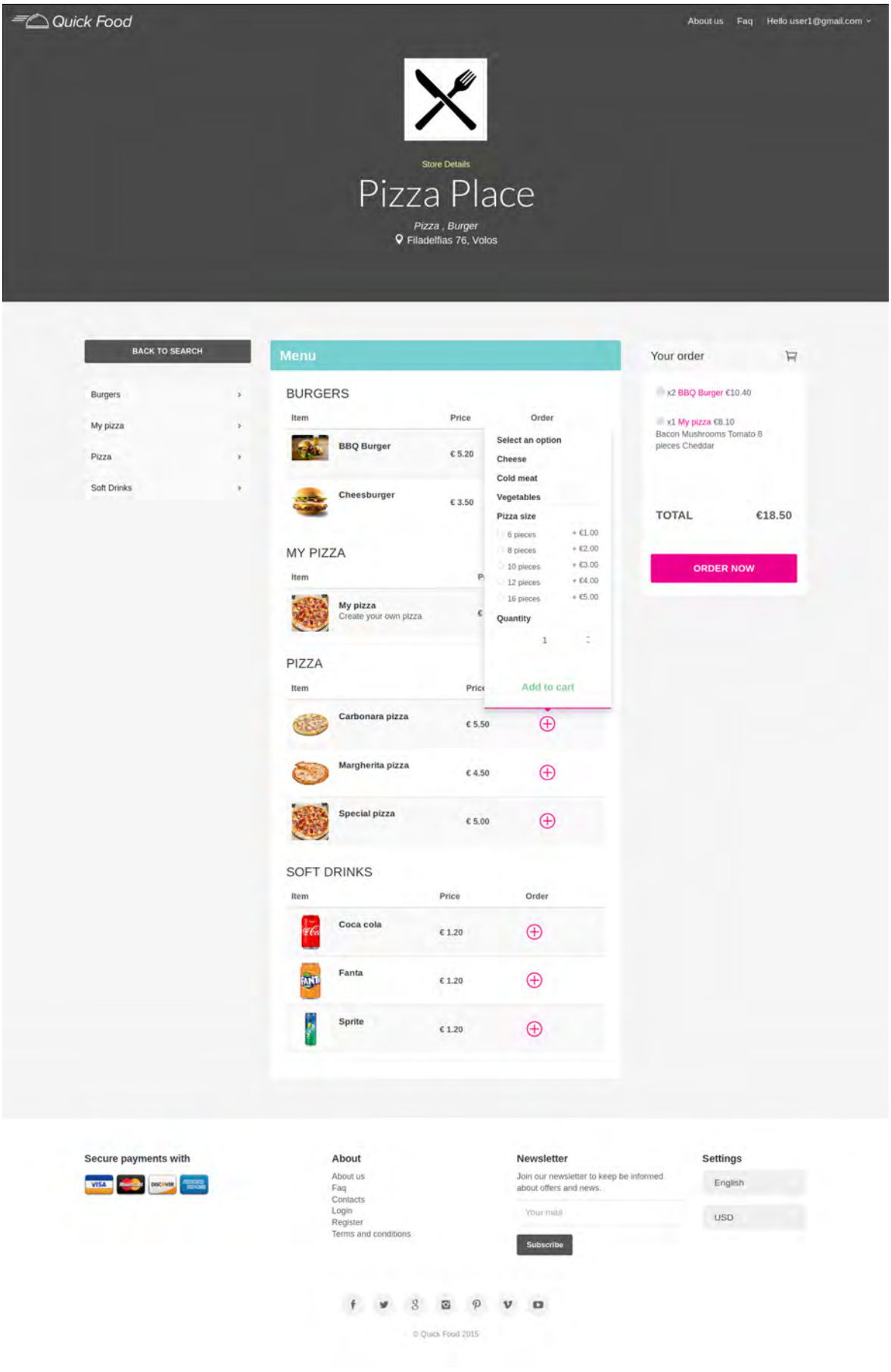
Στο σχήμα 6.5 φαίνεται η σελίδα που βλέπουν οι χρήστες μόλις επιλέξουν μια περιοχή. Εμφανίζονται τα καταστήματα της περιοχής από τα οποία μπορεί να επιλέξει ο χρήστης να παραγγείλει. Αριστερά υπάρχει η δυνατότητα φιλτραρίσματος των επιλογών με βάση την κουζίνα. Εάν ο χρήστης επιλέξει την επιλογή «Pizza», τότε θα εμφανιστούν μόνο τα εστιατόρια της περιοχής που προσφέρουν πίτσα, όπως φαίνεται και στο σχήμα 6.6. Κάτω από κάθε κατάστημα εμφανίζονται οι κατηγορίες κουζίνας στις οποίες ανήκει, καθώς και η διεύθυνσή του. Πατώντας το κουμπί «VIEW MENU», ο χρήστης μπορεί να δει το μενού του αντίστοιχου καταστήματος.



Σχήμα 6.5: Σελίδα καταστημάτων



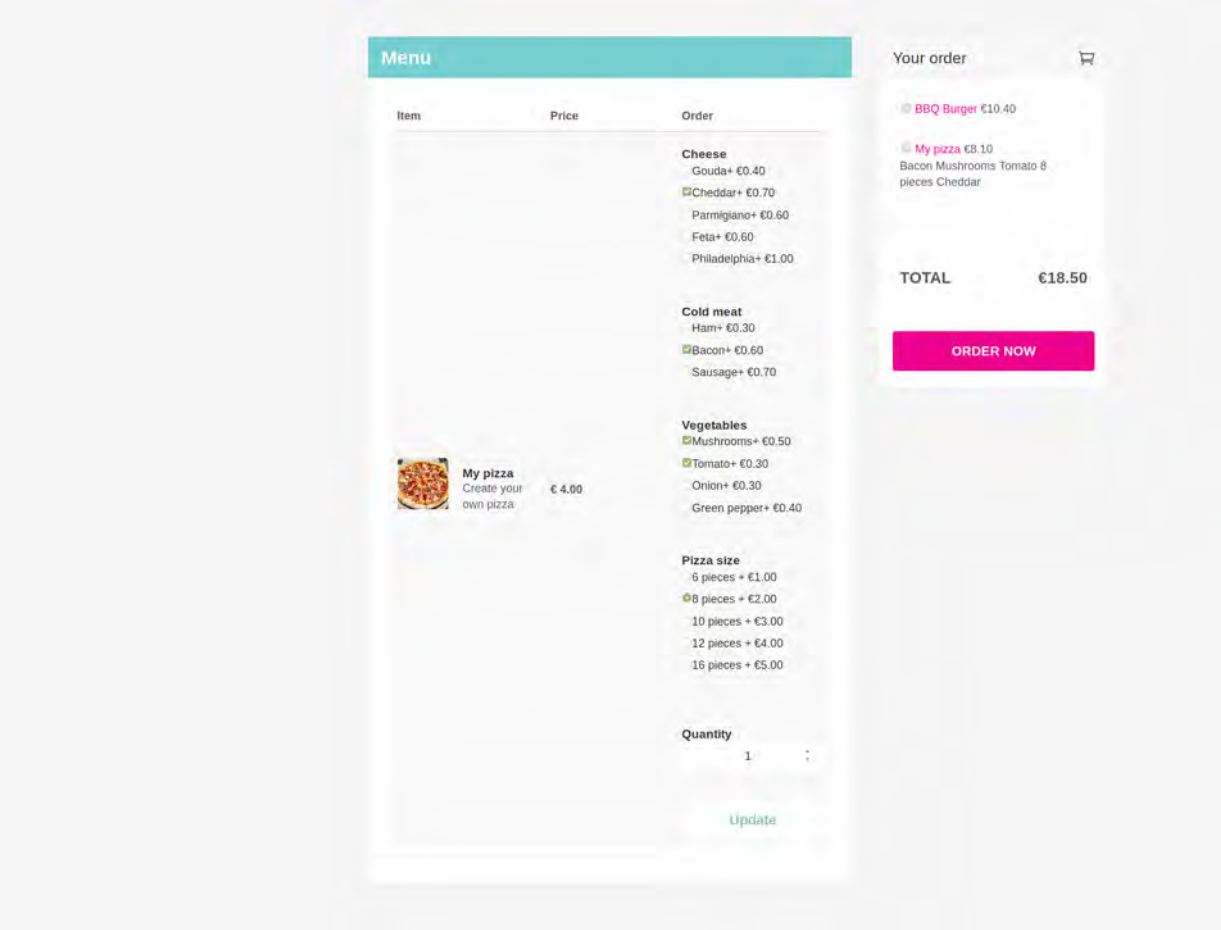
Σχήμα 6.6: Εφαρμογή φίλτρου για την εύρεση εστιατορίων



Σχήμα 6.7: Σελίδα εστιατορίου

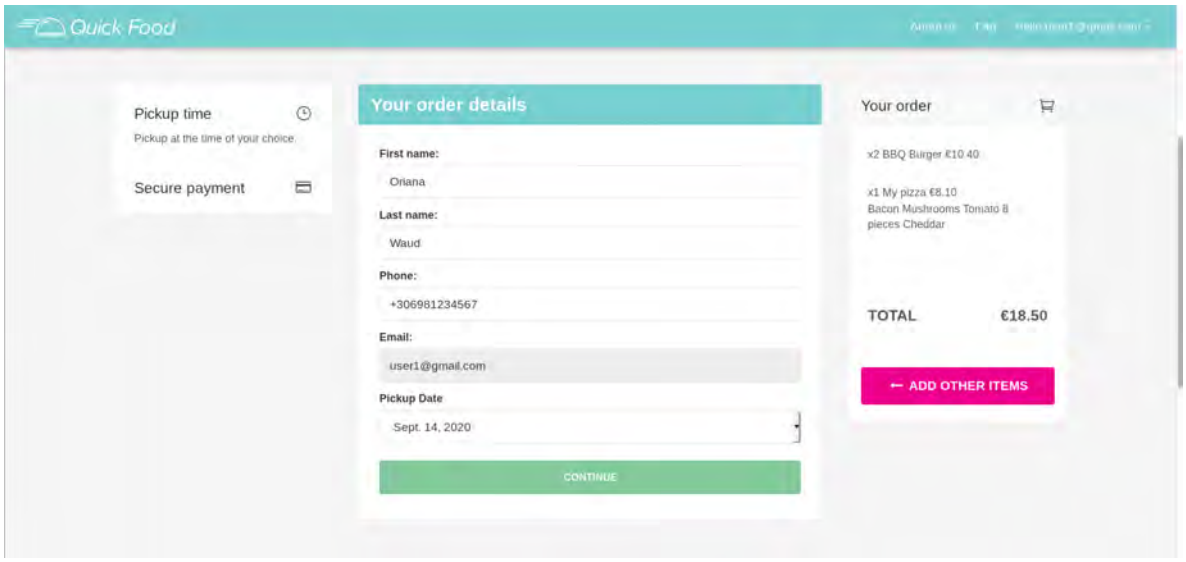
Στο σχήμα 6.7 εμφανίζεται το μενού του καταστήματος «Pizza Place». Ο χρήστης μπορεί να δει πληροφορίες για το κατάστημα πατώντας το κουμπί «Store Details» κάτω από το logo του. Αριστερά φαίνονται όλες οι κατηγορίες προϊόντων που προσφέρει το εστιατόριο, ενώ στο κέντρο της σελίδας για κάθε κατηγορία εμφανίζονται τα διαθέσιμα προϊόντα της. Για την προσθήκη ενός προϊόντος στο καλάθι, ο χρήστης πρέπει να πατήσει πάνω στο κουμπί «+» και αφού επιλέξει την ποσότητα και κάποια πρόσθετα υλικά και επιλογές -εάν αυτά υπάρχουν-, συνεχίζει πατώντας «Add to cart». Τα προϊόντα που έχουν προστεθεί στο καλάθι εμφανίζονται στη δεξιά πλευρά, ενώ ο χρήστης μπορεί να δει και το συνολικό ποσό προς πληρωμή.

Εάν ο χρήστης θέλει να διαγράψει ένα προϊόν από το καλάθι, αρκεί να πατήσει το κουμπί αριστερά από το όνομα του προϊόντος. Σε περίπτωση που θέλει να τροποποιήσει την ποσότητα ή τα υλικά που έχει επιλέξει μπορεί να πατήσει πάνω στο όνομα του προϊόντος και να μεταφερθεί στη σελίδα που φαίνεται στο σχήμα 6.8. Εκεί μπορεί να αλλάξει οτιδήποτε επιθυμεί και στη συνέχεια να πατήσει το κουμπί «UPDATE» ώστε να μεταφερθεί πίσω στη σελίδα με το μενού του καταστήματος. Όταν ο χρήστης είναι έτοιμος να παραγγείλει μπορεί να πατήσει το κουμπί «ORDER NOW».



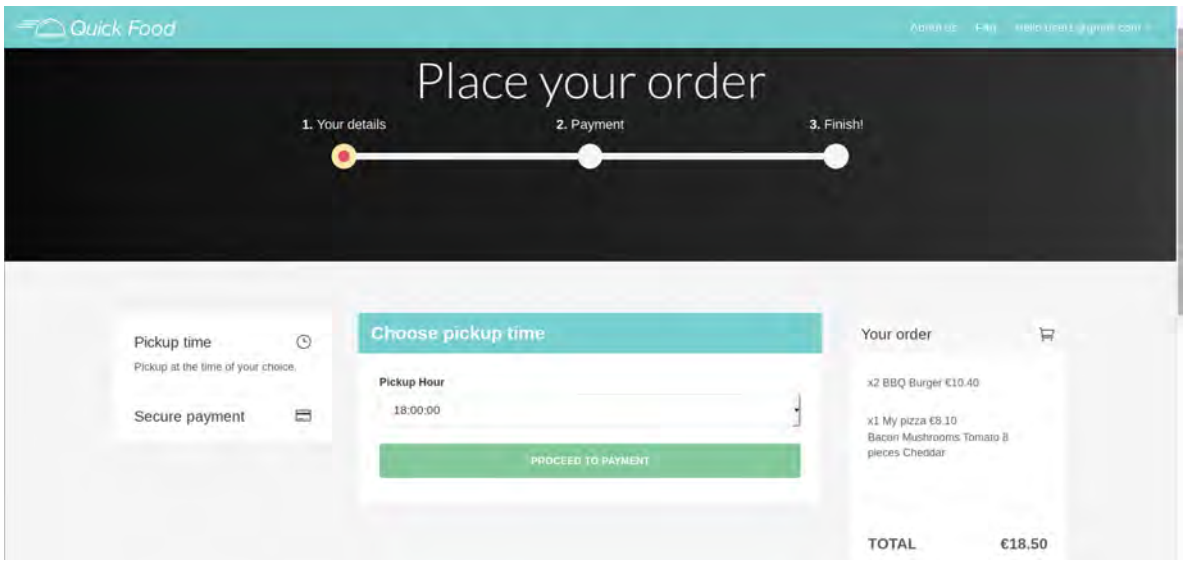
Σχήμα 6.8: Σελίδα επεξεργασίας προϊόντος του καλαθιού

Στη σελίδα του σχήματος 6.9, ο χρήστης εισάγει τα στοιχεία του όταν δεν είναι συνδεδεμένος, διαφορετικά συμπληρώνονται αυτόματα. Για να προχωρήσει στο επόμενο βήμα χρειάζεται η επιλογή της ημερομηνίας παραλαβής της παραγγελίας.



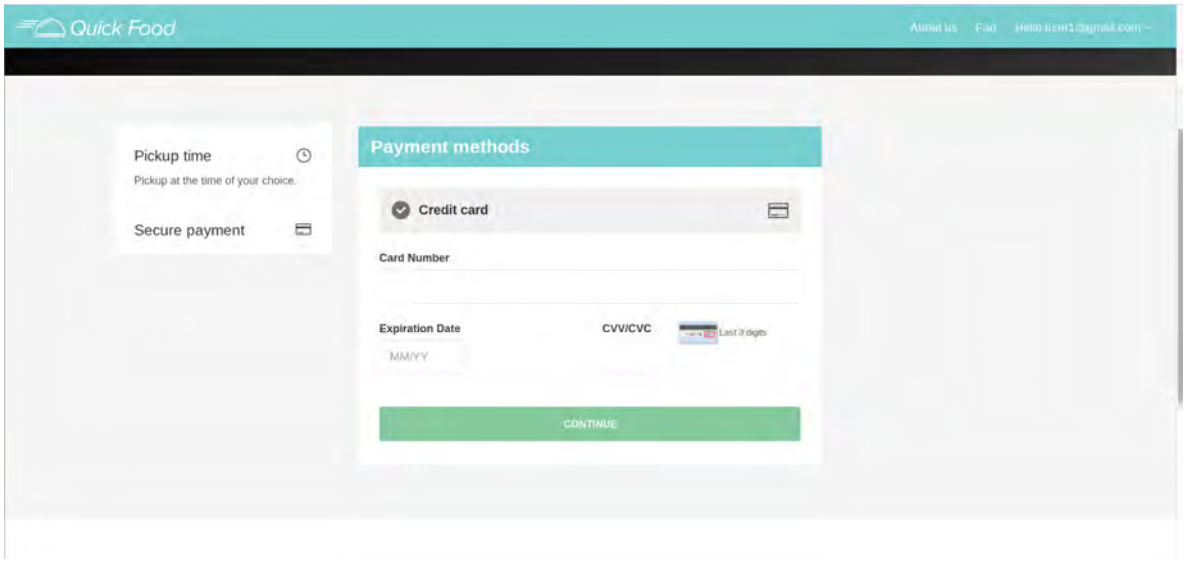
Σχήμα 6.9: Σελίδα επιλογής ημερομηνίας παραλαβής παραγγελίας

Αφού ο χρήστης πατήσει το κουμπί «CONTINUE», μεταφέρεται στη σελίδα επιλογής της ώρας παραλαβής, που φαίνεται στο σχήμα 6.10. Οι διαθέσιμες ώρες που εμφανίζονται στο χρήστη έχουν διαφορά μεταξύ τους ένα τέταρτο της ώρας και μετά την επιλογή μίας εξ αυτών, πατώντας το κουμπί «Proceed to payment» γίνεται μεταφορά στο επόμενο βήμα.



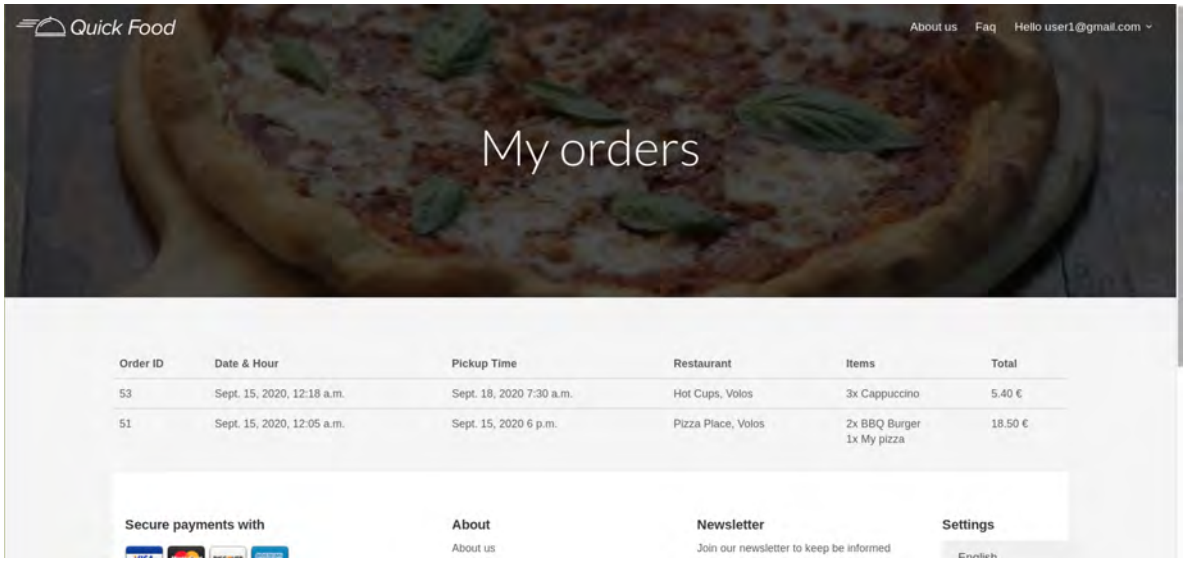
Σχήμα 6.10: Σελίδα επιλογής ώρας παραλαβής παραγγελίας

Στη σελίδα του σχήματος 6.11 ο χρήστης εισάγει τα στοιχεία της πιστωτικής ή χρεωστικής του κάρτας. Για να προχωρήσει στο επόμενο βήμα χρειάζεται το πάτημα του κουμπιού «CONTINUE».



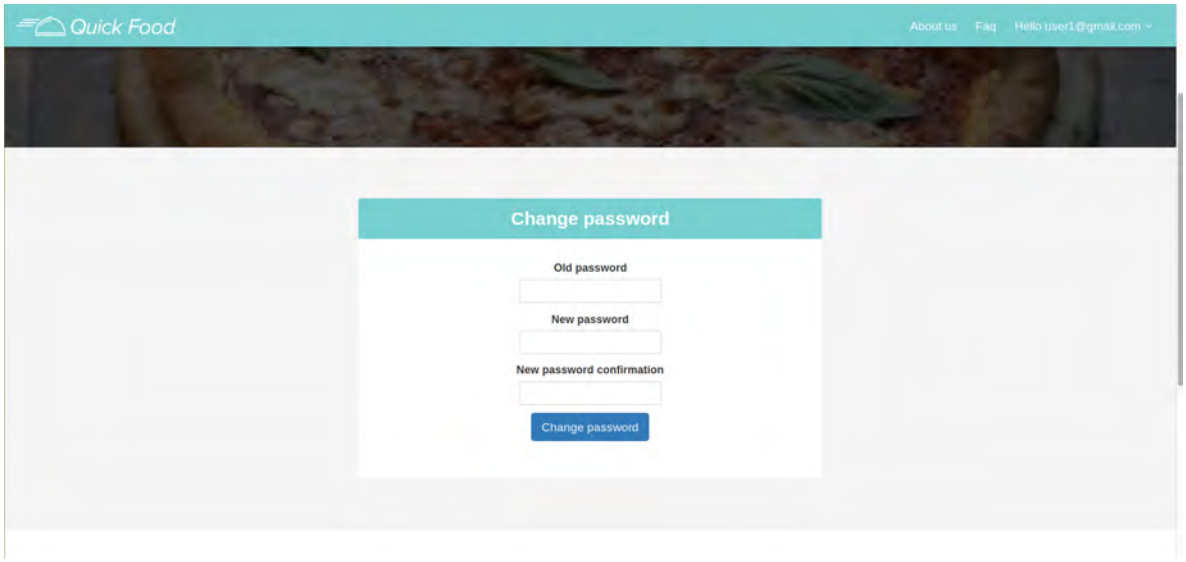
Σχήμα 6.11: Σελίδα πληρωμής

Αφού ολοκληρωθεί επιτυχώς η παραγγελία, το κατάστημα ενημερώνεται για αυτή με τρόπο θα αναλυθεί παρακάτω. Ο χρήστης μπορεί να δει τις παραγγελίες που έχει πραγματοποιήσει και που έχουν γίνει αποδεκτές από το κατάστημα πηγαίνοντας πάνω δεξιά στο email του και πατώντας «My orders». Το σχήμα 6.12 δείχνει τη σελίδα αυτή και τις πληροφορίες που εμφανίζονται.

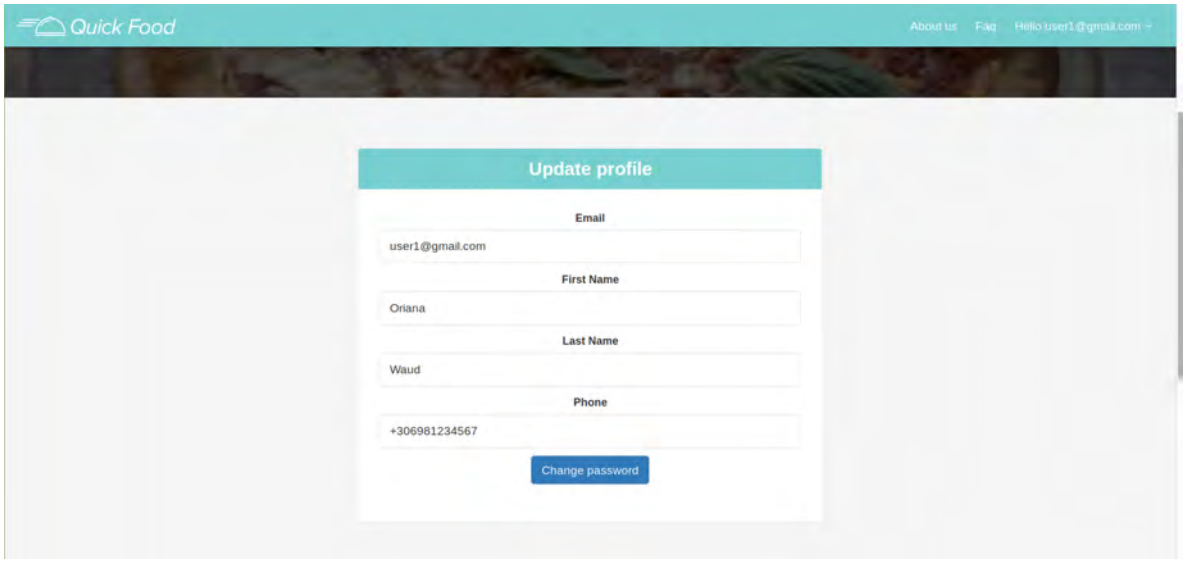


Σχήμα 6.12: Σελίδα προβολής πραγγελιών

Δύο ακόμα βασικές λειτουργίες που προσφέρονται στο χρήστη είναι η αλλαγή του κωδικού πρόσβασης και η επεξεργασία των στοιχείων του. Στο σχήμα 6.13 φαίνεται η σελίδα αλλαγής κωδικού, ενώ στο 6.14 φαίνεται πώς μπορεί να γίνει αλλαγή του email, του ονόματος, του επιθέτου ή και του τηλεφώνου του χρήστη.



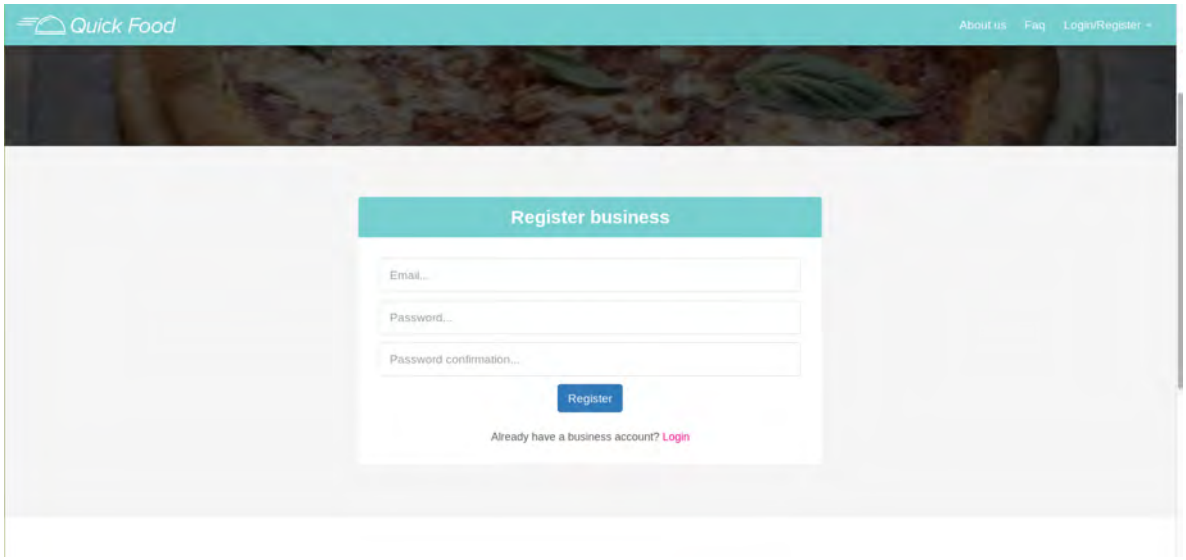
Σχήμα 6.13: Σελίδα αλλαγής κωδικού



Σχήμα 6.14: Σελίδα επεξεργασίας προφίλ

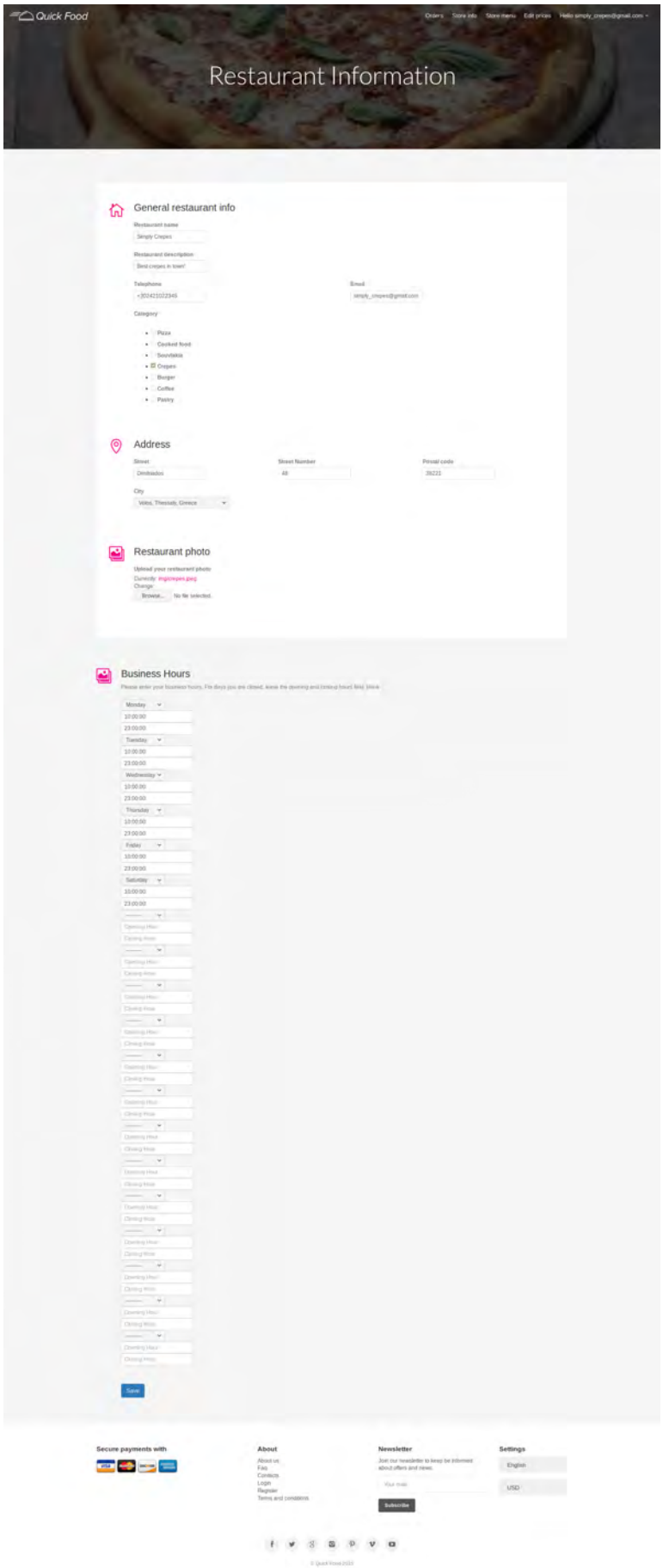
6.2.2 Σελίδες με πρόσβαση από τα καταστήματα

Για την εγγραφή ενός καταστήματος στην πλατφόρμα, όπως φαινεται και στο σχήμα 6.15, απαιτείται ένα email και ένας κωδικός πρόσβασης.



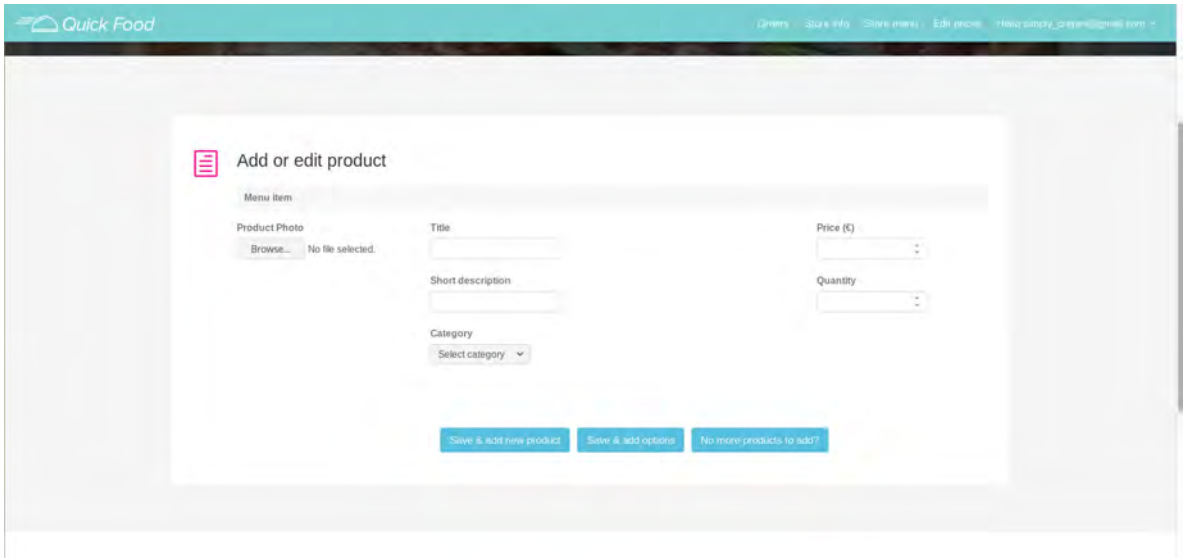
Σχήμα 6.15: Σελίδα εγγραφής καταστημάτων

Μετά την εγγραφή γίνεται ανακατεύθυνση στη σελίδα του σχήματος 6.16. Θα πρέπει να συμπληρωθούν τα πεδία της φόρμας που περιλαμβάνουν το όνομα του καταστήματος, μια προαιρετική περιγραφή, έναν αριθμό επικοινωνίας, το email που συμπληρώνεται αυτόματα, τη διεύθυνσή του, μια εικόνα, καθώς και τις κατηγορίες κουζίνας στις οποίες ανήκει. Τέλος, απαραίτητο είναι να εισαχθεί και το ωράριο λειτουργίας του καταστήματος. Για κάθε ημέρα της εβδομάδας είναι δυνατό να καταγραφούν δύο ωράρια. Για παράδειγμα, για τη Δευτέρα θα μπορούσε να εισαχθεί το ωράριο 08:00-14:00 και 16:00-22:00.



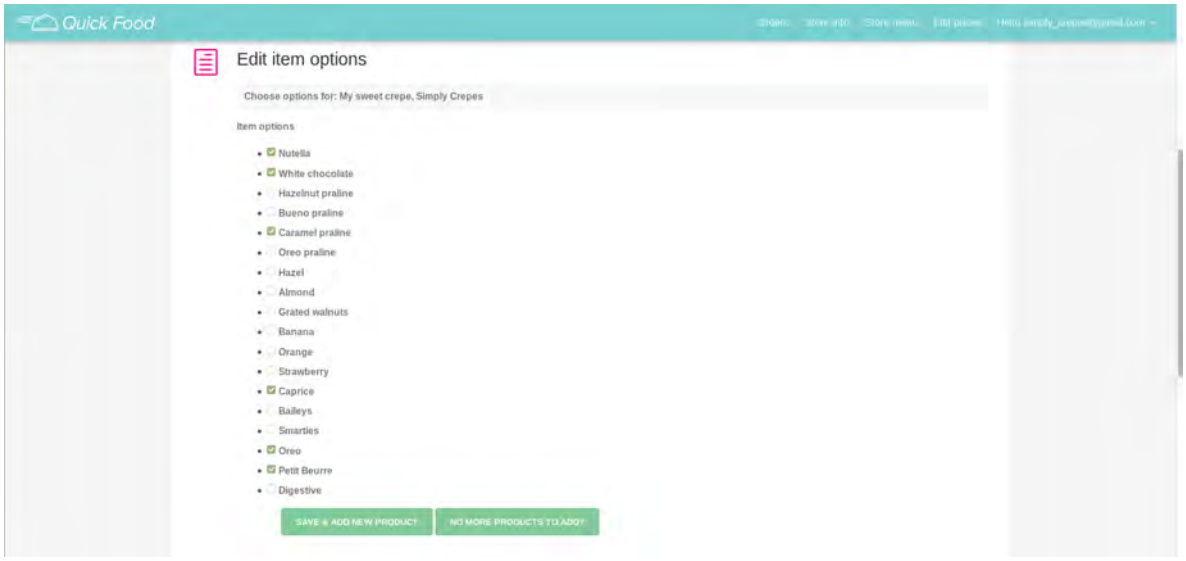
Σχήμα 6.16: Σελίδα εισαγωγής στοιχείων καταστήματος

Στη σελίδα εισαγωγής του μενού (σχήμα 6.17) εισάγονται κάποιες πληροφορίες σχετικά με το προϊόν και επιλέγεται η κατηγορία στην οποία ανήκει. Στη συνέχεια υπάρχει η επιλογή προσθήκης έξτρα υλικών, αποθήκευσης και προσθήκης νέου προϊόντος, ή αποθήκευσης και συνέχειας στο επόμενο βήμα.



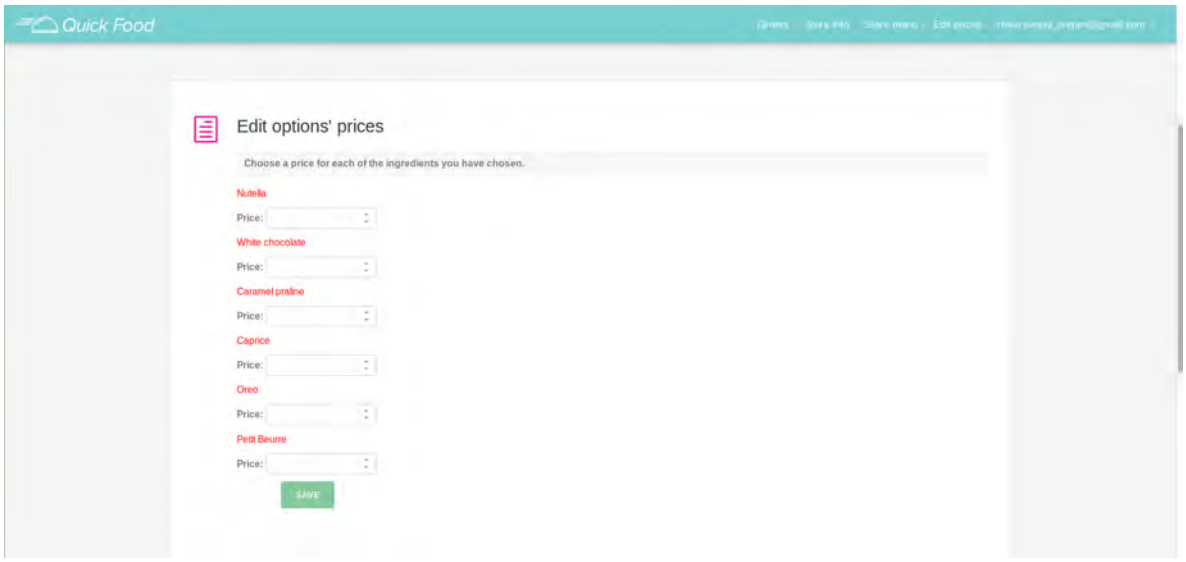
Σχήμα 6.17: Σελίδα εισαγωγής του μενού

Στην πρώτη περίπτωση, εμφανίζονται όλα τα υλικά που είναι διαθέσιμα στην κατηγορία προϊόντος που έχει επιλεχθεί. Στην εικόνα 6.18, το προϊόν στο οποίο προστίθεται η δυνατότητα επιλογής υλικών είναι η γλυκιά κρέπα. Αφού γίνει επιλογή των υλικών που θα είναι διαθέσιμα στο χρήστη, το κατάστημα μπορεί να επιλέξει ανάμεσα σε δύο κουμπιά για να συνεχίσει. Εάν επιλέξει το δεύτερο κουμπί θα μεταφερθεί στο επόμενο βήμα, αλλιώς θα γυρίσει στη σελίδα εισαγωγής νέου προϊόντος.



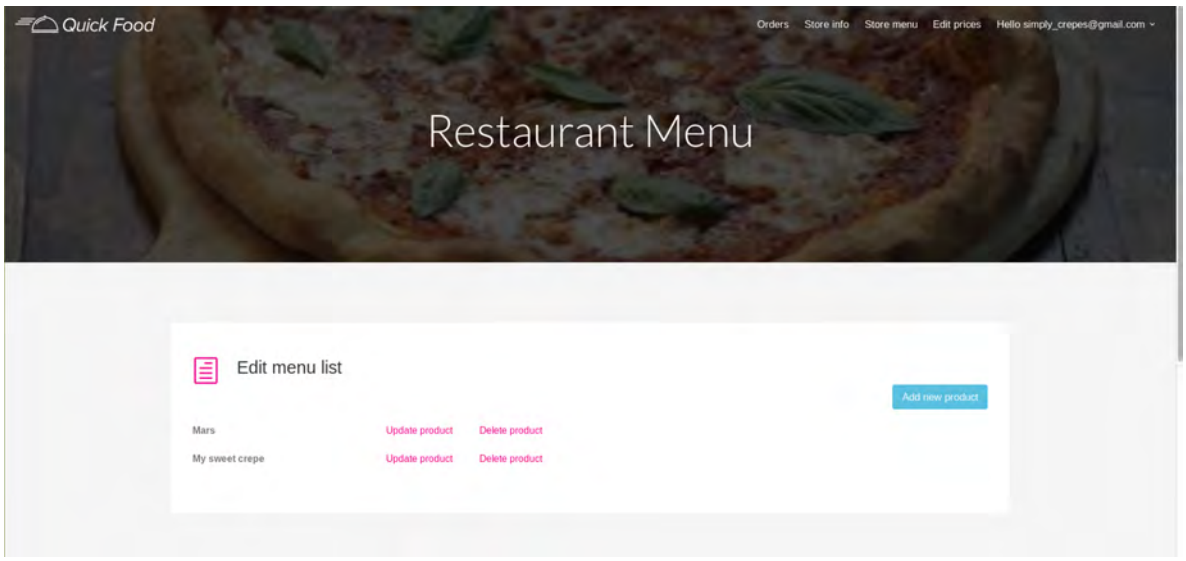
Σχήμα 6.18: Σελίδα εισαγωγής επιπλέον υλικών

Το τελευταίο βήμα για την ολοκλήρωση της δημιουργίας του μενού είναι η εισαγωγή της τιμής κάθε υλικού που μπορεί να προσφερθεί σαν πρόσθετο υλικό στα προϊόντα του καταστήματος.

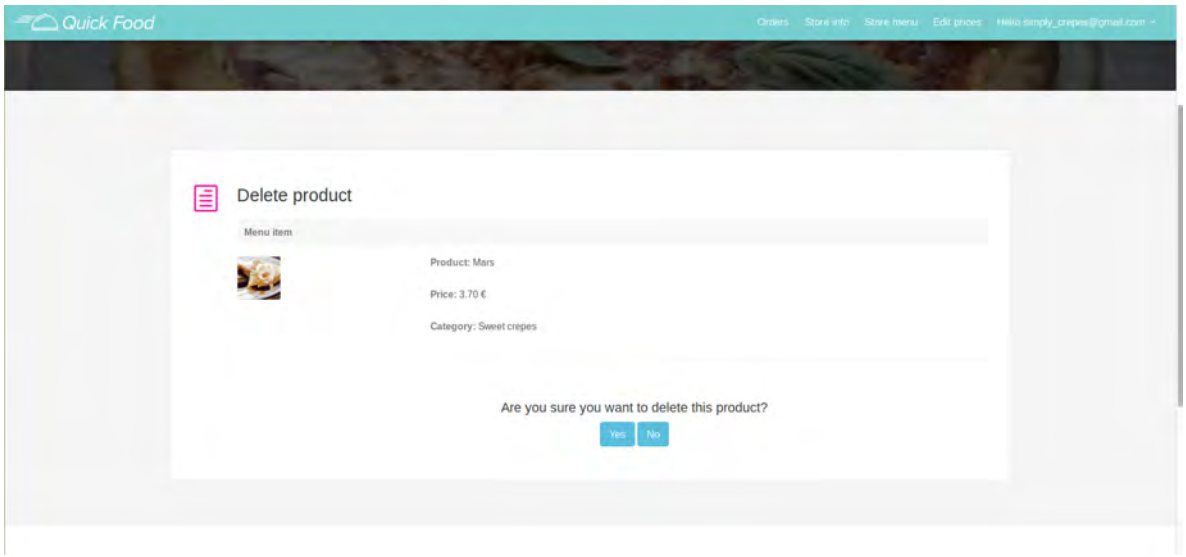


Σχήμα 6.19: Σελίδα επεξεργασίας των τιμών των επιπρόσθετων υλικών

Το κατάστημα μπορεί ανά πάσα στιγμή να αλλάξει τα στοιχεία του πατώντας το κουμπί «Store info». Για να τροποποιήσει το μενού χρειάζεται να πατήσει το «Store menu» για να ανακατευθυνθεί στη σελίδα του σχήματος 6.20, όπου μπορεί είτε να επεξεργαστεί το προϊόν είτε να το διαγράψει. Εάν πατήσει «Delete product», μεταφέρεται στη σελίδα του σχήματος 6.21, όπου πρέπει να επιβεβαιώσει ή όχι τη διαγραφή του προϊόντος. Για την επεξεργασία των τιμών των πρόσθετων υλικών χρειάζεται να επιλεγεί το «Edit prices».



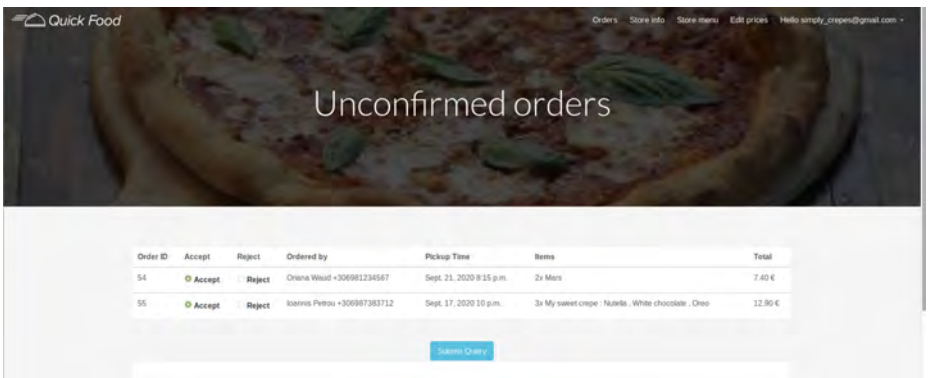
Σχήμα 6.20: Σελίδα επεξεργασίας του μενού



Σχήμα 6.21: Σελίδα διαγραφής προϊόντος

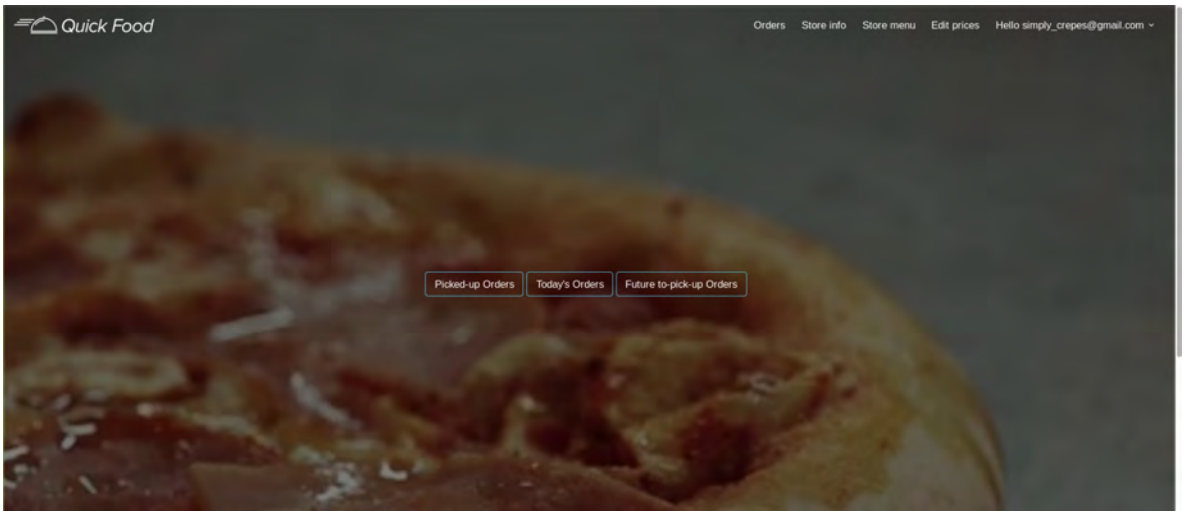
Στην αρχική σελίδα του καταστήματος (σχήμα 6.22) εμφανίζεται ένας πίνακας με όλες τις παραγγελίες του καταστήματος που ζητούν επιβεβαίωση. Όταν ένας χρήστης ολοκληρώσει την πληρωμή της παραγγελίας του, το κατάστημα ειδοποιείται για αυτή και μέχρι να την αποδεχτεί ή να την απορρίψει, αυτή συνεχίζει να εμφανίζεται στον πίνακα. Προϋπόθεση για την ενημέρωση του καταστήματος είναι να βρίσκεται συνδεδεμένο στην εφαρμογή, ενώ υπάρχει και ηχητικό εφέ που ακούγεται όσο η παραγγελία μένει αναπάντητη. Οι πληροφορίες της παραγγελίας που λαμβάνει το κατάστημα είναι ο κωδικός της, το άτομο που την παρήγγειλε (είτε είναι εγγεγραμμένος χρήστης είτε όχι), η ημερομηνία παραλαβής που έχει επιλέξει ο χρήστης, τα προϊόντα που περιλαμβάνει και το συνολικό ποσό προς πληρωμή. Το κατάστημα μπορεί να επιλέξει είτε «Accept» είτε «Reject», ώστε να αποδεχτεί ή όχι, αντίστοιχα, την παραγγελία. Και στις δύο περιπτώσεις ο χρήστης ειδοποιείται μέσω αντίστοιχου email που στέλνεται αυτόματα μόλις το κατάστημα απαντήσει.

Ο τρόπος με τον οποίο το κατάστημα ενημερώνεται αμέσως για μια παραγγελία είναι μέσω της jQuery. Ο πίνακας που βρίσκεται στην αρχική σελίδα της εφαρμογής ανανεώνεται κάθε πέντε δευτερόλεπτα και ενημερώνεται από το κατάλληλο μοντέλο του Django που είναι το «Order».

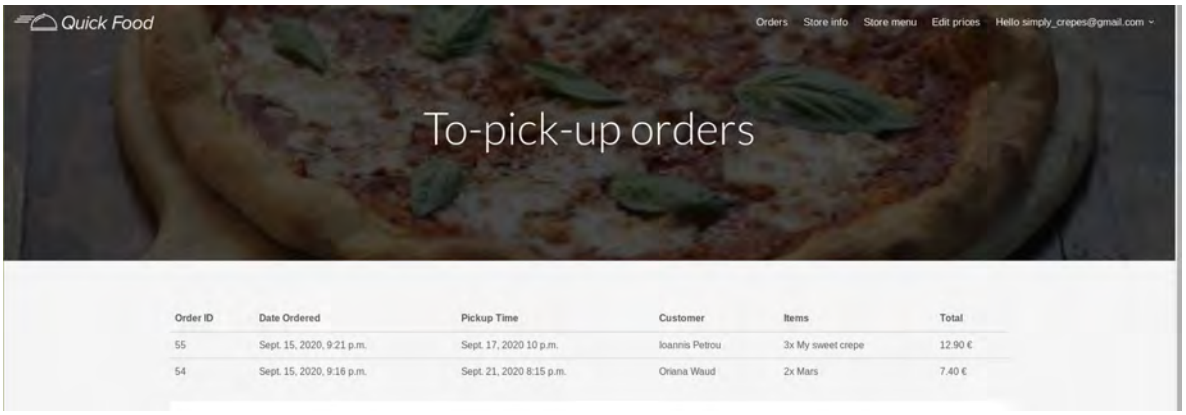


Σχήμα 6.22: Αρχική σελίδα καταστημάτων

Για να μπορέσει να ενημερωθεί για τις προγραμματισμένες παραγγελίες του το κατάστημα μπορεί να πατήσει το κουμπί «Orders». Αμέσως μετά θα μεταφερθεί στη σελίδα του σχήματος 6.23, όπου μπορεί να επιλέξει τις παραγγελίες ποιας ημέρας επιθυμεί να δει. Υπάρχουν τρεις επιλογές. Η πρώτη αφορά τις παραγγελίες που έχουν παραληφθεί τις προηγούμενες ημέρες από τους χρήστες. Η δεύτερη επιλογή εμφανίζει τις παραγγελίες που είναι προγραμματισμένες για παραλαβή την ίδια ημέρα, ενώ η τελευταία επιλογή αφορά όσες παραγγελίες θα παραληφθούν μελλοντικά. Στο σχήμα 6.24 φαίνεται η σελίδα που εμφανίζει τις παραγγελίες με μελλοντική παραλαβή.



Σχήμα 6.23: Σελίδα προβολής παραγγελιών



Σχήμα 6.24: Σελίδα προβολής παραγγελιών με μελλοντική ημερομηνία παραλαβής

Κεφάλαιο 7

Εγκατάσταση εφαρμογής

7.1 Εισαγωγή

Καθώς η ιστοσελίδα τρέχει σε τοπικό δίκτυο δεν μπορεί να έχει πρόσβαση κανείς, παρά μόνο όσοι βρίσκονται στο ίδιο δίκτυο. Εάν η εφαρμογή ήταν ανεβασμένη στο Διαδίκτυο, τότε για να τη χρησιμοποιήσει κάποιος δε θα χρειαζόταν να κάνει εγκατάσταση, αλλά μόνο αναζήτηση του URL της σε κάποιο φυλλομετρητή. Παρακάτω περιγράφονται τα βήματα εγκατάστασης της εφαρμογής σε έναν υπολογιστή που δεν ανήκει στο τοπικό δίκτυο. Οι εικόνες 7.1 και 7.2 παρουσιάζουν ακριβώς τα βήματα.

7.2 Βήματα εγκατάστασης

Βήμα 1: Άνοιγμα του terminal και δημιουργία ενός φακέλου σε οποιοδήποτε σημείο του υπολογιστή με την εντολή «mkdir MyFolder». Ο φάκελος μπορεί να ονομαστεί με οποιοδήποτε όνομα, αλλά για το παράδειγμα χρησιμοποιείται το «MyFolder».

Βήμα 2: Εκτέλεση της εντολής «cd MyFolder» για είσοδο στον φάκελο αυτό.

Βήμα 3: Εκτέλεση της εντολής «virtualenv -p python3 .» για δημιουργία εικονικού περιβάλλοντος.

Βήμα 4: Εκτέλεση της εντολής «source bin/activate» για ενεργοποίηση του εικονικού περιβάλλοντος.

Βήμα 5: Δημιουργία φακέλου με όνομα «src» μέσα στο φάκελο «MyFolder» με την εντολή «mkdir src».

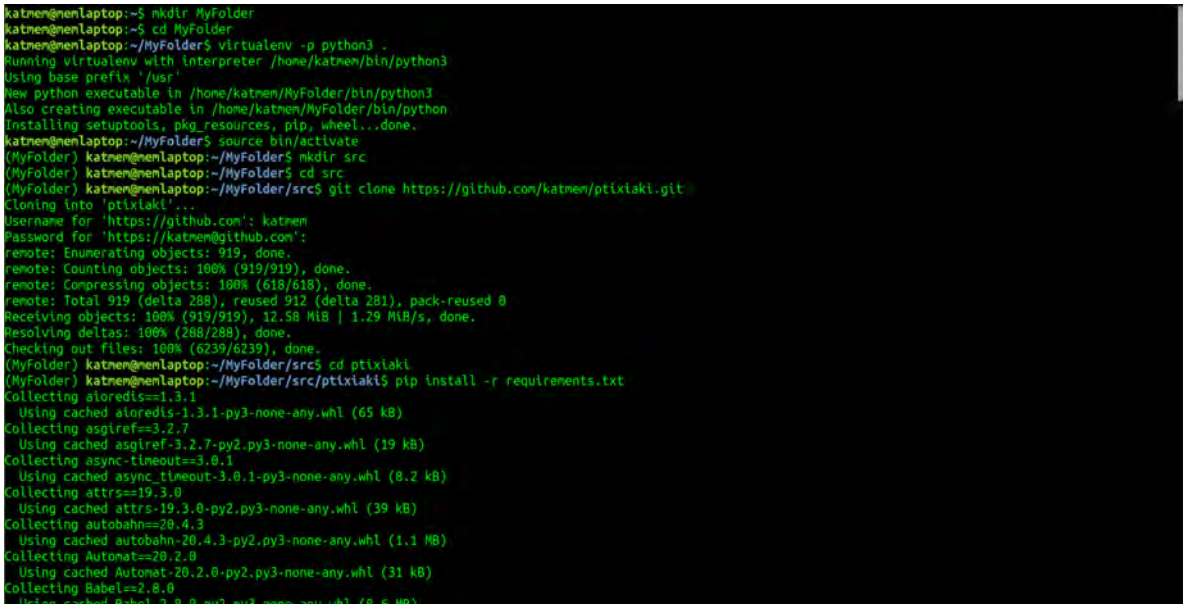
Βήμα 6: Εκτέλεση της εντολής «cd src» για είσοδο στον φάκελο αυτό.

Βήμα 7: Εκτέλεση της εντολής «git clone https://github.com/katmem/thesis.git» για αντιγραφή του project μέσα στο φάκελο «src».

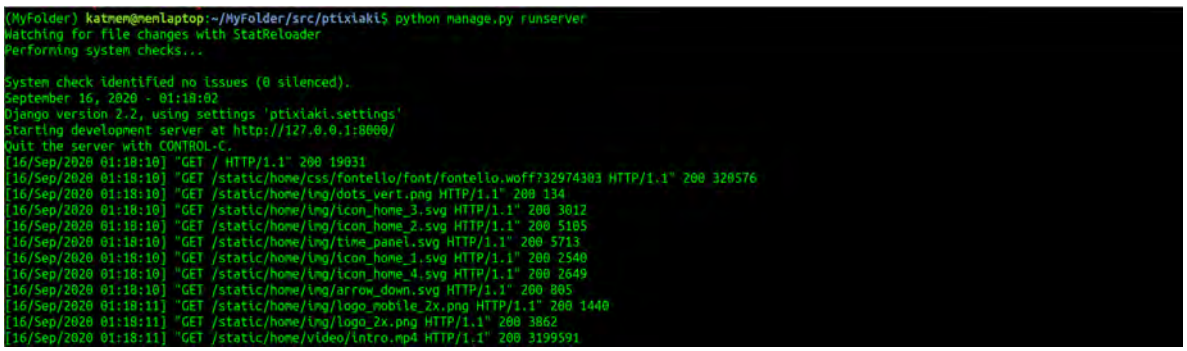
Βήμα 8: Εκτέλεση της εντολής «cd ptixiaki» και στη συνέχεια της «pip install -r requirements.txt».

Βήμα 9: Πρόσβαση στο φάκελο με path «MyFolder/lib/python3.6/site-packages/pymongo», άνοιγμα του αρχείου «mongo_client.py» και αντικατάσταση (στη γραμμή 90) του «HOST = ”localhost”» με το «HOST = ”mongodb+srv://katmem:katerina@cluster0-dmfgz.mongodb.net/test?retryWrites=truew=majority”» που είναι το string σύνδεσης στη βάση.

Βήμα 10: Εκτέλεση της εντολής «python manage.py runserver» και μετάβαση στην ιστοσελίδα «http://127.0.0.1:8000/».



Σχήμα 7.1: Βήματα εγκατάστασης



Σχήμα 7.2: Εκτέλεση εφαρμογής

Κεφάλαιο 8

Επίλογος

Στο κεφάλαιο αυτό θα γίνει μια σύνοψη των αποτελεσμάτων της διπλωματικής και θα αναφερθούν πιθανές επεκτάσεις που θα μπορούσαν να γίνουν στο μέλλον.

8.1 Σύνοψη και συμπεράσματα

Η ανάπτυξης μιας web εφαρμογής είναι σίγουρα μια πολύπλοκη διαδικασία που απαιτεί πολλή μελέτη και καλό σχεδιασμό. Μέσω της συγκεκριμένης εργασίας έγινε κατανοητή η διαδικασία που απαιτείται για να μπορέσει κανείς να κατασκευάσει μια εφαρμογή με τη χρήση του Django. Το Django είναι ένα πολύ χρήσιμο εργαλείο που προσφέρει πολλά έτοιμα στοιχεία που διαφορετικά θα χρειαζόταν να τα προγραμματήσει κανείς. Όσον αφορά τη βάση δεδομένων, η MongoDB ενώ προσφέρει καλή απόδοση, δεν είναι ιδανική για χρήση με το Django. Το Django ORM, δηλαδή η τεχνική με την οποία τα μοντέλα μετατρέπονται σε πίνακες και οι εντολές της Python σε queries, είναι κατασκευασμένη για τις σχεσιακές βάσεις δεδομένων. Έτσι η σύνδεση με τις NoSQL βάσεις γίνεται δυσκολότερη και πολλές φορές η εγκατάσταση πακέτων και η εκτέλεση κάποιων queries προκαλούν errors.

Η εφαρμογή αυτή σίγουρα πετυχαίνει το σκοπό δημιουργίας της που δεν είναι άλλος από τη διευκόλυνση μιας μερίδας ατόμων. Η έλλειψη χρόνου των εργαζομένων και φοιτητών είναι γεγονός, επομένως η επιλογή της ώρας παραλαβής του φαγητού τους βοηθά στην εξοικονόμηση χρόνου και τον καλύτερο προγραμματισμό τόσο αυτών όσο και των καταστημάτων εστίασης.

8.2 Μελλοντικές επεκτάσεις

Υπάρχουν αρκετές λειτουργίες που θα μπορούσαν να προστεθούν στην εφαρμογή μελλοντικά. Αρχικά, η επιλογή παράδοσης της παραγγελίας στο σπίτι σε επιλεγμένη από το χρήστη ημερομηνία και ώρα θα ήταν πολύ χρήσιμη, καθώς θα διευκόλυνε περισσότερους ανθρώπους και θα αύξανε τον αριθμό παραγγελιών των καταστημάτων. Είναι κάτι που φυσικά υπάρχει ήδη στην αγορά, όμως προσφέρεται από ελάχιστες πλατφόρμες καταστημάτων εστίασης.

Εκτός από τη δυνατότητα παραλαβής της παραγγελίας από το κατάστημα, θα ήταν ενδιαφέρουσα και η δυνατότητα κράτησης τραπεζιού για κατανάλωση της παραγγελίας στο εστιατόριο. Η επιλογή αυτή θα ήταν ιδιαίτερα χρήσιμη για μεγάλες ομάδες ατόμων για τις οποίες η διαδικασία παραγγελίας στο/στη σερβιτόρο/α και η ετοιμασία των γευμάτων είναι χρονοβόρα. Με την προσθήκη της νέας επιλογής όμως, το κατάστημα θα ξέρει ήδη το πρόγραμμα των κρατήσεων και την παραγγελία τους, επομένως η εξυπηρέτηση των πελατών θα είναι άμεση.

Ακόμα μια επέκταση που θα μπορούσε να γίνει είναι να προστεθούν και άλλα είδη επιχειρήσεων εκτός από τα καταστήματα εστίασης, όπως σούπερ μάρκετ, μανάβικα, φαρμακεία, και άλλα που προσφέρουν παρόμοιες υπηρεσίες.

Τέλος, η ανάπτυξη της εφαρμογής για Android και iOS συστήματα είναι από τις σημαντικότερες μελλοντικές επεκτάσεις που χρειάζεται να γίνουν, προκειμένου να αυξηθούν οι χρήστες της και να είναι πιο εύκολη η καταχώρηση μιας παραγγελίας.

Βιβλιογραφία

[1] Django routing. https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Home_page.

[2] Mongodb atlas. https://www.mongodb.com/cloud/atlas.

[3] Varsha Chavan, Priya Jadhav, Snehal Korade, and Priyanka Teli. Implementing customizable online food ordering system using web based application. *IJISET - International Journal of Innovative Science, Engineering Technology*, 2:1–6, April 2015.

[4] Hung Viet Nguyen. End-to-end e-commerce web application, a modern approach using mern stack. Bachelor's Thesis, Metropolia University of Applied Sciences, May 2020.

[5] Building a food ordering system with mean stack. https://steemit.com/utopian-io/@sirfreeman/building-a-food-ordering-system-with-mean-stack-mongo-express-angular-and-nodejs.

[6] Food-ordering-system. https://github.com/asif536/Food-Ordering-System.

[7] Python. https://www.python.org/doc/.

[8] Javascript. https://www.javascript.com/.

[9] jquery. https://jquery.com/.

[10] Template. https://themeforest.net/item/quickfood-delivery-or-takeaway-food-template/13958100.

[11] Cornelia Győrödi, Robert Gyorodi, Andrada Olah, and George Pecherle. A comparative study: Mongodb vs. mysql. June 2015.

[12] Ramakrishnan and Gehrke. *Database Management Systems*. McGraw-Hill, 1996.

[13] Yishan Li and Sathiamoorthy Manoharan. A performance comparison of sql and nosql databases. August 2013.

[14] Django. `https://www.djangoproject.com/`.

[15] django-cities-light. `https://pypi.org/project/django-cities-light/3.0.0/`.

[16] Geonames. `https://www.geonames.org/`.

[17] django-autocomplete-light. `https://django-autocomplete-light.readthedocs.io/en/master/`.

[18] django-credit-cards. `https://pypi.org/project/django-credit-cards/`.