ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Πανεπιστήμιο Θεσσαλίας

ΣΥΣΤΗΜΑ ΑΝΑΛΥΣΗΣ ΔΕΔΟΜΕΝΩΝ ΠΑΡΑΓΓΕΛΙΟΛΗΨΙΩΝ ΚΑΤΑΣΤΗΜΑΤΩΝ ΕΣΤΙΑΣΗΣ

A SYSTEM FOR THE ANALYSIS OF ORDERING DATA OF FOOD AND BEVERAGE SERVICES

Διπλωματική Εργασία

Χαρίσης Πέτρος

Επιβλέπων Καθηγητής: Βασιλακόπουλος Μιχαήλ

Αναπληρωτής Καθηγητής Π.Θ.

Βόλος, 2018

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Πανεπιστήμιο Θεσσαλίας

Διπλωματική Εργασία με τίτλο:

ΣΥΣΤΗΜΑ ΑΝΑΛΥΣΗΣ ΔΕΔΟΜΕΝΩΝ ΠΑΡΑΓΓΕΛΙΟΛΗΨΙΩΝ ΚΑΤΑΣΤΗΜΑΤΩΝ ΕΣΤΙΑΣΗΣ

A SYSTEM FOR THE ANALYSIS OF ORDERING DATA OF FOOD AND BEVERAGE SERVICES

Χαρίσης Πέτρος

Επιβλέπων Α’

Βασιλακόπουλος Μιχαήλ

Αναπληρωτής Καθηγητής

Επιβλέπων Β’

Τσαλαπάτα Χαρίκλεια

Ε.ΔΙ.Π.

Βόλος, 2018

Ευχαριστίες

Ευχαριστώ τον επιβλέποντα καθηγητή κ. Βασιλακόπουλο Μιχαήλ για την πολύτιμη βοήθεια του, το χρόνο που διέθεσε και τις συμβουλές του καθ’ όλη την διάρκεια της παρούσας διλπωματικής.

Περίληψη

Η διαχείριση ενός κέντρου διασκέδασης δεν είναι εύκολη υπόθεση. Υπάρχουν πολλά πράγματα που πρέπει να έχει ο διαχειριστής ταυτόχρονα στο μυαλό του, μερικά από αυτά είναι το μενού που προσφέρει, το προσωπικό που διαθέτει αλλά και τα εργαλεία που θα χρησιμοποιηθούν από αυτό για τη μέγιστη απόδοση. Πριν από μερικά χρόνια το μπλοκάκι και το στυλό του σερβιτόρου αντικαταστάθηκε από «mobile» ηλεκτρονικές συσκευές, τα PDA. Σήμερα ο κάθε υπάλληλος τέτοιου καταστήματος μπορεί να χρησιμοποιήσει τη συσκευή που έχει ούτως ή άλλως στην τσέπη του, το κινητό του, για παραγγελιοληψία, επεξεργασία μενού και πολλά άλλα.

Σκοπός της παρούσας διπλωματικής είναι η δημιουργία ενός εργαλείου-υπηρεσίας σε συνδυασμό με εξαγωγή πληροφορίας και συμπερασμάτων για τα είδη, για το προσωπικό και τις μέρες, μιας δηλαδή δυναμικής διαδικτυακής εφαρμογής για παρόμοια μαγαζιά με τη χρήση της πλατφόρμας Meteor και της NoSQL βάσης δεδομένων MongoDB, σε συνδυασμό με γλώσσες προγραμματισμού Javascript, HTML και CSS και εργαλεία όπως react, DOM και διάφορα πακέτα του Meteor.

Η εφαρμογή είναι διαθέσιμη και για ηλεκτρονικό υπολογιστή αλλά και κινητή συσκευή. Ακολουθεί αναλυτική επεξήγηση του προβλήματος, των εργαλείων που χρησιμοποιήθηκαν και βήμα προς βήμα η ανάπτυξη της εφαρμογής και των λειτουργιών της.

Abstract

The management of business establishments like restaurants and coffee shops is not an easy task. There are a lot of things a manager has to overcome, some of them are the menu creation, the proper staff management and of course the proper tools for the job for maximum performance. It was not long ago when the only tool a waiter had to take orders, was a paper and a pen. A lot has changed since then as lately almost all of them use a PDA (electronic digital assistant) and now they have the privilege to use their own mobile phones for order taking, menu configuration and even manage their whole bysiness.

The purpose of the current diploma thesis is to create such a tool-service in conjunction with extracting information and conclusions about the menu items, staff and days, a dunamic web application for similar stores, using Meteor platform and NoSQL database, combined using Javascript, HTML and CSS programming while tools like React, DOM and several Meteor packages.

The application is available both on a PC and a mobile device by using only the web browser. Here is a detailed explanation of the difficulties and the possible facilitations the application has to offer as well as descriptions of the tools used and step-by-step the development of the application and its functions.

Πίνακας Περιεχομένων

Λίστα πινάκων και εικόνων

---

1. Εισαγωγή

1.1. Διαχείριση καταστήματος

Οι ιδιοκτήτες των εστιατορίων αντιμετωπίζουν καθημερινά προβλήματα και δυσκολίες στη διαχείριση της λειτουργίας των επιχειρήσεών τους. Η διαχείριση ενός κέντρου διασκέδασης είναι αναμφισβήτητα ένα από τα πιο αποθαρρυντικά καθήκοντα που μπορεί να αναλάβει ένας επιχειρηματίας. Η διαχείριση συνεπάγεται πολλές διαδικασίες, οι οποίες, εάν γίνουν σωστά, θα διασφαλίσουν την ομαλή ροή της επιχείρησης και ακόμη και τα αυξημένα κέρδη.

Ωστόσο, η όλη σκληρή δουλειά δεν αρκεί όταν πρόκειται για διαχείριση εστιατορίων. Πολλά εστιατόρια καταλήγουν να κερδίζουν σχεδόν το ίδιο παρά τις καλύτερες προσπάθειες του προσωπικού και των διαχειριστών. Αυτό ισχύει ιδιαίτερα για μικρά εστιατόρια. Πολλές οικογενειακές επιχειρήσεις εξακολουθούν να αγωνίζονται για να συμβαδίζουν με τις καθημερινές απαιτήσεις της υπηρεσίας, όπως ο σχεδιασμός μενού, η λήψη παραγγελιών και η ενημέρωση αποθέματος.

Ευτυχώς, αναπτύσσονται πολλές τεχνολογίες για να βοηθήσουν τους ιδιοκτήτες των εστιατορίων στις καθημερινές τους επιχειρήσεις. Αυτές οι τεχνολογίες έχουν τη μορφή συσκευών και λογισμικού (web application) που έχουν σχεδιαστεί για να κάνουν κάπως καλύτερη τη ζωή των ιδιοκτητών εστιατορίων.

Σκοπός αυτών των εφαρμογών είναι η αυτοματοποίηση των επιχειρηματικών διαδικασιών που σχετίζονται με τη λειτουργία τέτοιων οργανισμών. Ωστόσο, πολλές διαδικασίες απαιτούν ειδικό λογισμικό, καθιστώντας την αυτοματοποίηση μια ακριβή πρόταση για όσους θέλουν να βελτιώσουν όλες τις πτυχές της επιχείρησής τους.

Χάρη στην ανάπτυξη εφαρμογών διαχείρισης εστιατορίων, όλες αυτές οι διαδικασίες μπορούν πλέον να εποπτεύονται χρησιμοποιώντας μια ενιαία λύση. Στην ιδανική περίπτωση, τα βασικά χαρακτηριστικά θα πρέπει να περιλαμβάνουν λειτουργίες όπως η χρέωση, το σύστημα παραγγελιοληψίας, το απόθεμα, η αναφορά και η ανάλυση. Η αυτοματοποίηση αυτών των εργασιών βοηθά στη μείωση των επιβαρύνσεων της διαχείρισης, βελτιώνοντας τις πωλήσεις και εμπλουτίζοντας την εμπειρία του πελάτη [1].

1.2. Τι είναι μια Εφαρμογή Ιστού;

Ένα web application είναι ένα πρόγραμμα που χρησιμοποιεί τα προγράμματα περιήγησης καθώς και τις τεχνολογίες ιστού για να την εκτέλεση εργασιών μέσω του διαδικτύου.

Εκατομμύρια επιχειρήσεις χρησιμοποιούν το διαδίκτυο ως οικονομικά αποδοτικό κανάλι επικοινωνίας. Τους επιτρέπει να ανταλλάσσουν πληροφορίες με το αγοραστικό κοινό τους και να πραγματοποιούν γρήγορες και ασφαλείς συναλλαγές. Ωστόσο, μια αποτελεσματική συναλλαγή είναι δυνατή μόνο όταν η επιχείρηση είναι σε θέση να καταγράψει και να αποθηκεύσει όλα τα απαραίτητα δεδομένα και να έχει ένα μέσο για την επεξεργασία αυτών των πληροφοριών και την παρουσίαση των αποτελεσμάτων στο χρήστη.

Οι εφαρμογές Web χρησιμοποιούν ένα συνδυασμό scripts από την πλευρά του διακομιστή (PHP και ASP) για να διαχειριστούν την αποθήκευση και την ανάκτηση των πληροφοριών, και δέσμες ενεργειών από την πλευρά του πελάτη (JavaScript και HTML) για την παρουσίαση πληροφοριών στους χρήστες. Αυτό επιτρέπει στους χρήστες να αλληλεπιδρούν με την εταιρεία χρησιμοποιώντας ηλεκτρονικές φόρμες, συστήματα διαχείρισης περιεχομένου, κάρτες αγορών και άλλα. Επιπλέον, οι εφαρμογές επιτρέπουν στους υπαλλήλους να δημιουργούν έγγραφα, να μοιράζονται πληροφορίες, να συνεργάζονται σε έργα και να εργάζονται σε κοινά έγγραφα ανεξάρτητα από την τοποθεσία ή τη συσκευή [2].

1.3. Πώς δουλεύει μια Εφαρμογή Ιστού

Οι εφαρμογές Web κωδικοποιούνται συνήθως σε γλώσσα που υποστηρίζεται από προγράμματα περιήγησης, όπως JavaScript και HTML, καθώς αυτές οι γλώσσες βασίζονται στο πρόγραμμα περιήγησης για να καταστήσουν εκτελέσιμο το πρόγραμμα. Ορισμένες από τις εφαρμογές είναι δυναμικές, απαιτώντας επεξεργασία από την πλευρά του διακομιστή ενώ άλλες είναι εντελώς στατικές χωρίς να χρειάζονται επεξεργασία που απαιτείται στο διακομιστή.

Η εφαρμογή Ιστού απαιτεί έναν διακομιστή ιστού να διαχειρίζεται αιτήματα από τον πελάτη και για την εκτέλεση των ζητούμενων εργασιών και, ορισμένες φορές, μια βάση δεδομένων για την αποθήκευση των πληροφοριών. Η τεχνολογία διακομιστή εφαρμογών κυμαίνεται από ASP.NET, ASP και ColdFusion έως PHP και JSP.

Ακολουθεί μια τυπική ροή εφαρμογών ιστού:

- Ο χρήστης ενεργοποιεί ένα αίτημα στο διακομιστή ιστού μέσω του διαδικτύου, είτε μέσω ενός προγράμματος περιήγησης ιστού, είτε μέσω της διεπαφής χρήστη της εφαρμογής
- Ο διακομιστής Web διαβιβάζει αυτό το αίτημα στον κατάλληλο διακομιστή εφαρμογών ιστού που εκτελεί την απαιτούμενη εργασία - όπως ερώτηση στη βάση δεδομένων ή επεξεργασία δεδομένων - στη συνέχεια, δημιουργεί τα αποτελέσματα των ζητούμενων δεδομένων
- Ο διακομιστής Web ανταποκρίνεται πίσω στον πελάτη με τις ζητούμενες πληροφορίες που εμφανίζονται στην οθόνη του χρήστη

Πλεονεκτήματα μιας εφαρμογής ιστού:

- Οι εφαρμογές ιστού εκτελούνται σε πολλαπλές πλατφόρμες ανεξάρτητα από λειτουργικό σύστημα ή συσκευή, εφόσον το πρόγραμμα περιήγησης είναι συμβατό
- Όλοι οι χρήστες έχουν πρόσβαση στην ίδια έκδοση, εξαλείφοντας τυχόν προβλήματα συμβατότητας
- Δεν είναι εγκατεστημένα στο σκληρό δίσκο, εξαλείφοντας έτσι τους περιορισμούς χώρου
- Μειώνουν το κόστος τόσο για την επιχείρηση όσο και για τον τελικό χρήστη, καθώς απαιτείται λιγότερη υποστήριξη και συντήρηση από την επιχείρηση και χαμηλότερες απαιτήσεις για τον υπολογιστή του τελικού χρήστη

Συμπέρασμα

Η αυξημένη χρήση του διαδικτύου μεταξύ των εταιρειών και των ατόμων έχει επηρεάσει τον τρόπο λειτουργίας των επιχειρήσεων. Αυτό οδήγησε στην ευρεία υιοθέτηση των εφαρμογών ιστού, καθώς οι εταιρείες μετατοπίζονται από παραδοσιακά μοντέλα σε μοντέλα που βασίζονται σε cloud και σε δίκτυα. Οι εφαρμογές ιστού δίνουν στις επιχειρήσεις τη δυνατότητα να αυξάνουν την αποδοτικότητα τους και να μειώνουν το κόστος [2].

1.4. Υπηρεσία διαχείρισης χώρων μαζικής εστίασης

Στη συγκεκριμένη περίπτωση, η ανταλλαγή πληροφοριών δε γίνεται ανάμεσα στην εταιρεία και στους πελάτες της, αλλά μεταξύ του προσωπικού και των διαχειριστών του εκάστοτε κέντρου διασκέδασης. Καταργείται η χειρόγραφη λήψη της παραγγελίας με αποτέλεσμα την άσκοπη μετακίνηση του σερβιτόρου από το τραπέζι στην κουζίνα. Τα μπλοκ παραγγελιών αντικαθίστανται από ασύρματα τερματικά και τα μεταδίδουν από απόσταση, ασύρματα, άμεσα και συγχρόνως, σε πολλαπλά σημεία στους χώρους παρασκευής και στο ταμείο.

Ο σερβιτόρος μπορεί να πάρει παραγγελία ή να ανανεώσει τυχόν υπάρχουσα στο τραπέζι που χρειάζεται, να ενημερώνει και να βλέπει ανά πάσα στιγμή την κατάσταση της κάθε παραγγελίας με πληροφορίες όπως ποια αντικείμενα έχουν παραδοθεί και πληρωθεί και από ποιον, αλλά και ποιά αντικείμενα είνα έτοιμα από την κουζίνα.

Η κουζίνα ενημερώνεται αυτόματα για τις παραγγελίες και τα πράγματα που πρέπει να ετοιμάσει, καθώς και να βγάλει από το μενού ό,τι επιθυμεί, με αυτόματη αδυναμία επιλογής του αντικειμένου σε παραγγελία απο τον σερβιτόρο.

Ο διαχειριστής μπορεί να κάνει όλες τις λειτουργίες του προσωπικού αν επιθυμεί, έχοντας όμως επίσης πρόσβαση στη βάση δεδομένων της εφαρμογής ώστε να μπορεί να επιβλέπει κάθε κίνηση του προσωπικού του, όπως και αναλυτικές φόρμες, για παράδειγμα για το συνολικό πόσο που έχει συλλέξει ο κάθε σερβιτόρος, πόσες παραγγελίες έχει παραδόσει κ.λ.π.

Η συγκεκριμένη εφαρμογή δημιουργήθηκε στην πλατφόρμα Meteor [3] με χρήση της MongoDB [4] βάσης δεδομένων.

1.5. Παρόμοια Applications

Οι εφαρμογές στον τομέα της διαχείρισης εστατορίων έχουν πλέον πολλά χαρακτηριστικά, όπως η διαχείριση πίνακα και ταμειακών ροών, ο προγραμματισμός των εργαζομένων, η παρακολούθηση των παραγγελιών, η μισθοδοσία, τα analytics κ.ο.κ. Κάποιες από τις πιο διαδεδομένες εφαρμογές στον τομέα αυτό είναι οι εξής:

- Toast POS
  Απευθύνεται σε επιχειρήσεις όπως καφετέριες, εστιατόρια και μπαρ. Μπορεί να χρησιμοποιηθεί τόσο από τους εργαζομένους όσο και από τους πελάτες. Δίνεται η δυνατότητα διαχείρισης του μενού και του τιμοκαταλόγου, η παραγγελιοληψία και η παρακολούθηση των παραγγελιών, ενώ παράλληλα καταγράφεται η απόδοση του προσωπικού. Οι πελάτες έχουν τη δυνατότητα να ολοκληρώσουν τις πληρωμές τους μέσω της εφαρμογής.
- Lightspeed Restaurant
  Έχει σχεδιαστεί ώστε να χρησιμοποιείται από επιχειρήσεις όπως εστιατόρια, καφετέριες και γενικότερα παρόμοιου τύπου καταστήματα. Η χρήση της εφαρμογής απλοποιεί τη διαχείριση του προσωπικού, των αποθεμάτων και των κρατήσεων, διευκολύνει τη δημιουργία του μενού, της διαχείρισης των παραγγελιών και των πληρωμών και ταυτόχρονα γίνεται διαρκής καταγραφή της απόδοσης.

Η διαχείριση των εστιατορίων όμως περιλαμβάνει και άλλες υποχρεώσεις, στις οποίες δίνουν λύση εφαρμογές με λίγο διαφορετική νοοτροπία. Υπάρχουν εφαρμογές που βοηθούν στην καταγραφή και παρακολούθηση των αποθεμάτων, παρέχοντας λίστες με προμηθευτές και κόστη των προϊόντων, παρακολούθηση των τιμών των προϊόντων που χρησιμοποιεί κάθε καταστηματάρχης. Άλλες εφαρμογές στοχεύουν στη διαχείριση των κρατήσεων και της λίστας αναμονής, ενώ αυτοματοποιούν την ενημέρωση των πελατών για την κατάσταση της κράτησής τους. Υπάρχουν επίσης εφαρμογές που μελετούν τις προτιμήσεις των πελατών με σκοπό να δημιουργηθεί μόνιμη πελατεία [1].

Στην εφαρμογή που αναπτύχθηκε για την συγκεκριμένη διπλωματική δόθηκε κατά κόρον βάση στην σωστή, γρήγορη και εύκολη επικοινωνία του προσωπικού σε ώρες αιχμής. Καλύπτει όλες τις βασικές λειτουργίες που πρέπει να έχει μια τέτοια εφαρμογή αλλά δημιουργήθηκε έτσι ώστε καταστήματα με πολλά άτομα προσωπικό να μπορούν να οργανώνονται σωστά και εύκολα, αποθηκεύοντας το κάθε τους βήμα αναλυτικά ώστε ο κάθε χρήστης που έχει πρόσβαση στην εφαμοργή να γνωρίζει την «δουλειά» που πρέπει να κάνει.

2. Εργαλεία Ανάπτυξης και Σχεδιασμού

2.1. Τι είναι το Meteor

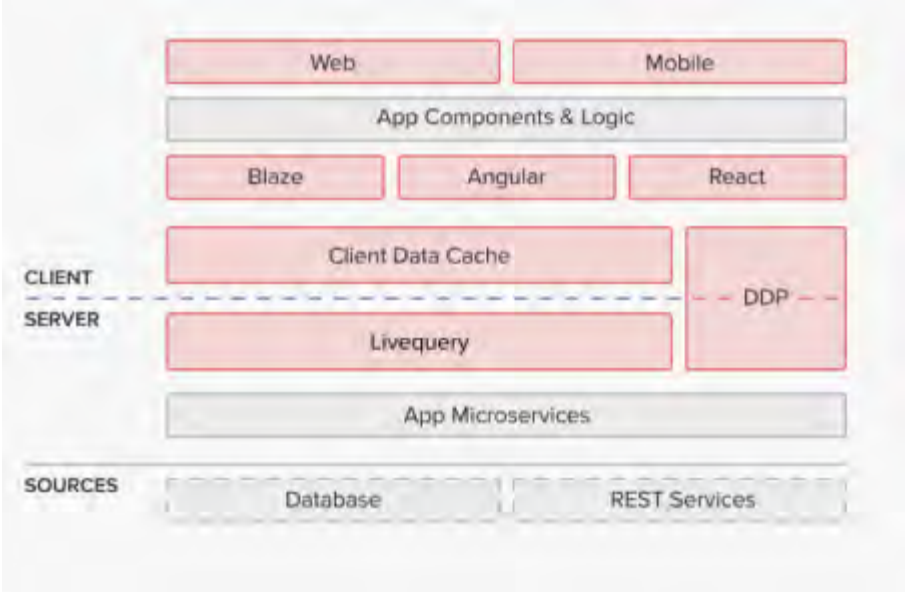
Το Meteor είναι μια JavaScript πλατφόρμα, ένα πλαίσιο κωδικοποίησης JS, καθώς προσφέρει έναν καινοτόμο τρόπο για την ανάπτυξη δυναμικών, πλούσιων, διαδραστικών ιστοσελίδων και εφαρμογών. Το Meteor [3] περιλαμβάνει ένα σύνολο βασικών τεχνολογιών για την κατασκευή αντιδραστικών(reactive) εφαρμογών, εργαλεία δημιουργίας και ένα επιμελημένο συνόλο πακέτων από το Node.js [5] και τη γενική κοινότητα JavaScript [6].

Το Meteor επιτρέπει την ανάπτυξη σε μία γλώσσα, JavaScript για όλα τα περιβάλλοντα: διακομιστή εφαρμογών, πρόγραμμα περιήγησης ιστού και κινητή συσκευή. Χρησιμοποιεί «data on the wire», δηλαδή ο διακομιστής στέλνει δεδομένα, όχι HTML, και παρέχει πλήρη αντιδραστικότητα στοίβας, επιτρέποντας στο περιβάλλον εργασίας(UI) να αντικατοπτρίζει άψογα την πραγματική κατάσταση της εφαρμογής με ελάχιστη αναπτυξιακή προσπάθεια. Θεωρείται καλή επιλογή τόσο για αρχάριους όσο και για προχωρημένους προγραμματιστές καθώς χρειάζεται γνώση μίας μόνο γλώσσας προγραμματισμού, δίνοντας την δυνατότητα δημιουργίας εγγενών εφαρμογών iOS και Android από την ίδια βάση κώδικα JS.

2.1.1. Meteor Αρχιτεκτονική

Η πλήρης αρχιτεκτονική του Meteor απεικονίζεται στο *σχήμα 1*, που δείχνει πώς η πλευρά του πελάτη(client-side) επικοινωνεί με την πλευρά του διακομιστή(server-side). Το Meteor αρχικά δημιουργήθηκε για να χρησιμοποιεί μόνο Blaze δομή ως στρώμα προβολής(view layer), το 2016, συμπεριλάμβανε επίσης το React και το AngularJS. Επί του παρόντος, χρησιμοποιεί το Blaze, React και το AngularJS ως δομή για προβολή και το Node.js και MongoDB για back-end. Το ανώτερο στρώμα του *σχήματος 1* δείχνει ότι το Meteor μπορεί να χρησιμοποιηθεί για την κατασκευή εφαρμογών ιστού ή κινητής. Κάτω από το ανώτερο στρώμα διακρίνονται οι τρεις Meteor δομές-τεχνολογίες front-end όπως Blaze, AngularJS και React, στη συνέχεια ακολουθεί η κρυφή μνήμη δεδομένων πελάτη (Client Data Cache).

Κάθε χρήστης που συνδέεται σε μια Meteor εφαρμογή θα έχει το αντίγραφο των δεδομένων πλευράς διακομιστή(server side data) σαν κρυφή μνήμη δεδομένων πελάτη, γνωστή ως Minimongo. Στη μέση του *σχήματος 1*, υπάρχει το πρωτόκολλο κατανεμημένων δεδομένων (DDP). Το Meteor χρησιμοποιεί το DDP για να επικοινωνήσει με τους πελάτες. Όπως απεικονίζεται στο *σχήμα 1*, το DDP είναι η γραμμή μεταξύ της πλευράς του πελάτη και του διακομιστή [7].



Εικόνα 2.1 Η αρχιτεκτονική Meteor (fjaguero.com (2017))

2.2. React

React είναι μια αποδοτική και ευέλικτη βιβλιοθήκη front-end της JS και είναι μια από τις πιο δημοφιλείς τεχνολογίες front-end για το σχεδιασμό web εφαρμογών. Το React χρησιμοποιεί JSX αρχεία, ένα βήμα παραπέρα των JS αρχείων, καθώς επιτρέπει τη σύνταξη XML μέσα στο ίδιο αρχείο. Το React κάνει την κατασκευή διαδραστικής διεπαφής χρήστη (UI) απλή. Σχεδιάζοντας μια απλή προβολή για κάθε κατάσταση σε μια εφαρμογή, το React θα ενημερώσει αποτελεσματικά και θα αποδώσει τα σωστά στοιχεία όταν μεταβάλλονται τα δεδομένα. Το React χτίζει ενσωματωμένα στοιχεία τα οποία διαχειρίζονται τις δικές τους καταστάσεις, και στη συνέχεια τα συνθέτει για να δημιουργήσει σύνθετα UI. Η λογική των React αντικειμένων στρέφεται γύρω από τη JS γλώσσα και δεν χρησιμοποιούνται πρότυπα(Templates), με αποτέλεσμα να είναι εύκολο να διαχειρίζονται μεγάλο όγκο δεδομένων και να μεταβάλλει την προβολή μόνο όταν χρειάζεται.

```
const element = <h1> Hello World </h1>
```

Η παραπάνω σύνταξη είναι ένα παράδειμα ενός JSX κώδικα που δεν είναι ούτε HTML ούτε JS. Κάθε στοιχείο JSX καθίσταται από ένα κεντρικό(root) αντικείμενο, που συνήθως γράφεται ως:

```
<div id = "root"></div>.
```

Το React χρησιμοποιεί διαφορετικά είδη εξαρτημάτων(components), όπως οι υποκατηγορίες. React.Component:

```
class ShoppingList extends React.Component {
  render() {
    return (
      <div className="shopping-list">
        <h1>Shopping List for {this.props.name}</h1>
        <ul>
          <li>Instagram</li>
          <li>WhatsApp</li>
          <li>Oculus</li>
        </ul>
      </div>
    );
  }
}

// Example usage: <ShoppingList name="Mark" />
```

Εικόνα 2.2 Παράδειγμα κώδικα React

Χρησιμοποιούνται εξαρτήματα τα όποια εμφανίζονται στην οθόνη. Όταν τα δεδομένα αλλάξουν, το React θα ενημερώσει αποτελεσματικά και θα αναδομήσει εκ νέου τα στοιχεία. Εδώ, το ShoppingList είναι μια κλάση συστατικού React. Ένα στοιχείο παίρνει παραμέτρους που ονομάζονται *props* (σύντομα για τις "properties"), και επιστρέφει μια ιεραρχία εξαρτημάτων για προβολή μέσω της μεθόδου rendering.

Η μέθοδος rendering επιστρέφει μια περιγραφή του τί επιθυμεί ο προγραμματιστής να εμφανιστεί στην οθόνη. Το React παίρνει την περιγραφή και εμφανίζει το αποτέλεσμα.

Για να περαστούν δεδομένα μέσω του *props,* όταν καλείται για παράδειγμα η Square συνάρτηση, τοποθετείται σαν παράμετρος με όνομα της επιλογής του προγραμματιστή το στοιχείο που επιθυμεί:

```
class Board extends React.Component {
  renderSquare(i) {
    return <Square value={i} />;
  }
```

Εικόνα 2.3 Πέρασμα μεταβλητής από γονιό

Και στο Square αρχείο διαβάζεται ως «{this.props.value}»

```
class Square extends React.Component {
  render() {
    return (
      <button className="square">
        {this.props.value}
      </button>
    );
  }
}
```

Εικόνα 2.4 Πέρασμα μεταβλητής σε παιδί

Εκτός από αυτές τις λειτουργίες, το React παρέχει επίσης ένα σύνολο επανακλήσεων(callbacks) που ενεργοποιούνται σε διάφορα σημεία κατά τη διάρκεια του κύκλου ζωής των εξαρτημάτων, πριν από τη φόρτωση, μετά τη φόρτωση, μετά την αποσυναρμολόγηση κ.ο.κ. Ένα λειτουργικό στοιχείο είναι μια συνάρτηση που παίρνει ένα αντικείμενο *props* ως όρισμα και επιστρέφει μια δέσμη HTML. Σχεδόν σαν ένα παραδοσιακό πρότυπο, με τη βασική διαφορά ότι μπορεί ακόμα να χρησιμοποιεί ό,τι JavaScript κώδικα χρειάζεται μέσα σε αυτή τη λειτουργία.

Μία από αυτές τις μεθόδους είναι State, όταν ένα στοιχείο πρέπει να αντιδρά σε δεδομένα που δεν προέρχονται από γονικό στοιχείο (όπως π.χ. είσοδος χρήστη). Το στοιχείο state είναι ένας τρόπος κράτησης, επεξεργασίας και χρήσης πληροφοριών που είναι εσωτερικές σε ένα δεδομένο στοιχείο. Το State είναι συνήθως ένα POJO (Απλό παλιό Java [Script] Object), και η αλλαγή του είναι ένας από τους λίγους τρόπους για να κάνει ένα συστατικό να ανακατασκευαστεί. Με απλά λόγια, κάθε φορά που μεταβάλλεται ένα state στοιχείο, ανανεώνονται όλα τα στοιχεία που προβάλλονται στην οθόνη από το συγκεκιρμένο JSX αρχείο. Είναι μια από τις πιο βασικές ιδέες πίσω από το React, αλλά έχει κάποιες ιδιότητες που το καθιστούν δύσκολο να χρησιμοποιηθεί και μπορεί να οδηγήσει σε απροσδόκητη συμπεριφορά στην εφαρμογή.

Τα States και τα props παίζουν σημαντικό ρόλο στην αλλαγή δυναμικών δεδομένων. Συνιστώσες JSX συνήθως αποδίδουν στο UI. Το React χρησιμοποιεί μια ενσωματωμένη λειτουργία για να δώσει προτεραιότητα στην απόδοση και να καταστρέψει την παραγόμενη συνιστώσα. Είναι πολύ σημαντικό να ελευθερώνονται οι πόροι που χρησιμοποιούνται από τα εξαρτήματα καταστρέφοντας τα παραγόμενα εξαρτήματα [8].

Το React χρησιμοποιεί ενσωματωμένες λειτουργίες για την παρακολούθηση του κύκλου ζωής των κατασκευαστικών στοιχείων. Οι δύο κοινώς χρησιμοποιούμενες ενσωματωμένες λειτουργίες του React είναι οι componentDidMount () και componentWillUnMount ().

Το στοιχείο componentDidMount () εκτελείται μετά την απόδοση της εξόδου του στοιχείου στο μοντέλο αντικειμένου εγγράφου(DOM).

Μετά τον κύκλο ζωής του κατασκευαστικού στοιχείου, το componentWillUnMount () θα καταστρέψει τα παραγόμενα στοιχεία [7].

2.3. Node.js

Το Node.js είναι ένα open-source, cross-platform JS run-time περιβάλλον για την ανάπτυξη διαφορετικών εφαρμογών. Το Node.js εισήγαγε ένα διαχειριστή πακέτων που ονομάζεται NPM τον Ιανουάριο του 2010. Η διαχείριση κόμβων (NPM) κάνει τη δημοσίευση και την κοινή χρήση της πηγής JS ευκολότερη και έχει σχεδιαστεί για την απλούστευση της εγκατάστασης, της ενημέρωσης και της χρήσης των βιβλιοθηκών. Τα δεδομένα δεν πρέπει να ζητούνται μόνο από τον πελάτη, για να κάνουν το web αντιδραστικό, τα δεδομένα θα πρέπει επίσης να ωθηθούν από το διακομιστή στον πελάτη. Το Node.js είναι καλό σε πραγματικό χρόνο web εφαρμογών μέσω της χρήσης τεχνολογίας ώθησης (technology over web-sockets). Το περιβάλλον χρόνου εκτέλεσης Node.js ερμηνεύει την JS αξιοποιώντας πλήρως το V8 της Google JS. Ο μηχανισμός V8 JS επιτρέπει στο Node.js να δημιουργήσει ένα περιβάλλον χρόνου εκτέλεσης που παρέχει JS από το διακομιστή στον πελάτη πολύ γρήγορα. Το V8 μετατρέπει το JS σε κώδικα μητρικής μηχανής, δίνοντας στο Node.js την μεγάλη ταχύτητα του. Η ταχύτητα απόκρισης, σε συνδυασμό με τον ασύγχρονο προγραμματισμό, κάνουν το Node.js τόσο αποκριτικό και επιθυμητό στην κατασκευή δυναμικών και αντιδραστικών εφαρμογών. Το Meteor εκμεταλλεύτηκε τις δυνατότητες του Node.js με τη χρήση του στο back-end για την παροχή περιεχομένου στον πελάτη [9].

3. MONGODB

3.1. Πως ξεκίνησαν όλα

Η MongoDB ιδρύθηκε το 2007 από τους Dwight Merriman, Eliot Horowitz και Kevin Ryan - την ομάδα πίσω από το DoubleClick. Στη διαδικτυακή εταιρία διαφήμισης DoubleClick (που τώρα ανήκει στη Google), η ομάδα ανέπτυξε και χρησιμοποίησε βάσεις δεδομένων προσαρμοσμένες ώστε να αντιμετωπίσει τις αδυναμίες των υφιστάμενων βάσεων δεδομένων. Η επιχείρηση εξυπηρετούσε 400.000 διαφημίσεις ανά δευτερόλεπτο, αλλά συχνά περιοριζόταν τόσο με την επεκτασιμότητα όσο και με την ευκινησία των δεδομένων. Αγανακτισμένη, η ομάδα εμπνεύστηκε για να δημιουργήσει μια βάση δεδομένων που ξεπερνούσε τις προκλήσεις που αντιμετώπισε στο DoubleClick.

Η MongoDB είναι η κορυφαία σύγχρονη NoSQL πλατφόρμα βάσεων δεδομένων γενικής χρήσης, σχεδιασμένη για να απελευθερώσει τη δύναμη του λογισμικού και των δεδομένων για τους προγραμματιστές και τις εφαρμογές που κατασκευάζουν. Με έδρα τη Νέα Υόρκη, με γραφεία σε όλη τη Βόρεια Αμερική, την Ευρώπη και την Ασία-Ειρηνικό, διαθέτει περισσότερους από 4.300 πελάτες σε περισσότερες από 85 χώρες. Η πλατφόρμα βάσης δεδομένων MongoDB έχει κατεβεί πάνω από 30 εκατομμύρια φορές και έχουν γίνει πάνω από 730.000 εγγραφές Πανεπιστημίου [10].

3.2. Τι είναι NoSQL;

Το NoSQL περιλαμβάνει μια μεγάλη ποικιλία διαφορετικών τεχνολογιών βάσεων δεδομένων που αναπτύχθηκαν ως απάντηση στις απαιτήσεις που παρουσιάζονται για την κατασκευή σύγχρονων εφαρμογών. Οι προγραμματιστές συνεργάζονται με εφαρμογές που δημιουργούν τεράστιους όγκους νέων, ταχέως μεταβαλλόμενων τύπων δεδομένων - δομημένα, ημιδομημένα, αδόμητα και πολυμορφικά δεδομένα. Οι εφαρμογές που εξυπηρετούνταν από ένα πεπερασμένο κοινό τώρα παρέχονται ως υπηρεσίες που πρέπει πάντα να είναι προσβάσιμες από πολλές διαφορετικές συσκευές και να κλιμακώνονται παγκοσμίως σε εκατομμύρια χρήστες. Οι οργανισμοί στρέφονται τώρα σε αρχιτεκτονικές κλιμάκωσης χρησιμοποιώντας λογισμικό ανοιχτού κώδικα, διακομιστές προϊόντων και cloud computing, αντί για μεγάλους μονολιθικούς servers και υποδομές αποθήκευσης. Οι σχεσιακές βάσεις δεδομένων (Relational databases) δε σχεδιάστηκαν για να αντιμετωπίσουν τις προκλήσεις της κλίμακας και της ευελιξίας που αντιμετωπίζουν οι σύγχρονες εφαρμογές, ούτε δημιουργήθηκαν για να επωφεληθούν από την ικανότητα επεξεργασίας και αποθήκευσης που διατίθενται σήμερα [11].

3.3. NoSQL v. SQL Σύνοψη

Πίνακας 3.1 NoSQL vs SQL

	SQL ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ	NOSQL ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ
Τύποι	Ένας τύπος (βάση δεδομένων SQL) με μικρές παραλλαγές.	Πολλοί διαφορετικοί τύποι, συμπεριλαμβανομένων των key-value stores, document databases, wide-column stores, and graph databases.
Ιστορικό ανάπτυξης	Αναπτύχθηκε στη δεκαετία του 1970 για να αντιμετωπίσει το πρώτο κύμα εφαρμογών αποθήκευσης δεδομένων.	Αναπτύχθηκε στα τέλη της δεκαετίας του 2000 για να αντιμετωπίσει τους περιορισμούς των βάσεων δεδομένων SQL, ιδιαίτερα την ικανότητα κλιμάκωσης, των πολλαπλών δομημένων δεδομένων.
Παραδείγματα	MySQL, Postgres, Microsoft SQL Server, Oracle Database.	MongoDB, Cassandra, HBase, Neo4j.
Μοντέλο αποθήκευσης δεδομένων	Οι μεμονωμένες εγγραφές (π.χ. «υπάλληλοι») αποθηκεύονται ως σειρές σε πίνακες, με κάθε στήλη να αποθηκεύει ένα συγκεκριμένο κομμάτι δεδομένων σχετικά με το συγκεκριμένο αρχείο (π.χ. «διαχειριστής», «ημερομηνία πρόσληψης», κλπ.), όπως σε ένα υπολογιστικό φύλλο. Τα σχετικά δεδομένα αποθηκεύονται σε ξεχωριστούς πίνακες και στη συνέχεια συνδέονται μεταξύ τους όταν εκτελούνται πιο σύνθετα ερωτήματα. Για παράδειγμα, τα «γραφεία» μπορούν να αποθηκευτούν σε ένα τραπέζι και οι «υπάλληλοι» σε ένα άλλο. Όταν ένας χρήστης θέλει να βρει τη διεύθυνση εργασίας ενός υπαλλήλου, ο μηχανισμός βάσης δεδομένων συνδέει τους πίνακες «υπαλλήλου» και «γραφείου» μαζί, για να πάρει όλες τις απαραίτητες πληροφορίες.	Διαφέρει ανάλογα με τον τύπο της βάσης δεδομένων. Για παράδειγμα, οι key-value stores λειτουργούν παρόμοια με τις βάσεις δεδομένων SQL, αλλά έχουν μόνο δύο στήλες ('κλειδί' και 'τιμή'), με πιο περίπλοκες πληροφορίες που μερικές φορές αποθηκεύονται ως BLOB μέσα στις στήλες 'value'. Οι βάσεις δεδομένων εγγράφων καταργούν εντελώς το μοντέλο πίνακα και γραμμών, αποθηκεύοντας όλα τα σχετικά δεδομένα σε ένα ενιαίο έγγραφο σε μορφή JSON, XML ή σε άλλη μορφή, η οποία μπορεί να λειτουργήσει ιεραρχικά.

Σχήματα	Οι δομές και οι τύποι δεδομένων καθορίζονται εκ των προτέρων. Για να αποθηκεύτουν πληροφορίες σχετικά με ένα νέο στοιχείο δεδομένων, ολόκληρη η βάση δεδομένων πρέπει να τροποποιηθεί, οπότε η βάση δεδομένων πρέπει να τεθεί εκτός σύνδεσης. (offline)	Τυπικά δυναμική, με κάποιους κανόνες επιβολής της επικύρωσης δεδομένων. Οι εφαρμογές μπορούν να προσθέσουν νέα πεδία σε κίνηση και, σε αντίθεση με τις σειρές πίνακα SQL, διαφορετικά δεδομένα μπορούν να αποθηκευτούν μαζί ανάλογα με τις ανάγκες. Για κάποιες βάσεις δεδομένων (π.χ. wide-column stores), είναι κάπως πιο δύσκολο να προσθέσουν δυναμικά νέα πεδία.
Κλιμάκωση	Η κλιμάκωση γίνεται κάθετα, που σημαίνει ότι ένας μόνο εξυπηρετητής πρέπει να γίνεται ολοένα και πιο ισχυρός για να αντιμετωπίσει την αυξημένη ζήτηση. Είναι δυνατή η εξάπλωση βάσεων δεδομένων SQL σε πολλούς διακομιστές, αλλά γενικά απαιτείται σημαντική πρόσθετη μηχανική και βασικά χαρακτηριστικά σχεσιακής συμπεριφοράς όπως JOIN, αναλογική ακεραιότητα και οι συναλλαγές συνήθως χάνονται.	Γίνεται οριζόντια, που σημαίνει ότι για να προστεθεί χωρητικότητα, ένας διαχειριστής βάσης δεδομένων μπορεί απλά να προσθέσει περισσότερους διακομιστές βασικών προϊόντων ή cloud instances. Η βάση δεδομένων εξαπλώνει αυτόματα τα δεδομένα μεταξύ των διακομιστών ανάλογα με τις ανάγκες.
Αναπτυξιακό μοντέλο	Μίγμα ανοιχτού κώδικα (π.χ. Postgres, MySQL) και κλειστού κώδικα (π.χ. Oracle Database).	Ανοικτού κώδικα.
Υποστηρίζει τις συναλλαγές	Ναι, οι ενημερώσεις μπορούν να ρυθμιστούν ώστε να ολοκληρωθούν πλήρως ή καθόλου.	Σε ορισμένες περιπτώσεις και σε συγκεκριμένα επίπεδα (π.χ. επίπεδο εγγράφου vs. επίπεδο βάσης δεδομένων).
Χειρισμός δεδομένων	Ειδική γλώσσα χρησιμοποιώντας τις επιλογές επιλογής, εισαγωγής και ενημέρωσης, π.χ. ΕΠΙΛΕΞΤΕ τα πεδία από τον πίνακα WHERE ...	Μέσα από αντικειμενοστρεφή API(object-oriented APIs).
Συνέπεια	Μπορεί να ρυθμιστεί για να έχει ισχυρή συνέπεια.	Εξαρτάται από το προϊόν. Μερικοί παρέχουν ισχυρή συνέπεια (π.χ. MongoDB, με συντονισμένη συνοχή για τις αναγνώσεις) ενώ άλλες προσφέρουν τελική συνέπεια (π.χ. Cassandra).

3.4. Τι είναι η MongoDB

Η MongoDB είναι μία εγγραφοκεντρική βάση δεδομένων (document-oriented database) που προσανατολίζεται προς διάφορες πλατφόρμες. Ως NoSQL βάση δεδομένων, αγνοεί την παραδοσιακή δομή σχεσιακής βάσης δεδομένων (table-based) υποστηρίζοντας κυρίως JSON έγγραφα με δυναμικά σχήματα (της μορφής BSON), καθιστώντας την ενσωμάτωση δεδομένων σε ορισμένους τύπους εφαρμογών ευκολότερη και ταχύτερη.

Οι βασικές δομικές πτυχές της MongoDB χωρίζονται σε:

Data Model Μοντέλο δεδομένων
GridFS
Sharding κλιμάκωση
Data partitioning
Aggregation
Replication Αναπαραγωγή
Index
Ειδικότερα:

A. Μοντέλο δεδομένων

Η MongoDB αποθηκεύει δεδομένα με τη μορφή κωδικοποιημένων εγγράφων JSON BSON-Binar. Τα πεδία στα έγγραφα BSON ενδέχεται να περιέχουν συστοιχίες τιμών ή ενσωματωμένα έγγραφα(documents). Στη MongoDB, η βάση δεδομένων είναι μια ομάδα από σχετικές συλλογές(collections). Κάθε βάση δεδομένων έχει ένα ξεχωριστό σύνολο αρχείων και μπορεί να περιέχει μεγάλο αριθμό από συλλογές. Μια ενιαία εγκατάσταση MongoDB μπορεί να έχει πολλές βάσεις δεδομένων.

Κάθε καταχώρη ση στη MongoDB θεωρείται ως ένα έγγραφο(document), που είναι μια δομή δεδομένων που αποτελείται από ζεύγη πεδίων(fields) και τιμών(values). Τα έγγραφα MongoDB είναι παρόμοια με τα αντικείμενα JSON με τις τιμές των πεδίων να μπορούν να περιλαμβάνουν άλλα έγγραφα, πίνακες και συστοιχίες εγγράφων. Αυτή είναι μια σημαντική διαφοροποίηση από τα συστήματα RDBMS(Relational Database Management System) όπου κάθε πεδίο πρέπει να περιέχει μόνο μία τιμή.

Η MongoDB αποθηκεύει έγγραφα σε συλλογές που είναι ανάλογες με τους πίνακες σε σχεσιακές βάσεις δεδομένων. Στα RDBMS όλοι οι πίνακες, σε μια βάση δεδομένων, πρέπει να έχουν το ίδιο σχήμα(schema), αλλά στη MongoDB δεν υπάρχει τέτοια απαίτηση. Αυτός ο σχεδιασμός χωρίς σχήματα είναι μια καινοτομία που καθιστά την MongoDB την πιο χρησιμοποιημένη βάση δεδομένων NoSQL. Ωστόσο, τα έγγραφα που είναι αποθηκευμένα σε μια συλλογή πρέπει να έχουν και ένα μοναδικό πεδίο _id που λειτουργεί ως πρωτεύον κλειδί(primary key). Τα έγγραφα σε μια συλλογή μπορούν να αποθηκευτούν είτε ως κανονικοποιημένα είτε ως ενσωματωμένα σε άλλο έγγραφο.

Εικόνα 3.1 Παράδειγμα από Documents [12]

a) Μοντέλα κανονικοποιημένων δεδομένων

Οι σχέσεις μεταξύ των δεδομένων αποθηκεύονται με συνδέσμους (αναφορές) από ένα έγγραφο σε άλλο. Οι αναφορές αυτές χρησιμοποιούνται από την εφαρμογή για την εξαγωγή των σχετικών δεδομένων.

b) Ενσωματωμένα μοντέλα δεδομένων

Τα ενσωματωμένα έγγραφα δημιουργούν τις σχέσεις μεταξύ των δεδομένων, αποθηκεύοντας τα σχετικά δεδομένα σε μια ενιαία δομή εγγράφων. Αυτά τα μοντέλα δεδομένων επιτρέπουν στις εφαρμογές να ανακτούν και να χειρίζονται τα σχετικά δεδομένα σε μία λειτουργία βάσης δεδομένων.

Με απλά λόγια τα μοντέλα κανονικοποιημένων δεδομένων είναι διαφορετικά έγγραφα, που συνδυάζονται διότι χρησιμοποιούν κοινά πεδία _id, ενώ τα ενσωματωμένα μοντέλα δεδομένων είναι ένα ενιαίο έγγραφο που περιέχει όλες τις πληροφορίες .

B. GridFS

Το GridFS είναι μια προδιαγραφή για την αποθήκευση και ανάκτηση αρχείων που υπερβαίνουν το όριο μεγέθους εγγράφου BSON 16MB. Αντί να αποθηκεύει ένα αρχείο σαν ένα ενιαίο έγγραφο, το GridFS διαιρεί ένα αρχείο σε τμήματα και αποθηκεύει κάθε τμήμα ως ξεχωριστό έγγραφο. Το GridFS χρησιμοποιεί δύο συλλογές για την αποθήκευση αρχείων. Μια συλλογή αποθηκεύει τα κομμάτια των αρχείων και τα άλλα αποθηκεύουν τα μεταδεδομένα(metadata) των αρχείων. Όταν η εφαρμογή ζητάει από το GridFS ένα αρχείο, αυτό ανασυνθέτει τα κομμάτια ανάλογα με τις ανάγκες. Επίσης υπάρχει πρόσβαση σε πληροφορίες από τυχόν τυχαία τμήματα των αρχείων. Αυτό το χαρακτηριστικό είναι αυτό που επιτρέπει βασικά την αναπαραγωγή ενός αρχείου βίντεο ή ήχου από το επιθυμητό σημείο.

C. Sharding

Τα συστήματα βάσεων δεδομένων με μεγάλα σύνολα δεδομένων και εφαρμογές υψηλής απόδοσης μπορούν να «προκαλέσουν» την ικανότητα ενός μόνο διακομιστή με πολλούς τρόπους, όπως:

Οι υψηλές συχνότητες αναζητήσεων πιέζουν πολύ τον CPU του διακομιστή. Τα όλο και μεγαλύτερα σύνολα δεδομένων υπερβαίνουν την χωρητικότητα αποθήκευσης ενός μόνο μηχανήματος. Τα μεγέθη δεδομένων που είναι μεγαλύτερα από τη μνήμη RAM του συστήματος υποβαθμίζουν την χωρητικότητα I / O των μονάδων δίσκου.

Για την αντιμετώπιση αυτών των ζητημάτων, τα συστήματα βάσεων δεδομένων έχουν τις δύο βασικές προσεγγίσεις που αναφέρθηκαν παραπάνω. Η MongoDB υποστηρίζει την κλιμάκωση, το «σπάσιμο» των δεδομένων, μέσω της διαδικασίας του χωρισμού συστοιχιών. Η διαδικασία γίνεται χωρίζοντας τα δεδομένα σε θραύσματα(shards) τα οποία στέλνονται σε διαφορετικούς διακομιστές. Όταν μια εφαρμογή χρειαστεί αυτά τα δεδομένα η MongoDB κάνει τη διασύνδεση με τα κατάλληλα θραύσματα και στη συνέχεια επιστρέφει τα αποτελέσματα στους πελάτες.

Οι διακομιστές διαμόρφωσης χρησιμοποιούν τα μεταδεδομένα του συμπλέγματος ως μέσο χαρτογράφησης του συνόλου των θραυσμάτων. Ο δρομολογητής(router) χρησιμοποιεί αυτά τα μεταδεδομένα για να στοχεύσει λειτουργίες σε συγκεκριμένα θραύσματα.

D. Data partitioning

Η MongoDB αποθηκεύει τα δεδομένα στο δίσκο ή στους διακομιστές ανάλογα με την συλλογή(collection) στην οποία ανήκει. Αυτό επιτυγχάνεται με τη χρήση ενός κλειδιού(shard key). Η MongoDB χρησιμοποιεί δύο είδη κλειδιών.

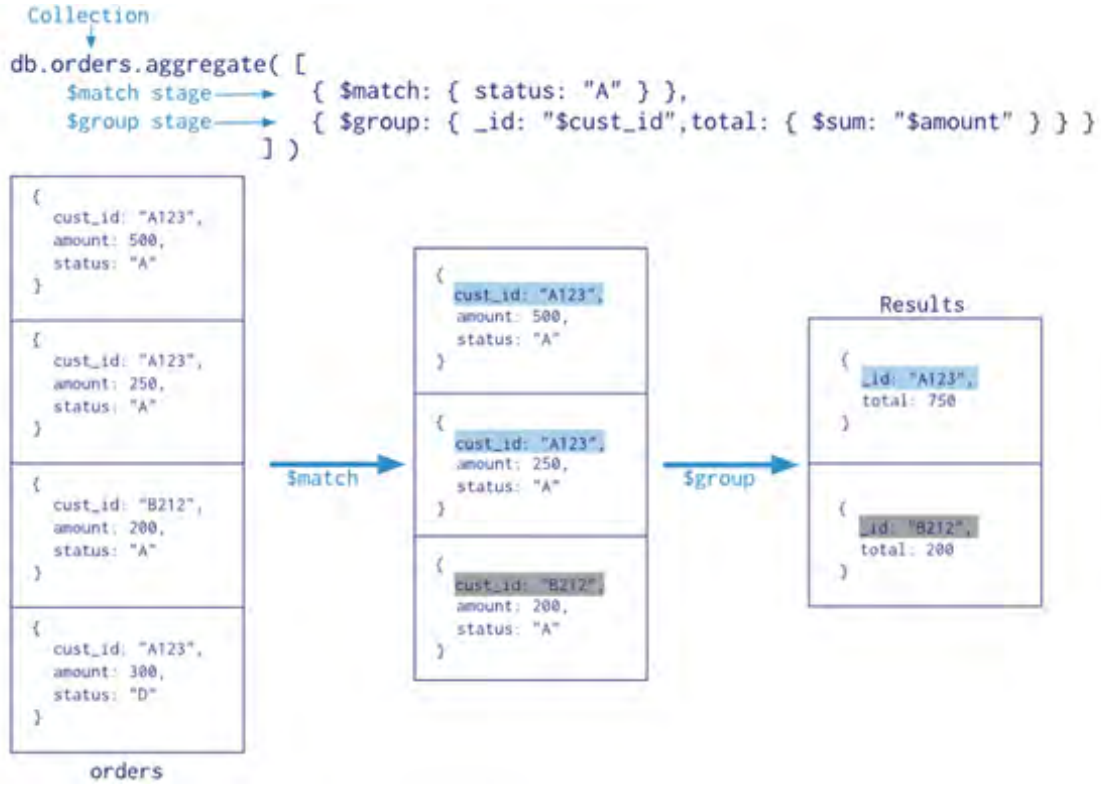
Τα κλείδια χωρίζονται ανάλογα με την περιοχή, δηλαδή θραύσματα που ανήκουν στο ίδιο δεδομένο έχουν ενα αριθμητικό κλειδί που είναι πολύ κοντά μεταξύ τους. Η MongoDB όμως μπορεί να δημιουργεί τα κλειδία της με βάση τον κατακερματισμό αντί για την περιοχή. Με απλά λογία, χωρίζονται με βάση το μέγεθος τους, με τη χρήση της μεταβλητής «HASH», ώστε δύο δεδομένα με κοντινές τιμές κλειδιού(παρόμοιο μέγεθος) να μην αποθηκεύονται στο ίδιο τμήμα. Αυτό εξασφαλίζει μια πιο τυχαία διανομή μιας συλλογής στο σύμπλεγμα.

E. Aggregations

«Aggregations» είναι λειτουργίες που επεξεργάζονται τα αρχεία δεδομένων και επιστρέφουν τα υπολογιζόμενα αποτελέσματα, είναι με απλά λόγια ένα είδος φίλτρου που χωρίζει τα δεδομένα. Η MongoDB χρησιμοποιεί συλλογές εγγράφων ως είσοδο με πιθανή έξοδο ενός ή περισσότερων

εγγράφων. Είναι μια σειρά διαδικασιών που εκτελούνται σταδιακά και παρέχουν φίλτρα που λειτουργούν σαν μετασχηματισμοί εγγράφων, που τροποποιούν τη μορφή του εγγράφου εξόδου.

Το MapReduce είναι ένα ισχυρό και ευέλικτο εργαλείο που χρησιμοποιείται για τη συγκέντρωση δεδομένων. Μπορεί να λύσει προβλήματα που είναι πολύπλοκα διαλύοντας το σε τεμάχια τα οποία στέλνει σε διαφορετικά μηχανήματα και αφήνει σε κάθε μηχανή να λύσει το δικό του τμήμα του προβλήματος. Όταν όλα τα μηχανήματα έχουν τελειώσει, όλα τα κομμάτια της λύσης έχουν συγχωνευθεί πίσω σε μια πλήρη μορφή.



Εικόνα 3.2 MongoDB aggregate [13]

F. Replication

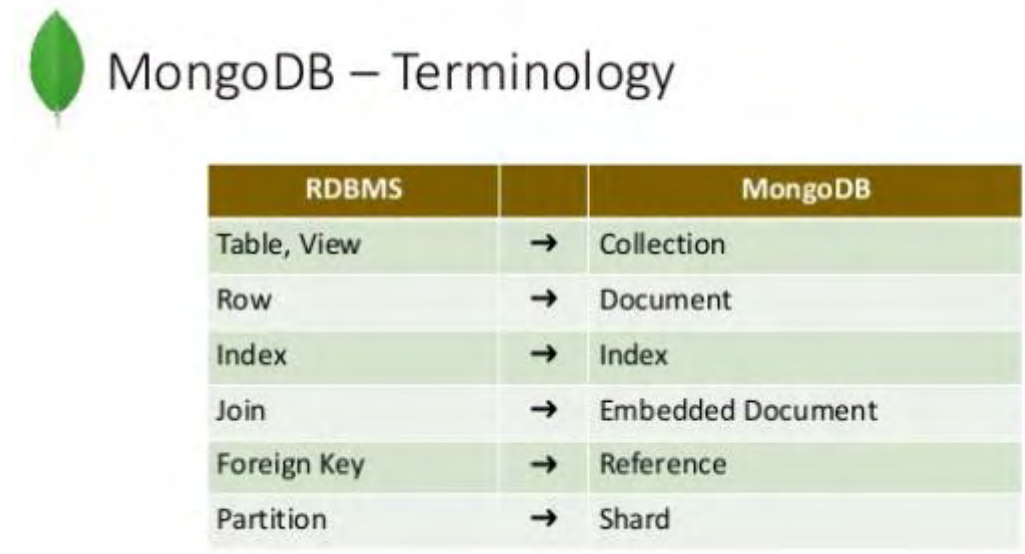
Η MongoDB δημιουργεί πολλαπλά αντίγραφα των δεδομένων σε διαφορετικούς διακομιστές με σκοπό να προστατεύει μια βάση δεδομένων από αποτυχία υλικού και διακοπές υπηρεσιών, που πιθανόν να προσφέρει ένας μόνο διακομιστής. Ουσιαστικά δημιουργεί μια ομάδα που φιλοξενεί το ίδιο σύνολο δεδομένων. Το πρωτεύον MongoDB, λαμβάνει όλες τις λειτουργίες εγγραφής και ανάγνωσης και μετέπειτα τα «αντίγραφα» εφαρμόζουν τις ίδιες πράξεις εγγραφής από το πρωτεύον, έτσι ώστε να έχουν το ίδιο σύνολο δεδομένων. Για να επιτευχθεί σωστά η «αντιγραφή», το πρωτεύον καταγράφει όλες τις αλλαγές στα σύνολα δεδομένων της στο oplog (αρχείο καταγραφής λειτουργίας). Εάν το πρωτεύον δεν είναι διαθέσιμο, τα «αντίγραφα» θα επιλέξουν ένα απο αυτά για να είναι πρωτεύον.

G. Index

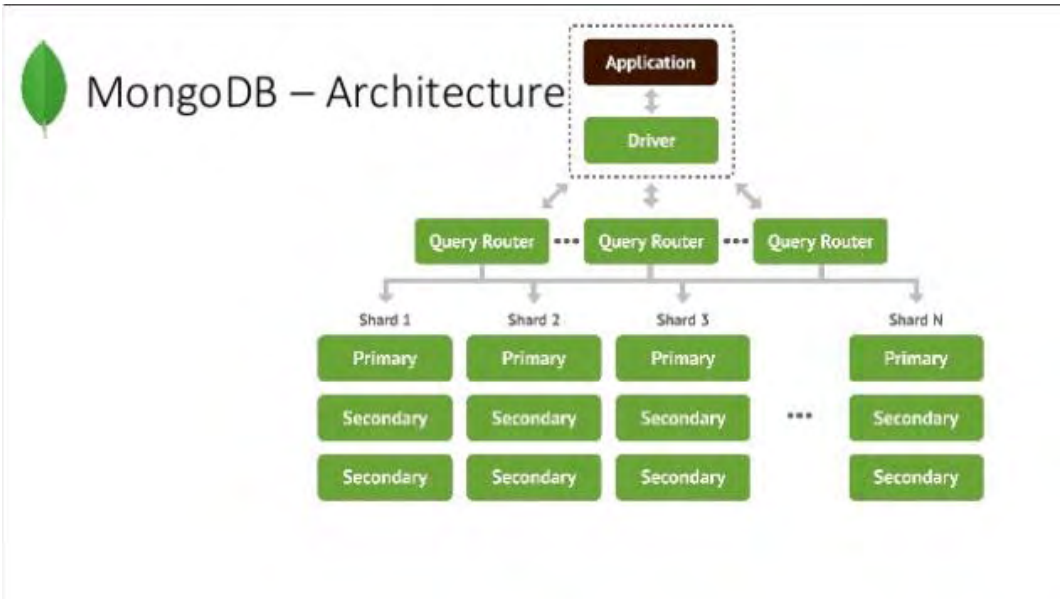
«Index» είναι ίσως η πιο χρήσιμη δυνατότητα που μπορεί να προσφέρει μια βάση δεδομένων για τη σύγχρονη εποχή. Όταν μία βάση δεδομένων έχει μεγάλο όγκο δεδομένων, όπως για παράδειγμα μία τράπεζα με εκατομμύρια πελάτες, θα ήταν πολύ μη-αποδοτικό κατά την αναζήτηση ενός συγκεκριμένου γκρουπ ατόμων, όπως με ένα συγκεκριμένο ποσό στον λογαριασμό τους, να έπρεπε να τους ψάχνει έναν-έναν σε όλη τη βάση της. Το «Index» λύνει ακριβώς αυτό το πρόβλημα, δίνει την δυνατότητα στη βάση να «κρατήσει ένα αρχείο» για το πού να βρει τα δεδομένα για συγκεκριμένες αναζητήσεις, ανάλογα με τις ανάγκες της εφαρμογής. Με απλά λόγια δημιουργεί μια σελίδα περιεχομένων όπως έχουν τα βιβλία, για το πού να βρεις την επιθυμητή λέξη ή κεφάλαιο [11].

3.5. Γιατί MongoDB

Δεν είναι σωστό να ειπωθεί πως η MongoDB είναι «καλύτερη» σε όλα από τις παραδοσιακές βάσεις δεδομένων. Είναι απλά διαφορετική. Οι SQL βάσεις δεδομένων είναι πολύ χρήσιμες και πρακτικές για δεδομένα με «σταθερό σχήμα», όπως δεδομένα με στατιστίκες πωλήσεων, από την άλλη όμως η MongoDB προσφέρει μια ευελιξία, ευκολία και ταχύτητα σε σχέση με τις παραδοσιακές βάσεις δεδομένων.

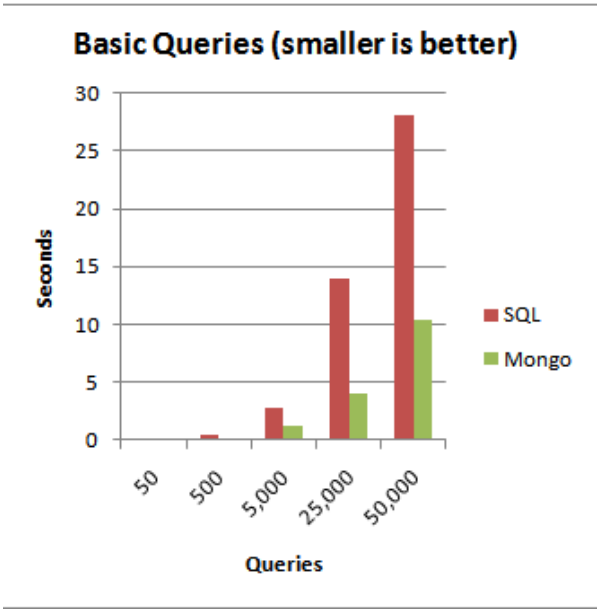


Εικόνα 3.3 Ορολογία RDBMS - MongoDB [14]



Εικόνα 3.4 Αρχιτεκτονική MongoDB [15]

Στην παραπάνω εικόνα, φαίνεται ίσως η πιο απλοϊκή μορφή της αρχιτεκτονικής της MongoDB. Για παράδειγμα, όταν η εφαρμογή θα αναζητήσει μια πληροφορία, η εφαρμογή δε χρειάζεται να γνωριζεί πού είναι αποθηκευμένη η πληροφορία, αρκεί να γνωρίζει τον δρομολογητή(Query Router), ενδέχεται να είναι και πολλοί και να μη βρίσκονται στο ίδιο «φυσικό» μηχάνημα. Στην συνέχεια, θα αναζητήσει τα θραύσματα που είναι σίγουρα διαθέσιμα, χάρη στη μέθοδο με τα αντίγραφα που προαναφέρθηκε και θα επιστρέψει το επιθυμητό ολοκληρωμένο με δυναμικό «σχήμα» έγγραφο.



Εικόνα 3.5 MongoDB vs RSDB [16]

Όπως φαίνεται από την παραπάνω αναπαράσταση, όταν αυξάνεται ο αριθμός των ερωτημάτων(queries) που «χτυπούν» τον διακομιστή, η MongoDB είναι σαφής νικητής. Η MongoDB χρησιμοποιείται συνήθως για αναλύσεις σε πραγματικό χρόνο, όπου ο χρόνος απόκρισης πρέπει να είναι χαμηλός και οι απαιτήσεις διαθεσιμότητας πολύ υψηλές.

Η MongoDB έχει έρθει στο προσκήνιο λόγω της ανάγκης των οργανισμών να αναλύουν ημιδομημένα, αδόμητα, γεωχωρικά δεδομένα και επειδή η δομή των δεδομένων μεταβάλλεται ταχύτατα στο σημερινό κόσμο. Τα παραδοσιακά συστήματα RDBMS δεν είναι σε θέση να ανταπεξέλθουν πλήρως στις απαιτήσεις αυτές, καθώς η εγγενής δομή τους δεν τους επιτρέπει να το κάνουν.

Παρόλο που γίνονται αλλαγές και στα συστήματα RDBMS, για να αντιμετωπιστεί αυτή η «έκρηξη» δεδομένων, οι βάσεις δεδομένων όπως η MongoDB με τη δομή των εγγράφων τους είναι οι πλέον κατάλληλες για την αντιμετώπιση των σημερινών απαιτήσεων.

3.6. MongoDB περιορισμοί

Υπάρχουν όμως και μερικοί περιορισμοί που πρέπει να έχει κάποιος υπόψη πριν επιλέξει την MongoDB, μερικοί απο τους οποίους είναι:

1. Μέγιστο μέγεθος εγγράφου στα 16 MB.
2. Μέγιστο αριθμό εγγράφων μέσα σε άλλα έγγραφα(100).
3. Ορισμένες «ομαδικές» εντολές δε θα λειτουργήσουν σε θραύσματα χωρισμένα σε διαφορετικούς διακομιστές.
4. Μέγιστο αριθμό bytes κλειδιών(shard keys) στα 512, τα οποία δεν μπορούν και να αλλάξουν σε τιμή μόλις δημιουργηθούν.
5. Για να γίνει χωρισμός σε θραύσματα πρέπει να είναι η συλλογή κάτω απο 256GB.
6. Τα ενσωματωμένα μοντέλα δεδομένων, όταν επεξεργάζονται δεδομένα με πολύπλοκες σχέσεις και επιρροές μεταξύ τους, καθιστούν δύσκολη την ροή λειτουργιών σε ένα ενιαίο έγγραφο [17].

4. Σχεδιασμός και Υλοποίηση

4.1. Ιεραρχία καταλόγων

Το Meteor δίνει την δυνατότητα στους προγραμματιστές να αναπτύξουν αρχεία κώδικα που θα χρησιμοποιούνται από κοινού από υπολογιστές, κινητά και τάμπλετ, οπότε δε χρειάζεται η ανάπτυξη διαφορετικών εκδόσεων για κάθε είδος συσκευής. Υπάρχει λοιπόν ένας φάκελος για τον Server του Web application και ένας για τον client.

Στο Server φάκελο υπάρχουν δύο μόνο αρχεία κώδικα, καθώς το application είναι κατά κόρον client-side και δε χρειάζεται πολλούς υπολογισμούς από την πλευρά του διακομιστή. Το πρώτο αρχείο με όνομα Publish περιέχει τις συναρτήσεις που δίνουν τη δυνατότητα στον client να αποκτήσει πρόσβαση σε ανάλογα κομμάτια της βάσης δεδομένων, χωρίς όμως να μπορεί την επεξεργαστεί. Το δεύτερο αρχείο με όνομα methods περιλαμβάνει τις συναρτήσεις που αν έχουν σωστά ορίσματα και από χρήστες με αντίστοιχα δικαιώματα, επεξεργάζονται ανάλογα με την επιθυμία του client τη βάση.

Στο φάκελο του client είναι τα κομμάτια κώδικα που «τρέχουν» στη συσκευή του κάθε χρήστη ανά πάσα στιγμή. Περιέχει αρχεία JSX για σωστή δημιουργία και διαχείριση των λογιαριασμών, ένα ξεχωριστό αρχείο JSX για κάθε λειτουργία του application, το κεντρικό «layout»(την πρώτη δηλαδή σελίδα κατά την εισαγωγή), έναν υποκατάλογο με βοηθητικά στοιχεία και κομμάτια κώδικα για τα jsx αρχεία της κάθε λειτουργίας και ένα αρχείο routes.jsx που περιέχει τις πληροφορίες για την τοποθεσία και ονομασία της κάθε λειτουργίας, ώστε να ξέρει ο περιηγητής ποιο αρχείο χρειάζεται και πού να το βρει.

Κατά τη δημιουργία του Web application στη συσκευή που αναπτύχθηκε, δημιουργήθηκαν και δύο ακόμα φάκελοι meteor και node_modules, με απαραίτητα αρχεία για το πρώτο «τρέξιμο» της εφαρμογής και κατά την αύξηση των απαιτήσεων των λειτουργιών της εφαρμογής προστέθηκαν διάφορα πακέτα από το accounts-ui για τη δημιουργία λογαριασμών και ground:db για την δημιουργία τοπικής βάσης δεδομένων στη συσκευή.

4.2. Heroku

Το συγκεκριμένο Web application για τις ανάγκες της διπλωματικής «ανέβηκε online» με την χρήση μιας δωρεάν cloud πλατφόρμας, το Heroku. Το Heroku είναι μια cloud πλατφόρμα υπηρεσίας (PaaS) που υποστηρίζει πολλές γλώσσες προγραμματισμού. Το Heroku, μια από τις πρώτες πλατφόρμες του cloud, αναπτύσσεται από τον Ιούνιο του 2007, όπου υποστηρίζε μόνο τη γλώσσα προγραμματισμού Ruby, αλλά τώρα υποστηρίζει Java, Node.js, Scala, Clojure, Python, PHP και Go. Για το λόγο αυτό, το Heroku λέγεται ότι είναι μια πολυγλωσσική πλατφόρμα, καθώς επιτρέπει στον προγραμματιστή να δημιουργεί, να τρέχει και να κλιμακώνει εφαρμογές με παρόμοιο τρόπο σε όλες τις γλώσσες. Η Heroku αποκτήθηκε από την Salesforce.com το 2010 για 212 εκατομμύρια δολάρια.

Οι εφαρμογές που εκτελούνται στο Heroku έχουν συνήθως ένα μοναδικό τομέα (συνήθως "applicationname.herokuapp.com") που χρησιμοποιείται για τη δρομολόγηση των αιτήσεων HTTP στο σωστό dyno. Κάθε ένα από τα δοχεία εφαρμογών ή dynos, είναι διασκορπισμένα σε ένα "πλέγμα dyno" που αποτελείται από διάφορους εξυπηρετητές. Ο διακομιστής Git της Heroku χειρίζεται το χώρο αποθήκευσης εφαρμογών που προωθείται από τους επιτρεπόμενους χρήστες. Όλες οι υπηρεσίες της Heroku φιλοξενούνται στην πλατφόρμα cloud-computing EC2 του Amazon [18].

4.3. MongoDb mlab

Η MongoDb δίνει την δυνατότητα στους χρήστες της να νοικίασουν δικούς της Server για την βάση δεδομένων τους αντί για τη δική τους συσκευή και άμα είναι κάτω από 512mb είναι δωρεάν. Με τη δημιουργία ενός λογαριασμού ο κάθε χρήστης έχει πρόσβαση στη βάση δεδομένων του.

Το mlab παρέχει:

- Κλιμάκωση με μηδενική διακοπή λειτουργίας και υψηλή διαθεσιμότητα
- Προηγμένη ασφάλεια, καθώς έχει ειδικά προγράμματα που υποστηρίζουν την κρυπτογράφηση σε κατάσταση αναπαύσεως, περιλαμβάνουν SSL δωρεάν και επιτρέπουν τη δημιουργία προσαρμοσμένων τείχων προστασίας, καθώς και την ανάκτηση VPC
- Απεριόριστα αντίγραφα ασφαλείας στα ειδικά προγράμματα. Δωρεάν ημερήσια αντίγραφα ασφαλείας σε άλλα σχέδια. Ελεύθερη και εύκολη δημιουργία αντιγράφων ασφαλείας.
- Εύκολο στη χρήση πρόγραμμα περιήγησης δεδομένων
- Web GUI για την επεξεργασία εγγράφων, την εκτέλεση ερωτημάτων (συμπεριλαμβανομένων των αποθηκευμένων αναζητήσεων) και την προβολή των αποτελεσμάτων σε μορφή πίνακα.
- Παρακολούθηση & Εργαλεία Analytics
- Συνεχής παρακολούθηση 24x7 με γραφήματα απόδοσης και προσαρμοσμένη ειδοποίηση.
- Δείκτες ευρετηρίου και απόδοσης που παρέχονται από το Slow Query Analyzer της mLab [19].

4.4. Session

Για την ανάπτυξη του συγκεκριμένου application χρησιμοποιήθηκε αρκετά ένα «default» πακέτο του Meteor, τα Sessions. Τα Sessions παρέχουν ένα global αντικείμενο στον πελάτη (client) που μπορεί να χρησιμοποιηθεί για να αποθηκεύσει ένα αυθαίρετο σύνολο ζευγών κλειδιού-τιμής(key-value). Χρησιμοποιείται για να αποθηκεύει στοιχεία όπως το τρέχον επιλεγμένο στοιχείο σε μια λίστα. Αυτό που είναι ιδιαίτερο για τα Sessions είναι ότι είναι αντιδραστικό(reactive). Εάν καλεστεί το Session.get ('currentList') μέσα από ένα πρότυπο(template), το πρότυπο αυτόματα θα αναδημοσιευτεί, όποτε το Session.set ('currentList', x) καλείται.

4.5. GroundDB

Όπως προαναφέρθηκε, για τη λειτουργία της εφαρμογής χρησιμποιείται μια online βάση δεδομένων. Εδώ είναι σημαντικό να σημιεωθεί πως το Meteor δεν επιτρέπει στον πελάτη να κάνει αλλαγές στη βάση δεδομένων παρά μόνο αιτήματα στον Server για να την επεξεργαστεί αυτός, με αποτέλεσμα οι πολλές συχνες μικρές αλλαγές που τυχόν θα χρειάζονται στη βάση να επιβαρύνουν παραπάνω από όσο χρειάζεται τον διακομιστή. Για να μειωθεί όσο γίνεται το «traffic» αλλά και για ευκολότερη ανάπτυξη της εφαρμογής χρησιμοποιήθηκε το πακέτο Groundb το οποίο δίνει την δυνατότητα στο χρήστη να αποθηκέυει τοπικά ό,τι επιθυμεί, χωρίς να επηρεάζει την κεντρική βάση δεδομένων.

Όταν για παράδειγμα ένας χρήστης δημιουργεί μια καινούργια παραγγελία, θα αποθηκεύει τοπικά αντικείμενο στη συσκευή και όταν ολοκληρώσει την εισαγωγή στοιχείων θα φτιάξει ένα ακριβές αντίγραφό του στην κεντρική βάση. Αυτό λύνει επίσης το πρόβλημα σε περίπτωση αδυναμίας του διακομιστή, καθώς αν ο χρήστης στείλει δεδομένα και δεν «φτάσουν» ποτέ στην κεντρική βάση δεδομένων, τα δεδομένα αυτα θα χαθούν, ενώ τώρα θα παραμείνουν τοπικά αποθηκευμένα μέχρι να σταλούν σωστά [20].

4.6. Πακέτα που χρησιμοποιήθηκαν

- **kadira:flow-router:** Κάνει δρομολόγηση για εφαρμογές από την πλευρά του πελάτη.
- **Check:** Χρησημοποιείται στο διακομιστή για έλεγχο ορισμάτων αν είναι της σωστής μορφής, όπως νούμερο ή string.
- **ultimatejs:tracker-react :** Οποιαδήποτε αντιδραστική πηγή δεδομένων (π.χ.: collection.find () ή Session.get ('foo')) που χρησιμοποιείται στη μέθοδο rendering είναι αυτόματα αντιδραστική.
- **meteortoys:allthings:** Με το Meteor Toys ο προγραμματιστής αποκτά αυτόματα πρόσβαση στη Mongol και το JetSetter, τα οποία ήταν τα αρχικά εργαλεία ανάπτυξης του Meteor.
- **accounts-ui@1.3.0**: Εύκολο στη χρήση, συνδυάζοντας τις ιδέες της διαμόρφωσης των λογαριασμών χρήστη και των λογαριασμών που ο καθένας ήδη γνωρίζει. Τα στοιχεία είναι παντού και επεκτάσιμα αντικαθιστώντας τα στο Accounts.ui.
- **accounts-password:** Μια υπηρεσία σύνδεσης που επιτρέπει την ασφαλή σύνδεση με κωδικό πρόσβασης.
- **blaze-html-templates**: Ένα μετα-πακέτο που περιλαμβάνει όλα όσα χρειάζεται για να μεταγλωττίσει και να εκτελέσει πρότυπα Meteor με το Spacebars και το Blaze.
- **themeteorchef:bert** Είναι ένα client-side, σύστημα πολλαπλών στυλ ειδοποιήσεων για το Meteor.
- **255kb:meteor-status :** Ειδοποιεί αυτόματα τους χρήστες όταν έχει χαθεί η σύνδεση με το διακομιστή. Εμφανίζει επίσης μια αντίστροφη μέτρηση (σε δευτερόλεπτα) έως την επόμενη επανάληψη.
- **smowden:offline-js:** Είναι μια βιβλιοθήκη που ειδοποιεί αυτόματα τους χρήστες, όταν έχουν χάσει τη σύνδεση στο διαδίκτυο, όπως το Gmail.
- **accounts-facebook :** Βιβλιοθήκη για χρήση του facebook λογιαριασμού αντί για τη δημιουργία καινούργιου.
- **momentjs:moment:** Εύρεση της τωρινής ημερομηνίας και ώρας.

4.7. Συναρτήσεις συστήματος

4.7.1. Δημιουργία μενού

Όπως προαναφέρθηκε, στη συγκεκριμένη εφαρμογή χρησιμοποιήθηκε μια NoSQL βάση δεδομένων, η MongoDB, η οποία αντί για tables χρησιμοποιεί αρχεία Json χωρισμένα σε κατηγορίες - Collections. Για την καλύτερη διαχείριση της βάσης αποφασίστηκε το μενού να χρησιμοποιεί ένα ξεχωριστό Collection, κάθε αρχείο - Document σε αυτό το collection είναι ένα ξεχωριστό αντικείμενο στο μενού. Είναι ένα json αρχείο της μορφής

Εικόνα 4.1 Μορφή json αρχείου του μενού

Όπου:

1. "_id" το όνομα του αντικειμένου
2. "category" σε ποια κατηγορία ανήκει
3. "price" τιμή
4. "Ingredients" συστατικά
5. "complete" παράμετρος για προσωρινή αφαίρεση από το μενού
6. "createdAt" ημερομηνία δημιουργίας
7. "user" απο ποιον δημιουργήθηκε
8. "shop" μαγαζί που ανήκει

Το Collection του μενού εμφανίζεται χωρισμένο στις κατηγορίες του. Αυτό γίνεται με τα Query της MongoDB, όπου αρχικά βρίσκει όλες τις κατηγορίες και τις αποθηκεύει σε ένα πίνακα με όνομα "distinctEntries".

```
mycategories() {
    var distinctEntries = _.uniq(Mcollections.find({}, {
        sort: {"category": 1}, fields: {"category": true}
    }).fetch().map(function(x) {
        return x.category;
    }), true);
    return distinctEntries;
}
```

Εικόνα 4.2 Παράδειγμα κώδικα για διαχωρισμό κατηγοριών

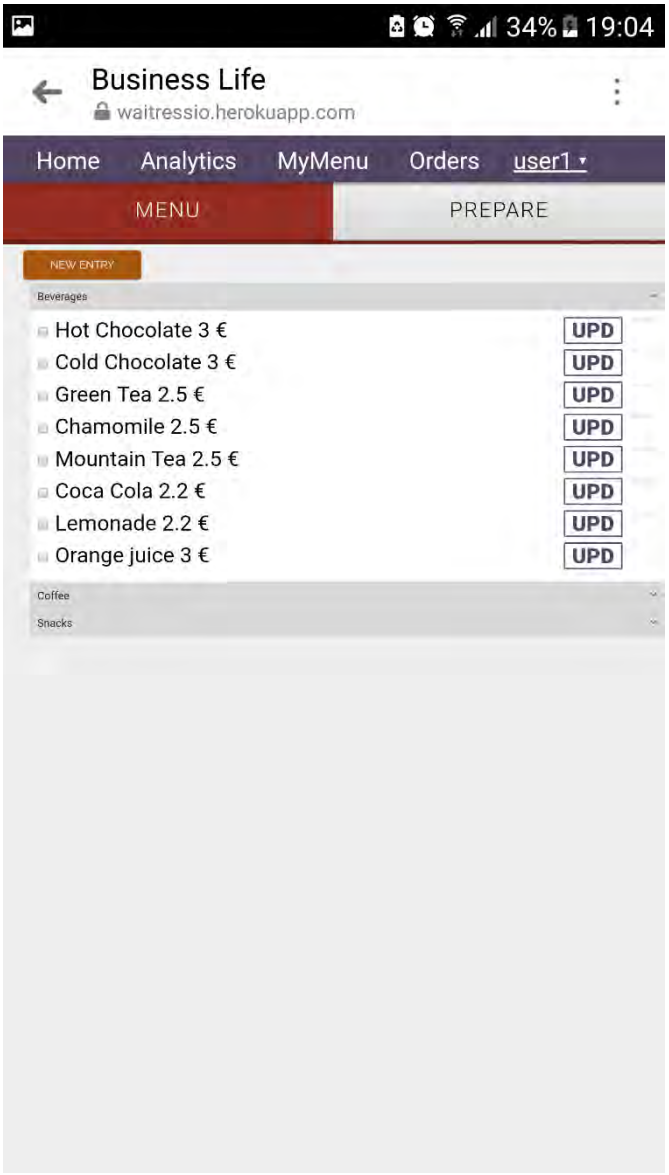
Στη συνέχεια, με τη χρήση της συνάρτησης map η οποία έχει σαν όρισμα μια λίστα από αντικείμενα, στην προκειμένη περίπτωση λίστα από String, «καλεί» επανειλημμένα όσο και ο αριθμός των κατηγοριών μία άλλη συνάρτηση map με αριθμό επαναλήψεων όσα και τα αντικείμενα σε κάθε κατηγορία και κάθε φορά εμφανίζει ένα JSX αρχείο με τις απαραίτητες πληροφορίες για το καθένα.

```
const category_names = this.mycategories();
const listItems = category_names.map((category_name, index) =>
    <Collapsible key={index} className="resolutions" trigger={`${category_name}`}>
    {this.eachcategory(category_name).map( (resolution)=>{
        return <ResolutionSingle key={resolution._id} resolution={resolution} />
            })
    }
    </Collapsible>
);
```

Εικόνα 4.3 Κώδικας για την εμφάνιση του μενού ανά κατηγορίες

Τελικά το ResolutionSingle.jsx αρχείο καλείται τόσες φορές όσες και τα συνολικά σε αριθμό αντικείμενα σε σειρά, ανάλογα τις κατηγορίες. Στο ResolutionSingle.jsx γίνεται εμφάνιση του καθενός αντικειμένου ξεχωριστά και δίνει τη δυνατότητα στο χρήστη να σημειώσει ένα αντικείμενο ως μη διαθέσιμο, να αλλάξει το όνομά του και να το διαγράψει αν επιθυμεί.

Ο χρήστης έχει την παρακάτω συνολική εικόνα:



Εικόνα 4.4 Σελίδα Εφαρμογής για επεξεργασία μενού

4.7.2. Παραγγελιοληψία

Το σύστημα παραγγελιοληψίας χρησιμοποιεί και αυτό ξεχωριστό Collection στη βάση δεδομένων και αποθηκεύει την κάθε παραγγελία με τις εξής πληροφορίες:

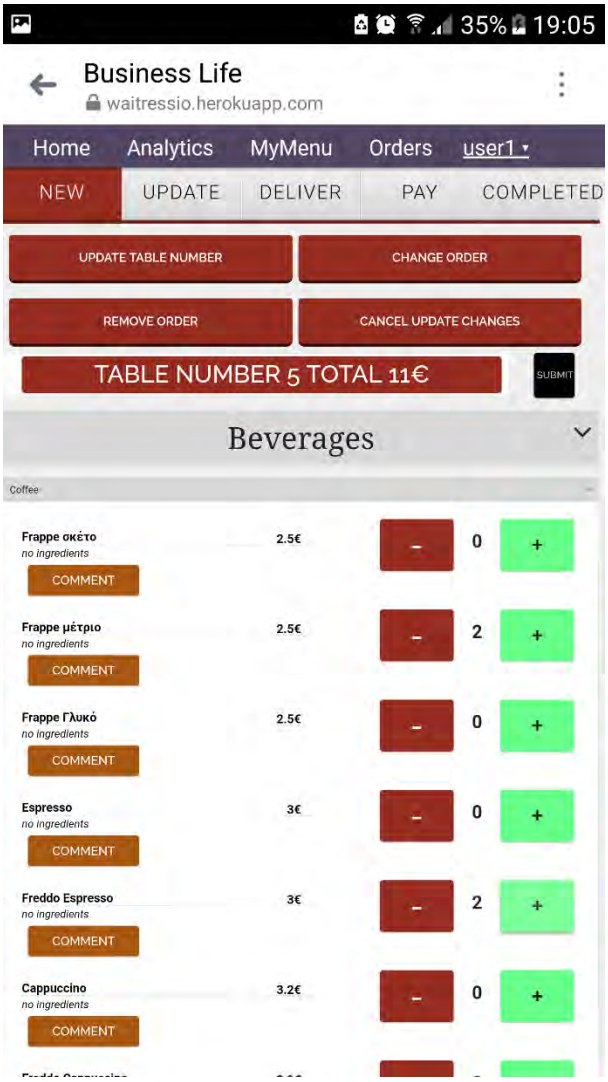
Εικόνα 4.5 Json μορφή για κάθε παραγγελία

Κάθε αντικείμενο που προστίθεται στην παραγγελία, προστίθεται σε ξεχωριστή θέση στη λίστα με τα αντικείμενα, ανεξάρτητα από κοίνα στοιχεία. Αυτό γίνεται για καλύτερη διαχείριση της βάσης, ειδικά στο κομμάτι της παράδοσης και πληρωμής. Το κάθε αντικείμενο αποθηκεύεται ως εξής:

Εικόνα 4.6 Json μορφή για κάθε αντικείμενο παραγγελίας

Κατά τη δημιουργία καινούργιας παραγγελίας, ο χρήστης αρχικά διαλέγει το τραπέζι με βάση τον αριθμό του για να προχωρήσει. Μέσω της HTML συνάρτησης επίλεγει ανάμεσα από το ένα εώς τον αριθμό τραπεζιών πού έβαλε ο διαχειριστής του μαγαζιού κατά τη δημιουργία του, εκτός των τραπεζίων που έχουν ήδη παραγγελία σε αναμονή. Αυτό γίνεται με τη βοήθεια ενός Query της MongoDB που ελέγχει τα πεδία "table_number", όπου εάν βρει αριθμό τραπεζίου στο Collection των παραγγελιών με πεδία status και paid αληθή, το αφαιρεί από τις επιλογές μέχρι να ολοκληρωθεί.

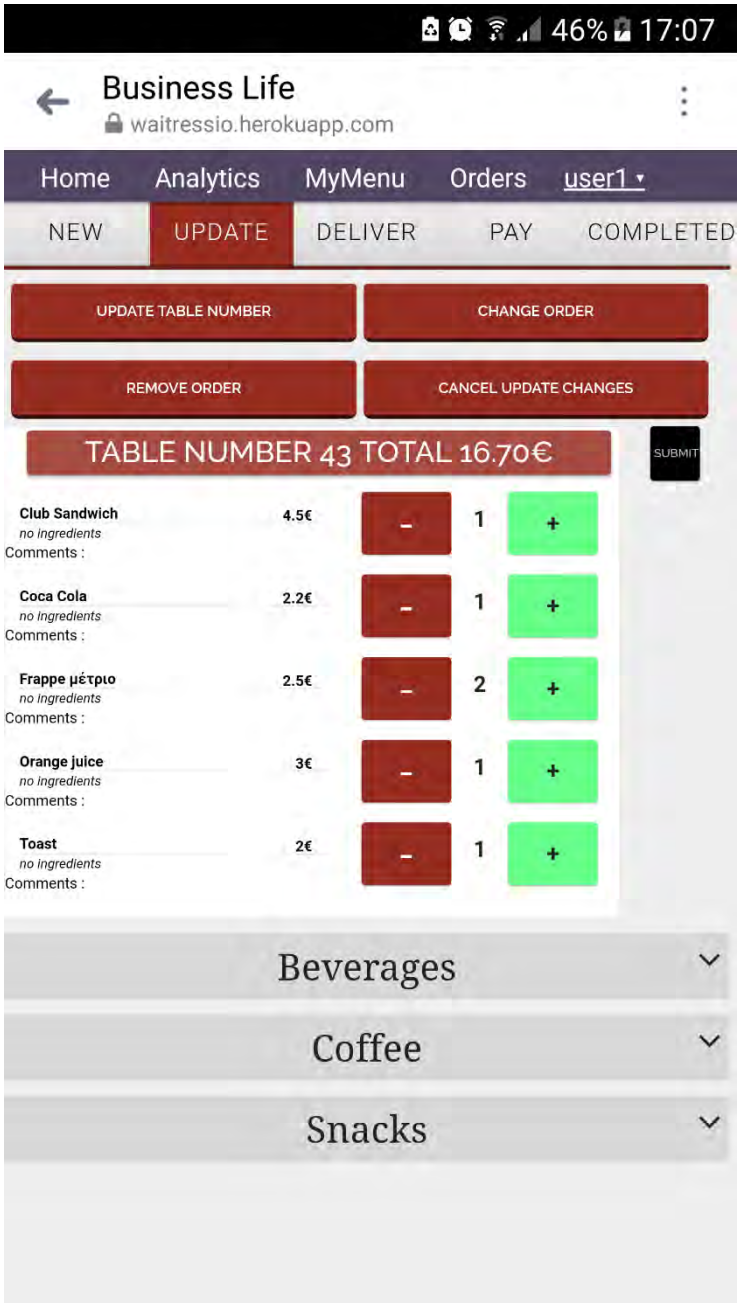
Η εμφάνιση του μενού για την προσθαφαίρεση αντικειμένων γίνεται πάλι με τη χρήση της συνάρτησης map, απλά σε αυτή την περίπτωση καλείται η OrderResolutionSingle.jsx, όπου αντί για τις λειτουργίες update kai delete, έχουμε την επιλογή για εισαγωγή σχολίου και τα αντίστοιχα plus-minus κουμπία.



Εικόνα 4.7 Σελίδα εφαρμογής για παραγγελιοληψία

Κάθε φορά που ο χρήστης προσθέσει ή αφαιρέσει ένα αντικείμενο, αλλάζει η λίστα «items» της τοπικής βάσης δεδομένων, γιατί όπως προαναφέρθηκε, η κεντρική βάση του μαγαζιού θα επηρεαστεί μόνο μετά τη χρήση του κουμπιού submit στην οποία θα αποθηκευτεί ο κλώνος του αντικειμένου της τοπικής βάσης. Το κουμπί submit θα εμφανιστεί στο χρήστη αφού έχει στην τοπική βάση του τουλάχιστον ένα αντικείμενο.

Οπότε, σε περίπτωση που η παραγγελία δεν είναι άδεια, ο χρήστης έχει την ανάλογη εικόνα:



Εικόνα 4.8 Χρήση λειτουργίας εφαρμογής για εμφάνιση τωρινής παραγγελίας πριν την αποστολή

Αυτό επιτυχνάνεται αρκετά εύκολα με τη χρήση ενός Session που ισούται με το συνολικό κόστος της παραγγελίας, με αποτέλεσμα κάθε φορά που μεταβάλλεται, να ανανεώνονται τα components της σελίδας.

```
            show_listItems=listItems
            select_table=this.select_table(0,[]);
            buttons=this.complete_cancel_change_button(1);
            change_order=buttons[3];
            cancel_button=buttons[1];
            cancel_order=buttons[4];
            change_table_number_button=buttons[2];
            if(Session.get('total')){
                complete_order_button=buttons[0];
            }
        }
```

Εικόνα 4.9 Κώδικας χρήσης Session για εμφάνιση κουμπιού submit και ολοκλήρωση παραγγελίας

Μόλις ο χρήστης θελήσει να στείλει την παραγγελία του, η συνάρτηση AddToOrder θα στείλει ένα αντίγραφο του αντικειμένου στο server, καθώς μόνο αυτός μπορεί να επεξεργαστεί τη βάση και να αρχικοποιήσει τα πάντα από την αρχή για τη δημιουργία καινούργιας.

```
addToOrder(event){
        Meteor.call('addOrder', LocalOrder.find().fetch()[0]);
        LocalOrder.clear();
        Session.set('table_number','');
        this.setState({ selectedOption: ''});
        Session.set('total',0);
}
```

Εικόνα 4.10 Κώδικας Συνάρτησης αποστολής παραγγελίας στην κεντρική βάση δεδομένων

Η ίδια νοοτροπία επικρατεί και όταν ο χρήστης επιχειρεί να ανανεώσει υπάρχουσα παραγγελία, το μόνο που αλλάζει είναι πως τώρα, η επιλογή των διαθέσιμων τραπεζιών είναι αντίστροφη, καθώς θα εμφανίζει όλα τα τραπέζια που έχουν έστω ένα από τα πεδία paid και status με τιμή false ή 0. Με το που γίνει η επιλογή θα πρέπει να αποθηκευτεί στην τοπική βάση η επιλεγμένη παραγγελία, που βρίσκεται στην online κεντρική βάση ώστε να σημειωθούν οι επιθυμητές

αλλαγές και να ξανασταλεί στο server, ο οποίος θα διαγράψει την προηγούμενη και θα προσθέσει την καινούργια.

```
if(this.props.nav_option==="update"){
    LocalOrder.insert(currOrder.find({"table_number" : Session.get('table_number'),
     "completed" : false }).fetch()[0]);
    Session.set('total',currOrder.find({"table_number" : Session.get('table_number'),
      "completed" : false}).fetch()[0].total);
}
```

Εικόνα 4.11 Κώδικας ελέγχου λειτουργίας για ανανέωση παραγγελίας ή δημιουργία καινούργιας

4.7.3. Διαχείριση Παραγγελιών

Οι παραγγελίες είναι αποθηκεύμενες στη βάση με την μορφή json, όπως προαναφέρθηκε, υπάρχουν δηλαδή τρεις παράμετροι ready-paid-status που αφορούν την παραγγελία σαν σύνολο και παρόμοιες τρείς παράμετροι για κάθε ένα αντικείμενο στη λίστα “items”. Υπάρχουν τρεις λοιπόν ξεχωριστές σελίδες που ο χρήστης μπορεί να επηρεάσει την κάθε παράμετρο αντίστοιχα. Και οι τρεις σελίδες ακολουθούν το ίδιο σκεπτικό.

Έστω λοιπόν ότι ο χρήστης θέλει να δηλώσει μερικά αντικείμενα ή και όλοκληρες παραγγελίες πληρωμένες. Από τη στιγμή που θα συνδεθεί στην PAY σελίδα θα ζητήσει από το Server να του επιστρέψει σε μορφή λίστας json αντικειμένων όλες τις παραγγελίες που αφορούν το συγκεκριμένο μαγαζί. Αυτό γίνεται εύκολα με τη χρήση της συνάρτησης publish που αναφέρθηκε πριν, η οποία ουσιαστικά επιστρέφει στον client τα δεδομένα που επιθυμεί, αν αρχικά είναι εγγεγραμένος χρήστης και ανήκει σε κάποιο μαγαζί.

```
Meteor.publish("allOrders", function(){
    if(!Meteor.userId()){throw new Meteor.Error('not-authorized');}
    var obj=Meteor.users.find({"_id" : Meteor.userId()}).fetch();
    var shop=obj[0].profile.shop;

    return currOrder.find({shop: shop} );

});
```

Εικόνα 4.12 Κώδικας συνάρτησης που δίνει πρόσβαση στο χρήστη όλες τις παραγγελίες του μαγαζιού που ανήκει

Αφού συλλέξει τα δεδομένα, καλείται το αντίστοιχο JSX αρχείο δύο φορές, με τη διαφορά όμως ότι στο ένα έχει σαν όρισμα paid=”false” ενώ στο άλλο paid=”true”, ώστε να γίνει ένας διαχωρισμός των πληρωμένων και μη παραγγελιών.

```
{this.currOrders().map( (resolution)=>{
        return <PayOrderSingle
        key={resolution._id}
        resolution={resolution}
        paid="false"
        callback={this.updateNow} />
    })}
```

Εικόνα 4.13 Κώδικας κλίσης συνάρτησης για τη λειτουργία πληρωμής της παραγγελίας

Κάθε φορά που καλείται λοιπόν το PayOrderSingle έχει ως παράμετρο όλο το αντικείμενο-Object μιας διαφορετικής παραγγελίας.

Ελέγχει με τη χρήση ενός «if» αν αρχικά δεν είναι πληρωμένη, καθώς βρίσκεται στο κομμάτι που το αφορά να μην είναι πληρωμένη (το καταλαβαίνει από το όρισμα) και φυσικά να μην είναι ολοκληρωμένη-completed (δηλαδή παλιά παραγγελία).

```
if(!this.props.resolution.paid && this.props.paid==="false" && this.props.resolution.completed===false){
```

Εικόνα 4.14 Κώδικας ελέγχου για κατάσταση τωρινής παραγγελίας όσον αφορά την πληρωμή

Από τη στιγμή που είναι θετικό το «if», θα δημιουργηθεί η επιλογή στο χρήστη να δηλώσει την παραγγελία σαν πληρωμένη με τη χρήση ενός «Checkbox», το οποίο αρχικά θα ελέγξει αν η παραγγελία είναι και σερβιρισμένη (δηλαδή status= TRUE), με αποτέλεσμα να τη δηλώσει πλέον σαν ολοκληρωμένη. Στη συνέχεια θα πρέπει να δηλώσει κάθε μη-πληρωμένο αντικείμενο στη λίστα της παραγγελίας ως πληρωμένο από τον τωρινό χρήστη.

```
for(var i=0; i<test[0].items.length; i++){
    if(!test[0].items[i].paid){
        var by=Meteor.users.find({_id : Meteor.userId() }).fetch()[0].username;
        Meteor.call('togglePayOrderItem',
            test[0],test[0].items[i].paid,by,test[0].items[i].name,
            test[0].items[i].itemid,true, function (err, res) {
            if(err){
                console.log(err);
            }else{}
        });
    }
}
```

Εικόνα 4.15 Κώδικας μετατροπής όλων των αντικειμένων μιας παραγγελίας σε πληρωμένα και ανάποδα

Ο χρήστης όμως έχει επίσης τη δυνατότητα να μπορεί να δηλώσει και μεμονωμένα αντικείμενα ως πληρωμένα. Με την χρήση ενός «Label-κουμπιού» εμφανίζονται αναλυτικά τα αντικείμενα της παραγγελίας.

```
        });
    listItems=test.items.map((resolution,index)=>{
            return <PayOrderSingleItem key={index}
            resolution={resolution}
            collection={test}
            callback={this.updateNow} />
    })
}
```

Εικόνα 4.16 Κώδικας κλίσης συνάρτησης για την λειτουργία πληρωμής του κάθε αντικειμένου της παργγελίας

Καλείται πλέον ένα καινούργιο JXS αρχείο επανειλημμένα με τις εξής ιδιότητες:

1. Έλεγχος αν είναι πληρωμένο και σερβιρισμένο με εμφάνιση αντίστοιχου μηνύματος στην οθόνη.

```
const status= this.props.resolution.paid ? <span className="completed_paid"
>paid to {this.props.resolution.paidby}</span> : '';
const status2= this.props.resolution.status ? <span className="completed_checked"
>delivered by {this.props.resolution.statusby}</span> : '';
```

Εικόνα 4.17 Κώδικας εμφάνισης αντίστοιχου μηνύματος για την κατάσταση πληρωμής και παράδοσης κάθε αντικειμένου της παραγγελίας

2. Δυνατότητα αλλαγής κατάστασης της παραμέτρου Paid με τη χρήση ενός “checkbox” με ανάλογη κλήση συνάρτησης του Server.

```
toggleChecked() {
    var by=Meteor.users.find({_id : Meteor.userId() }).fetch()[0].username;
    const test=Meteor.call('togglePayOrderItem',
        this.props.collection,this.props.resolution.paid,by,this.props.resolution.name,
        this.props.resolution.itemid,false, function (err, res) {
        if(err){
            console.log(err);
         }else{
         }
    });
```

Εικόνα 4.18 Κώδικας κλίσης συνάρτησης με τις απαιτούμενες παραμέτρους για αλλαγή κατάστασης πληρωμής ενός αντικειμένου

Παρόμοια βήματα ακολουθούνται και στις περιπτώσεις του status και ready με μικρές διαφορές, όπως για παράδειγμα όταν δηλώνεται μια παραγγελία ως σερβιρισμένη να γίνεται έλεγχος για το αν είναι και πληρωμένη, ώστε να θεωρηθεί ολοκληρωμένη και να μην εμφανίζεται πια για επεξεργασία.

4.7.4. Διαχείριση Μαγαζιού

Η δημιουργία λογαριασμών στο Meteor γίνεται αρκετά εύκολα με τη χρήση ενός έτοιμου πακέτου, οι πληροφορίες όμως που αποθηκεύονται είναι ένα όνομα-email και ένας κωδικός. Αποφασίστηκε να δημιουργηθεί ένα καινούργιο πεδίο profile στο οποίο θα αποθηκεύονται πληροφορίες σχετικά με την τωρινή κατάσταση του κάθε χρήστη για το μαγαζί στο οποίο είναι συνδεδεμένος και δουλεύει. Κατά την είσοδο του χρήστη, αν δεν υπάρχει αυτό το πεδίο, δίνεται η επιλογή στο χρήστη να συνδεθεί σε κάποιο ή να δημιουργήσει ένα καινούργιο.

Κατά τη δημιουργία, θα δημιουργηθεί σε συγκεκριμένο collection της κεντρικής βάσης ένα καινούργιο document της μορφής:

Εικόνα 4.19 Json μορφή του κάθε μαγαζιού

Αυτόματα, ο δημιουργός του μαγαζιού θα συνδεθεί και ως διαχειριστής και θα ενημερωθούν αντίστοιχα και τα πεδία του λογαριασμού του μέσω των μεθόδων:

1.Δημιουργία μαγαζιού

```
Shop.insert({
    shop: shop,
    tables: tables,
    owner: name,
    owner_id : Meteor.userId(),
    manager : [{
        name : name,
        name_id : Meteor.userId()
    }]
});
```

Εικόνα 4.20 Κώδικας δημιουργίας ενός καινούργιου μαγαζιού

2.Αποθήκευση του χρήστη στο μαγαζί

```
var user_id=obj[0]._id;
if(!Meteor.userId()){
    throw new Meteor.Error('not-authorized');
}
Shop.update({"shop" : shop},
    {$push : { [position] : {
            name:name,
            name_id : user_id}
}},{upsert:true});
Shop.update({"shop" : shop},
    {$pull : { requests : {name:name}} } );
```

Εικόνα 4.21 Κώδικας επεξεργασίας στοιχείων μαγαζιού κατά την είσοδο ή έξοδο ενός χρήστη στο μαγαζί

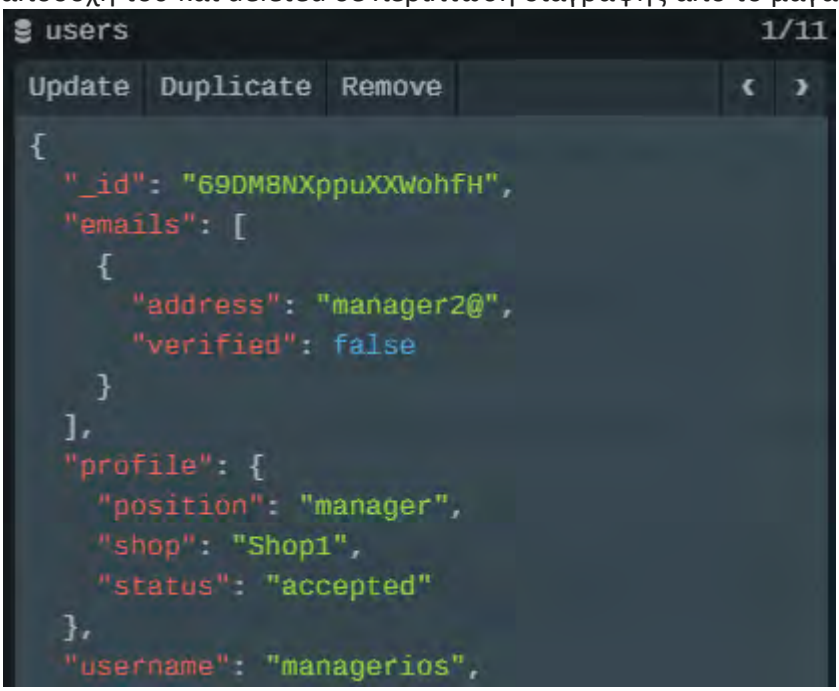
3.Ενημέρωση document λογαριασμού χρήστη

```
register(name,shop,position){
    var obj=Meteor.users.find({"username" : name}).fetch();
    Meteor.users.update(obj[0]._id,
        {$set : {"username": name,"profile.shop":shop,
        "profile.position":position,"profile.status":"accepted"}
        ,{upsert:true});
},
```

Εικόνα 4.22 Κώδικας επεξεργασίας στοιχείων χρήστη κατά την είσοδό του στο μαγαζί

Αν ένας χρήστης επιθυμεί να συνδεθεί σε κάποιο μαγαζί, ενημερώνεται το αρχείο του λογαριασμού του, αλλά και του μαγαζιού, με τις ανάλογες πληροφορίες.

1. του χρήστη θα ενημερωθεί το πεδίο profile με το όνομα του μαγαζίου που επιθυμεί, τη θέση αλλά και την κατάστασή του, δηλαδή pending κατά την αίτηση του, accepted κατά την αποδοχή του και deleted σε περίπτωση διαγραφής απο το μαγαζί.



Εικόνα 4.23 Json μορφή του κάθε χρήστη

2. Του μαγαζίου με εισαγωγή καινούργιο sub-document στο πεδίο με τα request του, ώστε να ενημερωθούν οι διαχείριστες για την επιθυμία κάποιου χρήστη για σύνδεση στο μαγαζί

Εικόνα 4.24 Json μορφή των αιτημάτων από το χρήστη στο μαγαζί

Και ανάλογα με την επιλογή τους, αν γίνει αποδεκτός θα γίνει διαγραφή του αντικειμένου από τα requests και εισαγωγή του στην αντίστοιχη θέση δαιφορετικά θα γίνει απλά διαγραφή του αιτήματος με παράλληλη ενημέρωση του «profile.status» του χρήστη με τη χρήση των μεθόδων που αναφέρθηκαν πιο πάνω.

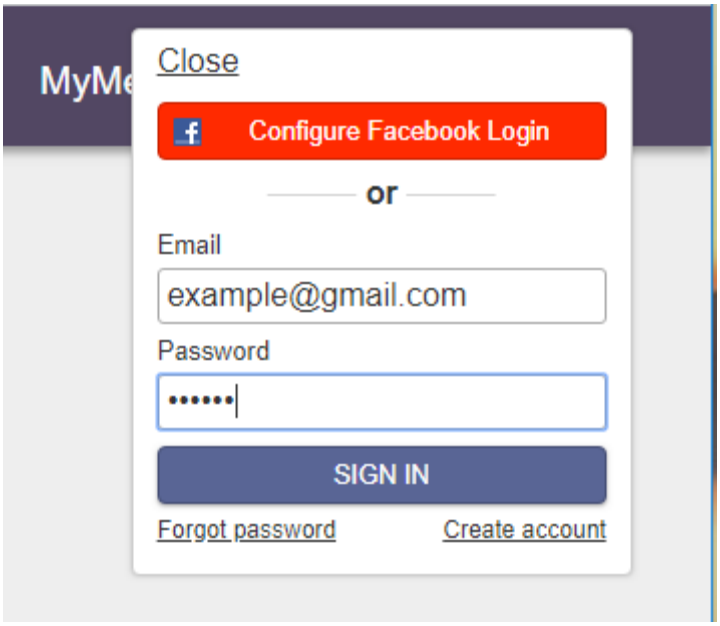
5. Παρουσίαση Εφαρμογής - Εγχειρίδιο

5.2. Είσοδος χρήστη

Κατά την είσοδο του χρήστη στο URL: https://waitressio.herokuapp.com για να αποκτήσει πρόσβαση σε οποιαδήποτε λειτουργία της εφαρμογής, πρέπει αρχικά να συνδεθεί ή να δημιουργήσει ένα καινούργιο λογαριασμό.

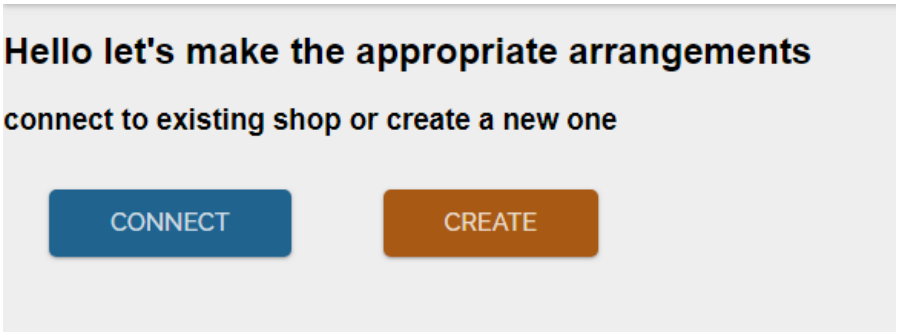
Εικόνα 5.1 Αρχική εικόνα σελίδας

Με τη χρήση ενός email και ενός 6-ψήφιου κωδικού μπορεί να συνδεθεί στην εφαρμογή.



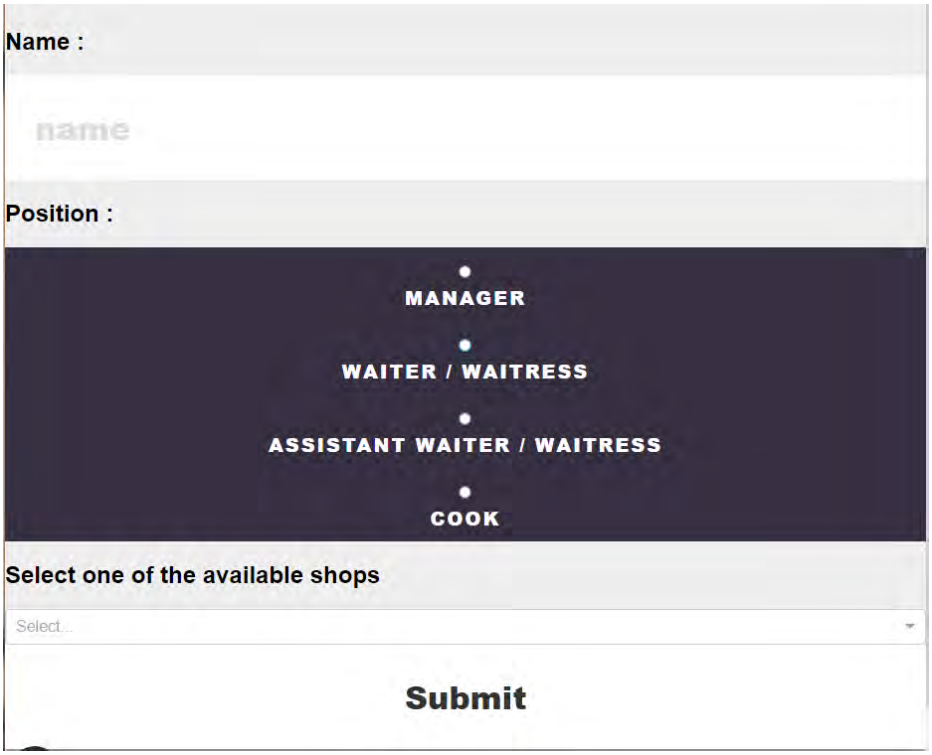
Εικόνα 5.2 Σύνδεση ή δημιουργία λογαριασμού

Στην συνέχεια ο χρήστης έχει τις εξής επιλογές:



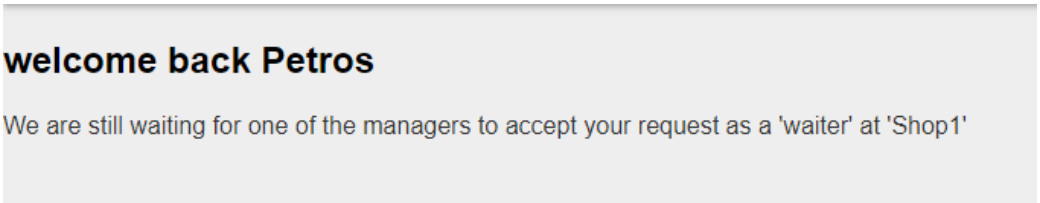
Εικόνα 5.3 Σύνδεση σε μαγαζί ή δημιουργία καινούργιου

- Σύνδεση σε υπάρχον κατάστημα, όπου εμφανίζεται αντίστοιχη φόρμα με τις απαραίτητες ρυθμίσεις:
    1. Όνομα που επιθυμεί να έχει ο χρήστης
    2. Επιλογή ανάμεσα από 4 διαθέσιμες θέσεις
        1) Διαχειριστής – Manager
        2) Σερβιτόρος/α – Waiter / Waitress
        3) Βοηθός σερβοτόρου – Assistant waiter
        4) Μάγειρας – Cook
    3. Επιλογή του μαγαζίου που επιθυμεί να δουλέψει



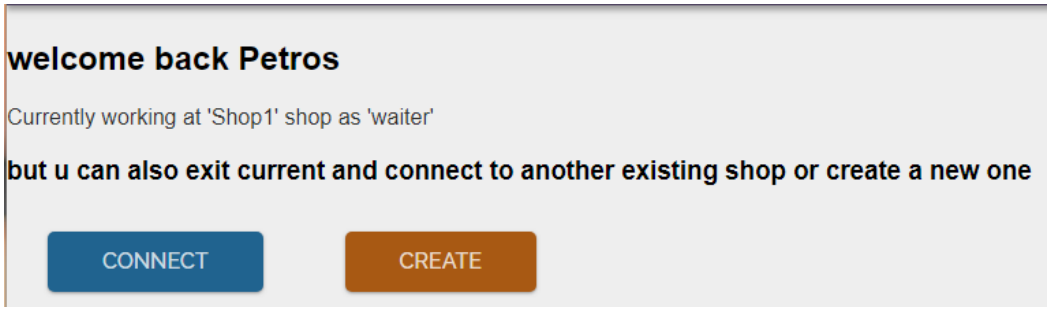
Εικόνα 5.4 Φόρμα για σύνδεση σε μαγαζί

Αφού έχει στείλει την αίτηση, περιμένει την αποδοχή από έναν από τους διαχειριστές του αντίστοιχου μαγαζιού.



Εικόνα 5.5 Ενημέρωση χρήστη για αποστολή αιτήματος στο μαγαζί που επιθυμεί

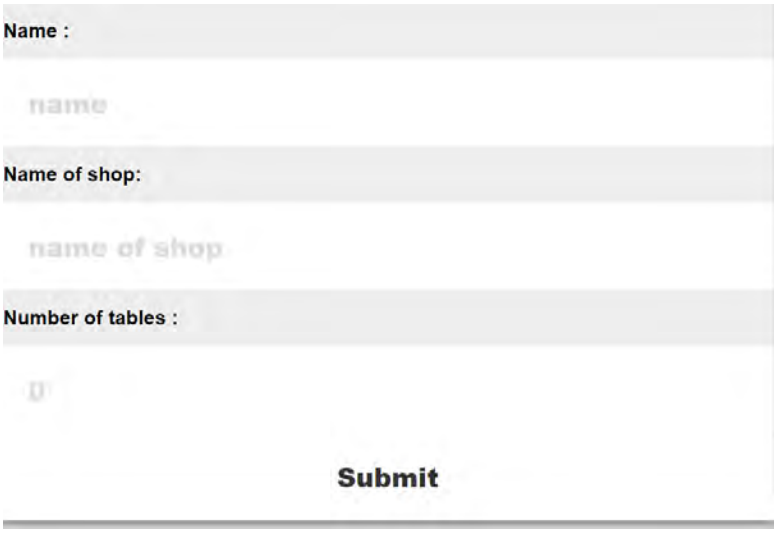
Μόλις ένας από τους διαχειριστές απαντήσει στην αίτηση, θα ενημερωθεί ο χρήστης ανάλογα.



Εικόνα 5.6 Homepage σελίδα και ενημέρωση του χρήστη για την τωρινή κατάσταση του λογαριασμού του

Όπου του δίνεται η δυνατότητα πάλι να αποχωρήσει από το μαγαζί, με την σύνδεσή του σε ένα άλλο ή με τη δημιουργία καινούργιου.

- Κατά την επιλογή του χρήστη για δημιουργία καινούργιου μαγαζιού, του εμφανίζεται η αντίστοιχη φόρμα με την εισαγωγή πληροφοριών όπως :
    1) Όνομα χρήστη
    2) Όνομα μαγαζιού
    3) Αριθμός τραπεζιών



Εικόνα 5.7 Φόρμα για τη δημιουργία μαγαζιού

Στο οποίο θα συνδεθεί και ο ίδιος αυτόματα ως διαχειριστής.

5.3. Δικαιώματα χρηστών

Η κάθε θέση στο μαγαζί έχει τα ανάλογα δικαιώματα. Ο διαχειριστής έχει πρόσβαση σε κάθε λειτουργία που προσφέρει η εφαρμογή :

1) Παραγγελιοληψία
2) Ανανέωση υπάρχουσας παραγγελίας
3) Σημείωση αντικειμένου ή ολόκληρης παραγγελίας ως έτοιμη για παράδοση
4) Σημείωση αντικειμένου ή ολόκληρης παραγγελίας ως παραδοτέα
5) Σημείωση αντικειμένου ή ολόκληρης παραγγελίας ως πληρωμένη
6) Επεξεργασία μενού
7) Πρόσβαση σε αναλυτικές πληροφορίες για το μενού και το προσωπικό
8) Προσβασή στο EventLog (αποθηκεύει κάθε βήμα όπως η λήψη παραγγελίας, από ποιον και πότε)
9) Διαχείριση μαγαζιού στο Home page

Ο σερβιτόρος έχει τη δυνατότητα για :

1) Παραγγελιοληψία
2) Ανανέωση υπάρχουσας παραγγελίας
3) Σημείωση αντικειμένου ή ολόκληρης παραγγελίας ως έτοιμη για παράδοση
4) Σημείωση αντικειμένου ή ολόκληρης παραγγελίας ως παραδοτέα

Ο βοηθός :

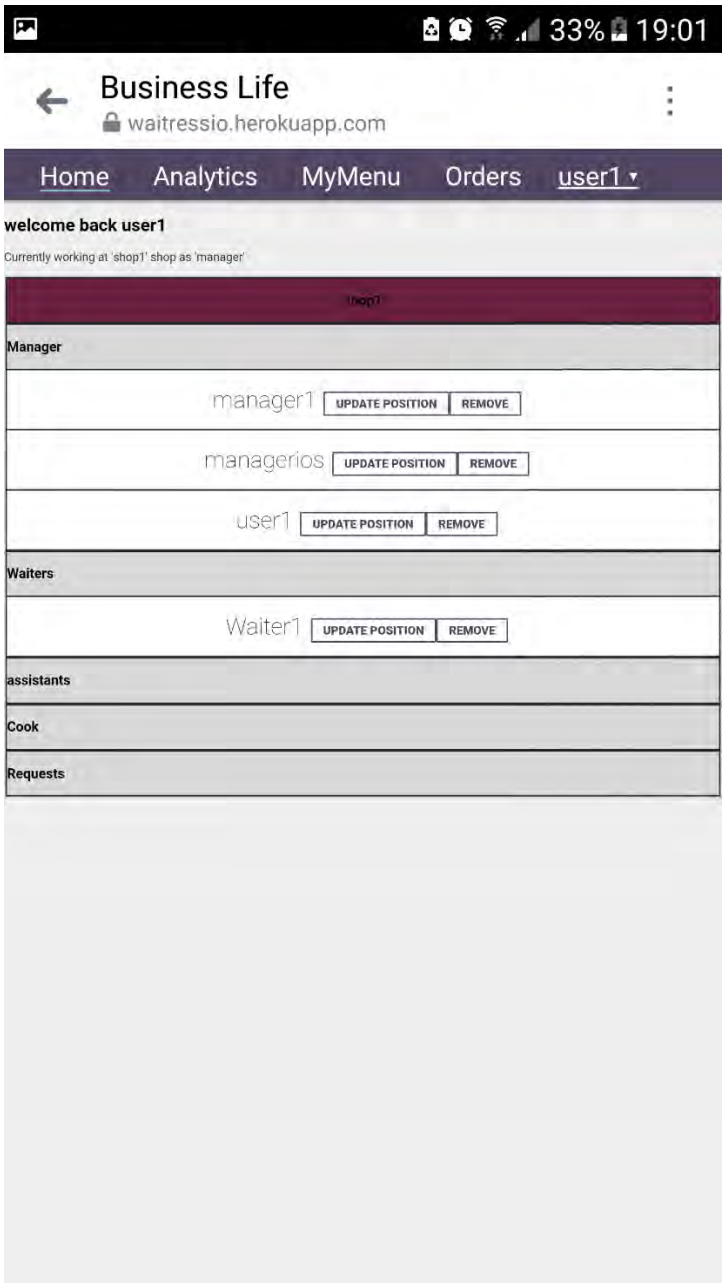
1) Σημείωση αντικειμένου ή ολόκληρης παραγγελίας ως έτοιμη για παράδοση
2) Σημείωση αντικειμένου ή ολόκληρης παραγγελίας ως παραδοτέα

Ο Μάγειρας :

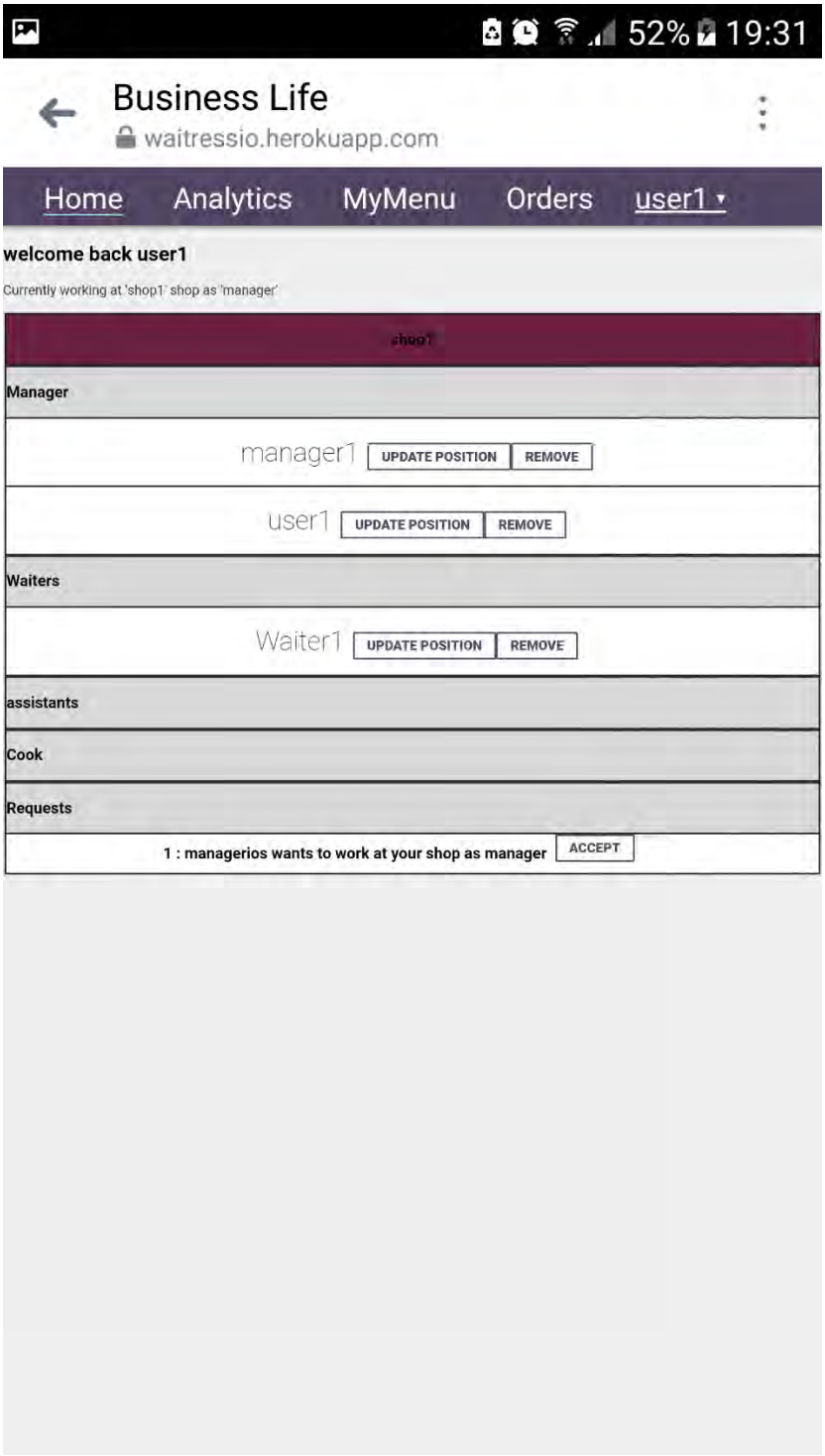
1) Σημείωση αντικειμένου ή ολόκληρης παραγγελίας ως έτοιμη για παράδοση
2) Επεξεργασία μενού

5.4. Home page

Στο Home page ο κάθε χρήστης δεν έχει και πολλές λειτουργίες να αξιοποιήσει μετά την είσοδό του σε κάποιο μαγαζί, πέρα απο την έξοδό του από αυτό, εκτός εάν είναι διαχειριστής, όπου στη συγκεκριμένη περιπτωσή, έχει μια πλήρη είκονα του προσωπικού που είναι συνδεδεμένο στο μαγαζί του, σε ποια θέση είναι ο καθένας αλλά και με δικαιώματα διαγραφής του αντίστοιχου ατόμου ή ανανέωση της θέσης του, για παράδειγμα, από βοηθός σερβιτόρου σε σερβιτόρο.



Εικόνα 5.8 Homepage σελίδα για διαχείριση του μαγαζιού



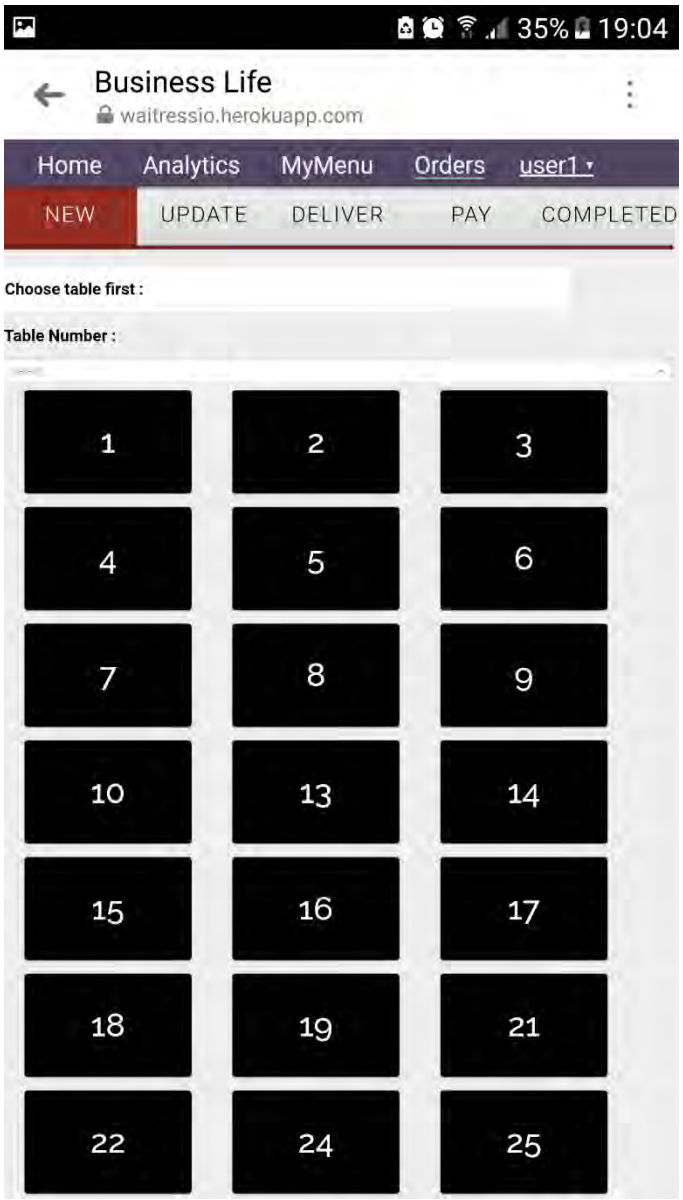
Εικόνα 5.9 Home page για τους διαχειριστές μαγαζιών με αίτημα πρόσβασης από άλλο χρήστη

Αν τώρα υπάρχουν αιτήματα(requests) μπορεί να τα αποδεχθεί ή να τα απορρίψει.

5.5. Παραγγελίες (Orders)page

5.5.1. New - Update

Κατά την είσοδο ενός σερβιτόρου ή διαχειριστή στη συγκεκριμένη σελίδα του εμφανίζεται η παρακάτω φόρμα:

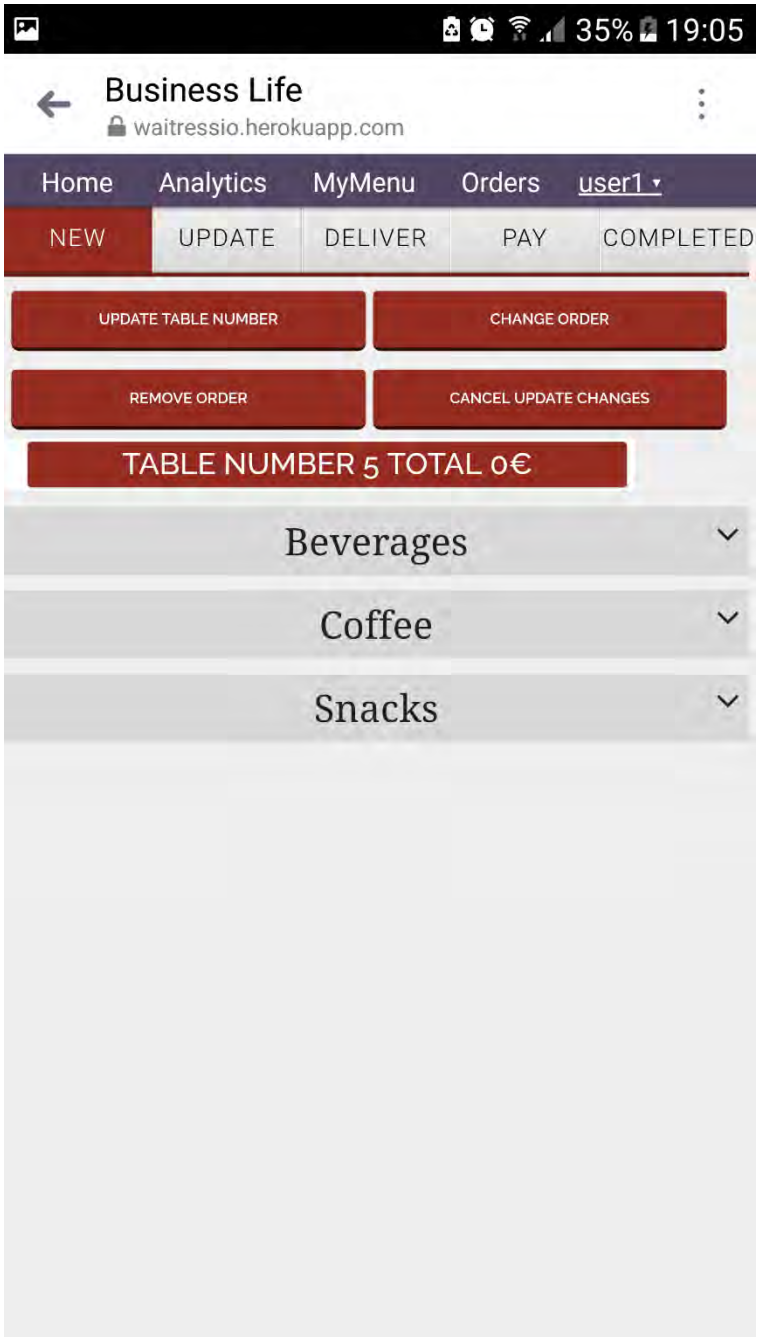


Εικόνα 5.10 Αρχική σελίδα για δημιουργία καινούργιας παραγγελίας - Επιλογή τραπεζιού

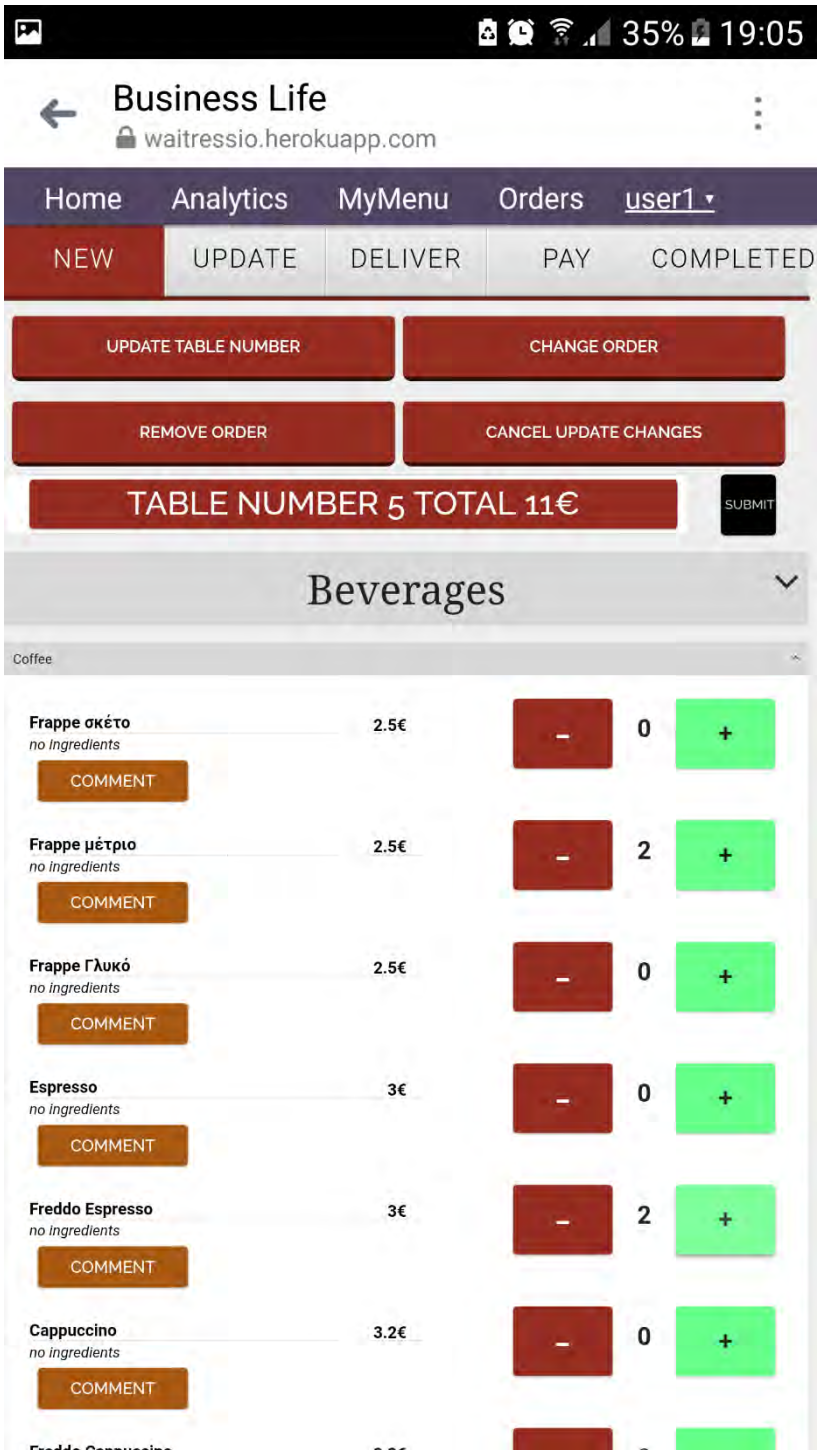
Σε διαφορετική περίπτωση, όπως είσοδο βοηθού, εμφανίζεται αντίστοιχο μήνυμα για μη δυνατότητα πρόσβασης στη συγκεκριμένη λειτουργία λόγω έλλειψης δικαιωμάτων.

Ο χρήστης έχει τη δυνατότητα, αφού επιλέξει αριθμό τραπεζιού να:

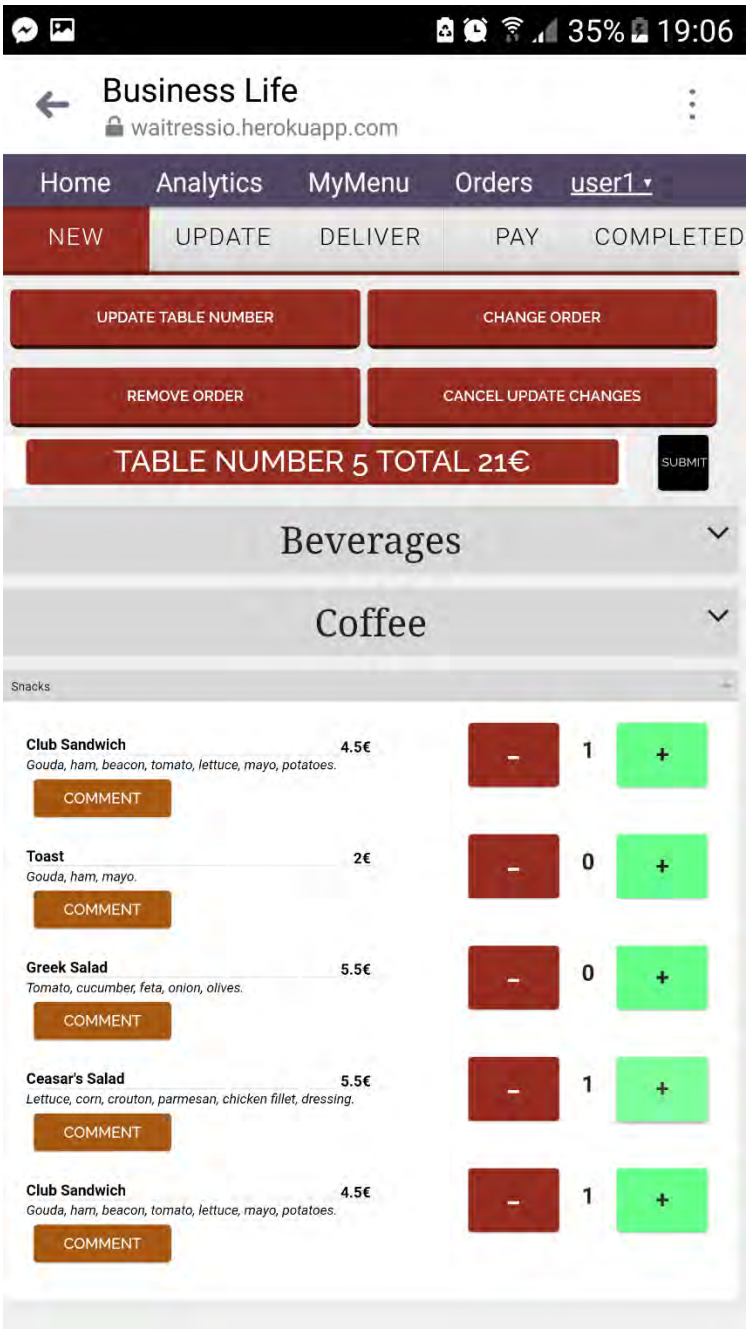
1) Προσθέσει - αφαιρέσει αντικείμενα του μενού στην παραγγελία
2) αλλάξει το αριθμό τραπεζιού
3) να διαγράψει τελείως την παραγγελία
4) να βλέπει ανά πάσα στιγμή το περιεχόμενο της παραγγελίας



Εικόνα 5.11 Εμφάνιση του μενού ανά κατηγορίες για εισαγωγή αντικειμένων στην παραγγελία

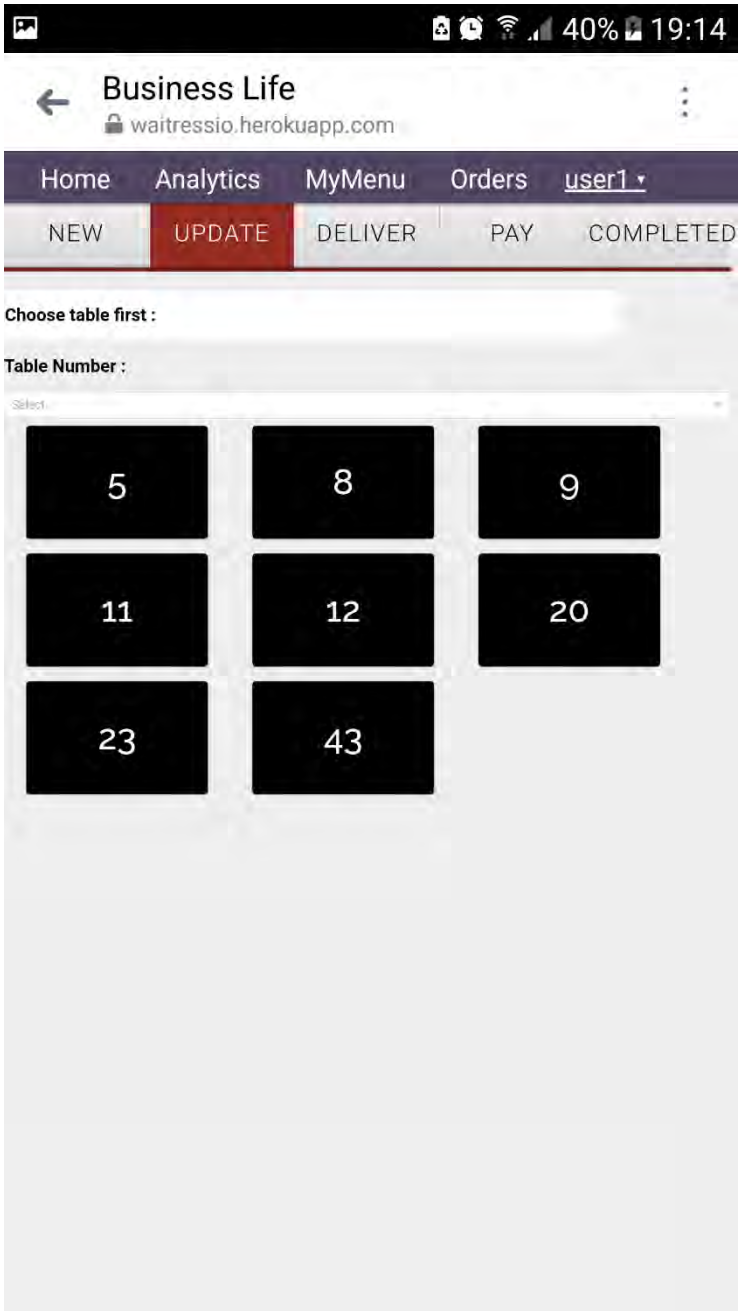


Εικόνα 5.12 Αναλυτική εμφάνιση κατηγορίας του μενού για εισαγωγή αντικειμένων pt1



Εικόνα 5.13 Αναλυτική εμφάνιση κατηγορίας του μενού για εισαγωγή αντικειμένων pt2

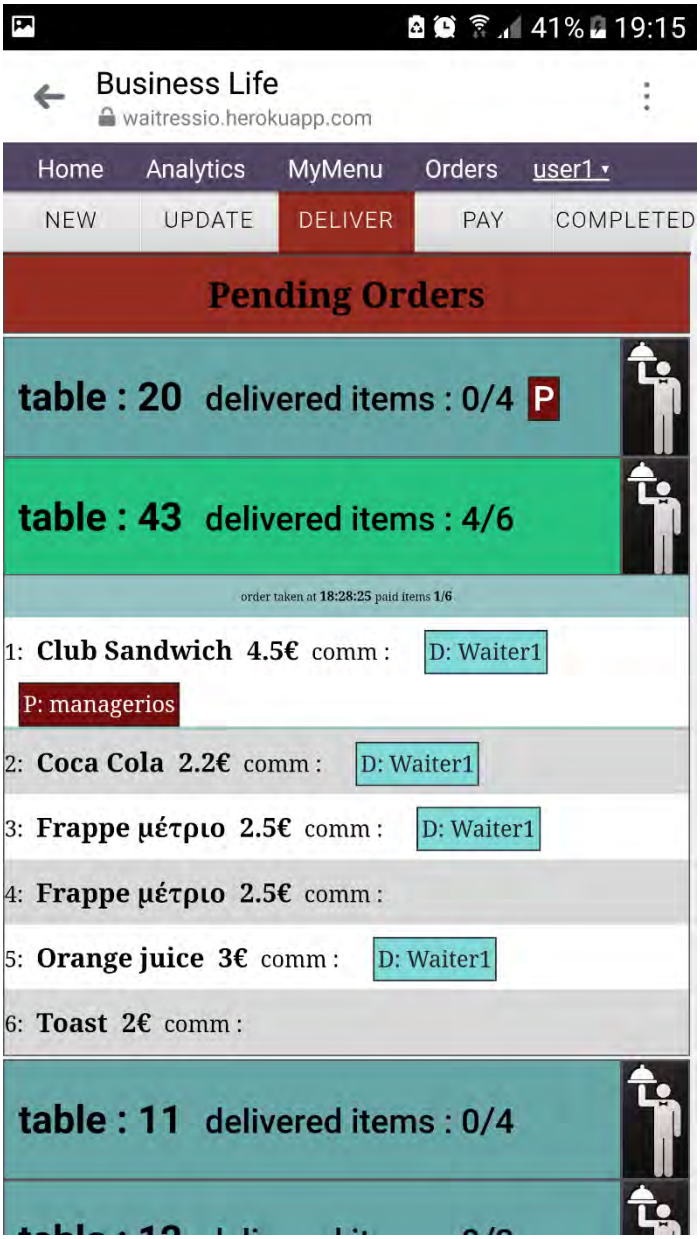
Στην Update σελίδα, ο χρήστης έχει την ίδια εικόνα, αλλά στην αρχική επιλογή του τραπεζιού έχει διαθέσιμα μόνο τα τραπέζια που έχουν μη ολοκληρωμένη παραγγελία (δηλαδή είναι μη πραδοτεά και πληρωμένα).



Εικόνα 5.14 Εμφάνιση τραπεζιών με μη ολοκληρωμένη παργγελία για την επεφεργασία αντικειμένων

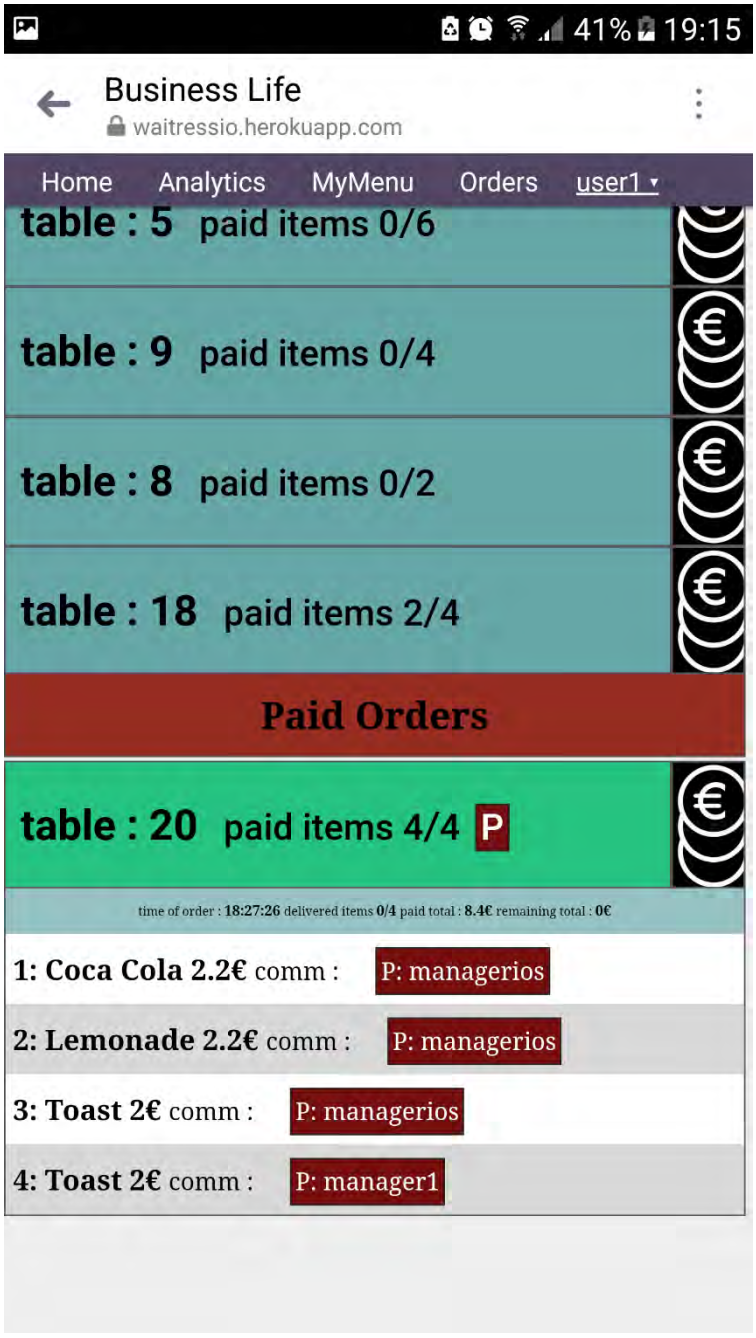
5.5.2. Παράδοση πληρωμή - Deliver pay

Στις Deliver και Pay σελίδες, οι χρήστες σε θέση μάγειρα μπορούν μόνο να δουν τί κατάσταση επικρατεί ανά πάσα στιγμή στα τραπέζια και τις παραγγελίες, χωρίς όμως να έχουν τη δυνατότητα να αλλάξουν τίποτα. Οι χρήστες στις άλλες θέσεις στη σελίδα Deliver έχουν τη δυνατότητα αρχικά να δουν ποια από τα αντικείμενα της κάθε παραγγελίας είναι έτοιμα προς παράδοση, αλλά και να σημειώσουν και οι ίδιοι τα αντικείμενα ή και ολόκληρες τις παραγγελιές που σερβίρουν.



Εικόνα 5.15 Deliver : Εμφάνιση όλων των μη παραδοτέων παραγγελιών

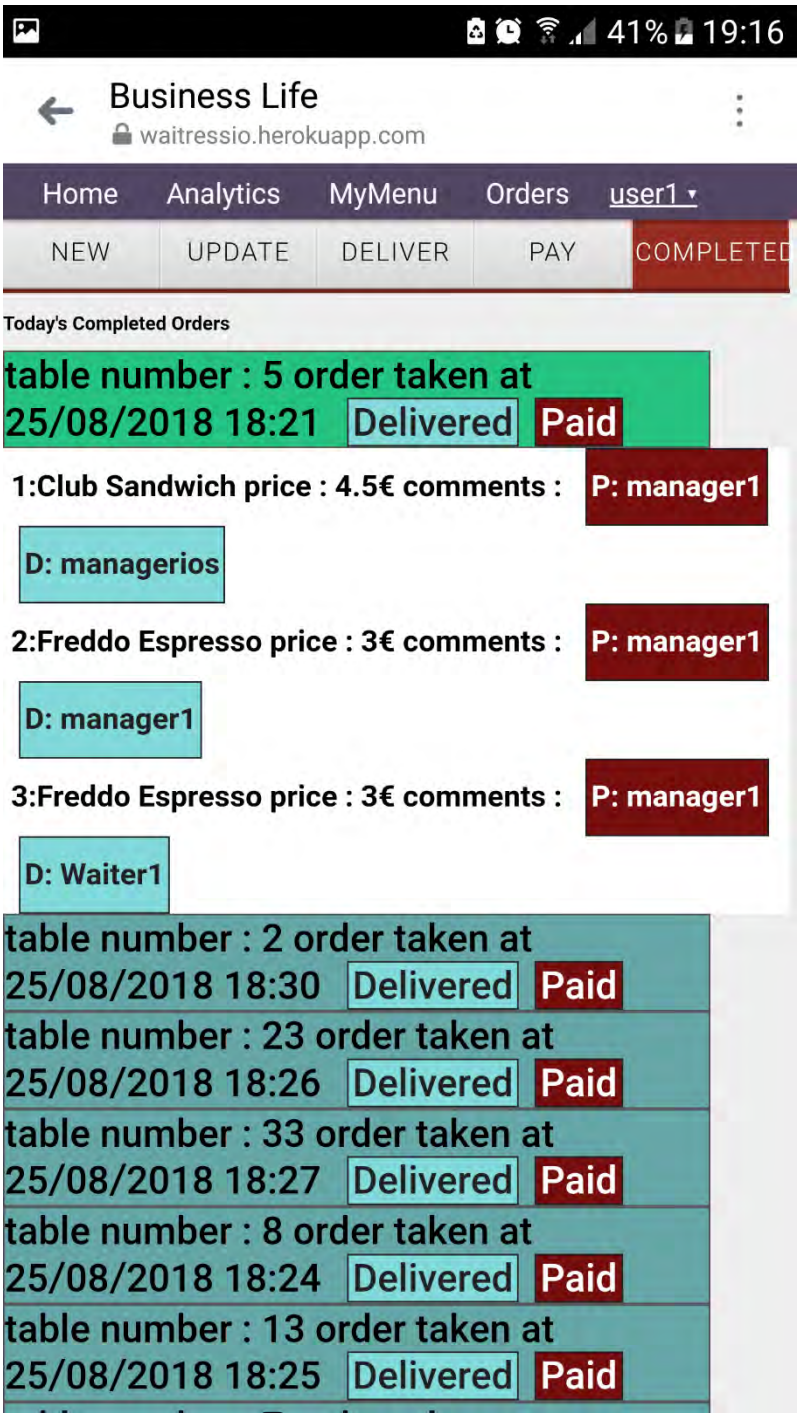
Στην Pay καρτέλα μόνο οι σερβιτόροι και οι διαχειριστές μπορούν να δηλώσουν μια παραγγελία ή αντικείμενα αυτής πληρωμένα, βλέποντας ταυτόχρονα αν έχουν παραδοθεί ολόκληρες οι παραγγελίες.



Εικόνα 5.16 Pay : Εμφάνιση όλων των πληρωμένων παραγγελιών αλλά μη ολοκληρωμένων

Στην Completed καρτέλα εμφανίζονται οι σημερινές ολοκληρωμένες παραγγελίες (σερβιρισμένες και πληρωμένες), σαν αρχείο για το προσωπικό.

Εικόνα 5.17 Completed : Εμφάνιση όλων των σημερινών ολοκληρωμένων παραγγελιών

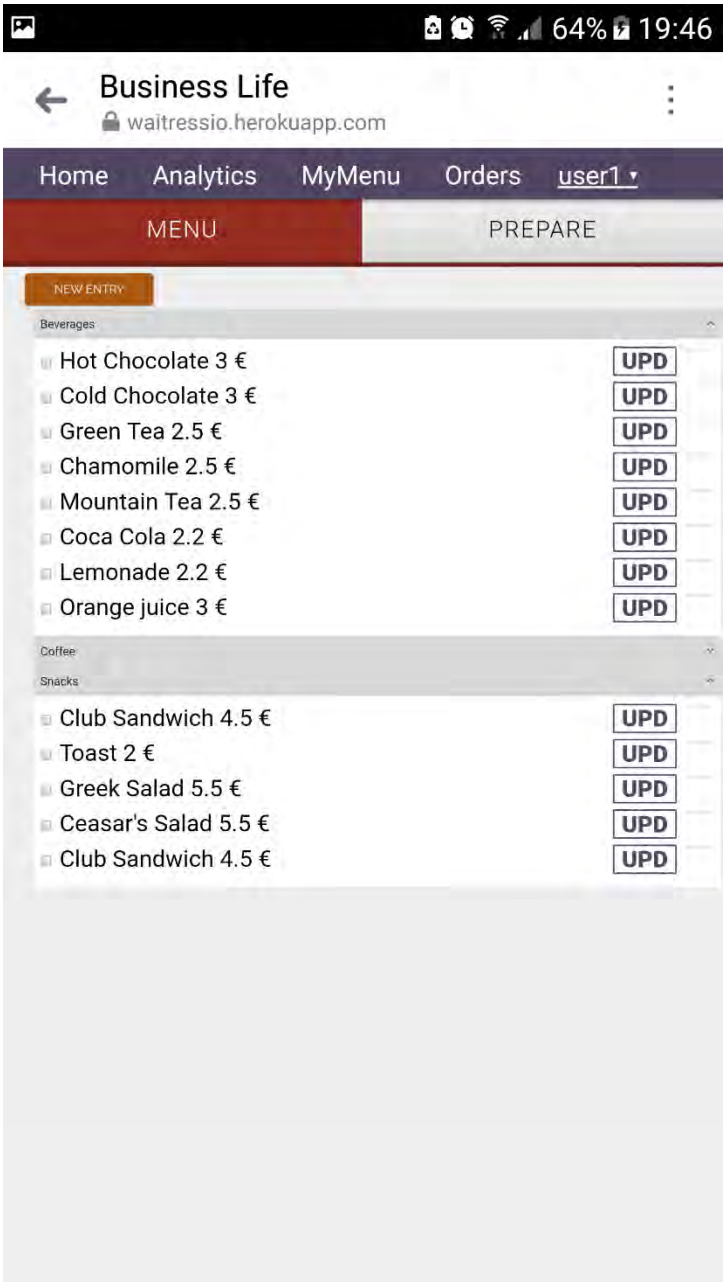


Εικόνα 5.18 Completed : Εμφάνιση όλων των σημερινών ολοκληρωμένων παραγγελιών με τα αντικείμενά τους

5.6. Μενού - MyMenu

5.6.1. Μενού

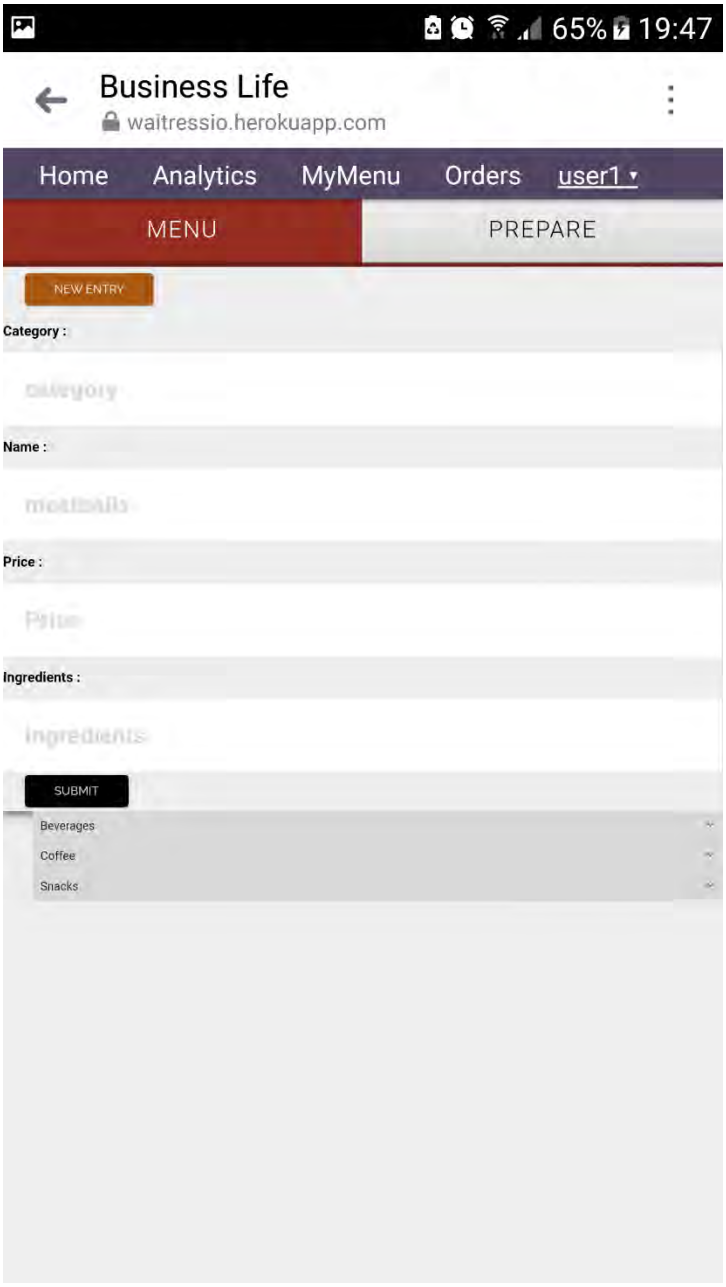
Στη Menu σελίδα, πρόσβαση έχουν μόνο οι διαχειριστές και μάγειρες, στην οποία μπορούν να δουν αναλυτικά το μενού του μαγαζιού, να διαγράψουν ή να ανανεώσουν το όνομα ενός αντικειμένου, αλλά και να βγάλουν προσωρινά εκτός μενού ένα αντικείμενο που τυχόν «τελείωσε», αφαιρώντας έτσι την δυνατότητα σε κάποιον σερβιτόρο να το προσθέσει σε μια παραγγελία.



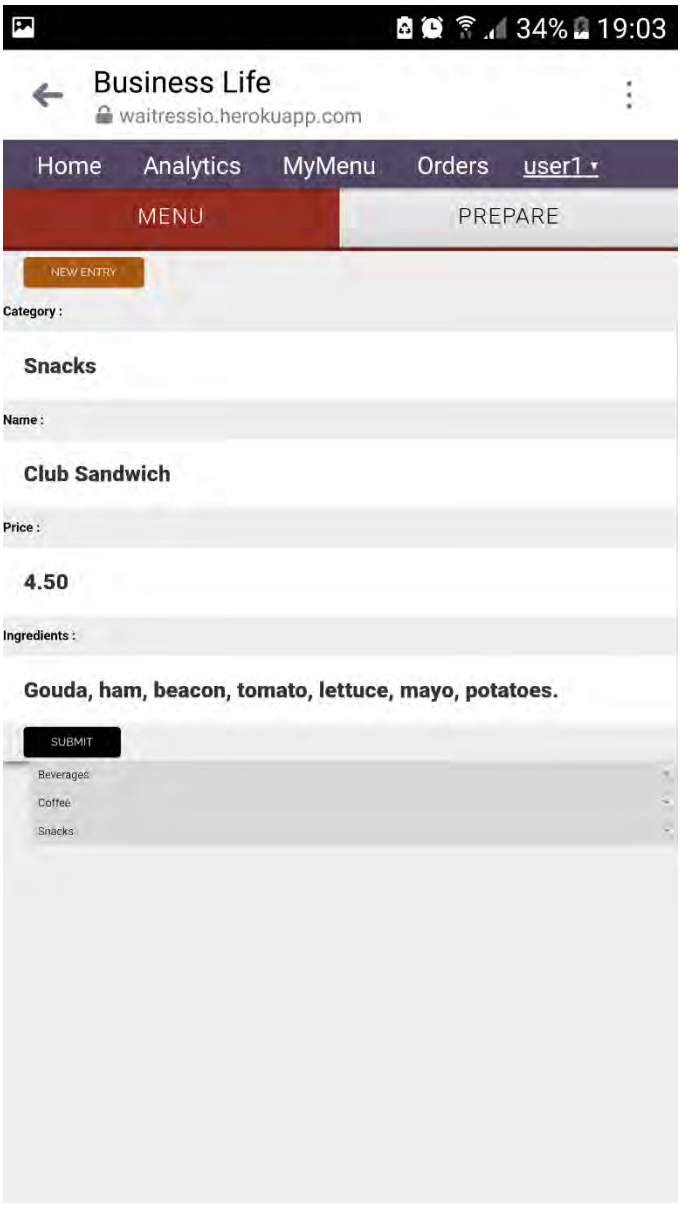
Εικόνα 5.19 Menu : Εμφάνιση και επεξεργασία μενού

Με τη χρήση του κουμπιού New Entry μπορούν να προσθέσουν ένα καινούργιο αντικείμενο στο μενού με στοιχεία :

1) Κατηγορία αντικειμένου
2) Όνομα
3) Τιμή
4) Υλικά



Εικόνα 5.20 Menu : Φόρμα εισαγωγής καινούργιου αντικειμένου στο μενού



Εικόνα 5.21 Menu : Παράδειγμα εισαγωγής καινούργιου αντικειμένου

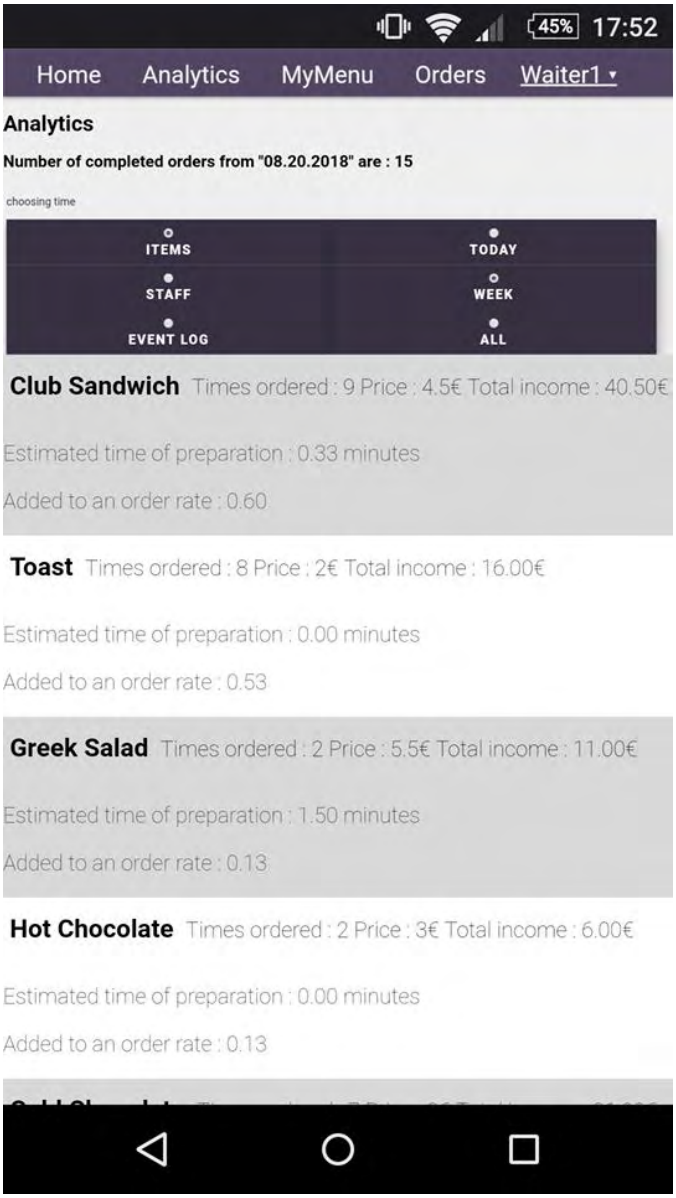
5.6.2. Προετοιμασία - Prepare

Σε αυτή την καρτέλα, όλοι οι χρήστες εκτός των σερβιτόρων μπορούν να δηλώσουν πως ένα αντικείμενο είναι έτοιμο για παράδοση από τα μαγειρία, ώστε να μπορούν να οργανωθούν καλύτερα για το πώς θα διαχειριστούν το σέρβις, ειδικά αν είναι πολλά άτομα προσωπικό.

5.7. Analytics

Η συγκεκριμένη σελίδα αφορά τους διαχειριστές, στην οποία μπορούν αρχικά να διαλέξουν «items» που είναι τα αντικείμενα του μενού τους όπου εμφανίζονται πληροφορίες για το καθένα :

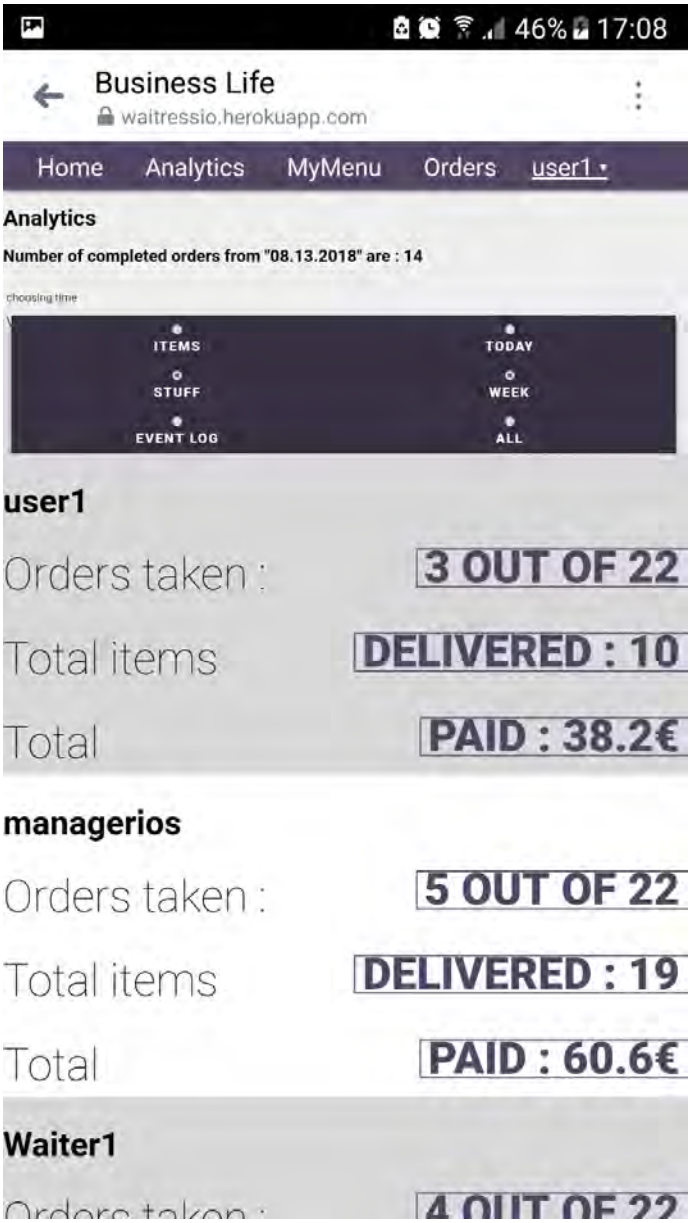
1) Φορές που προστέθηκε σε παραγγελία
2) Η τιμή του
3) Συνολικά έσοδα απο αυτό
4) Εκτιμώμενος χρόνος ετοιμασίας
5) Ποσοστό εμφάνισης αντικειμένου στις ολοκληρωμένες παραγγελίες



Εικόνα 5.22 Analytics: Αναλυτική των αντικειμένων του μενού

Οι διαχειριστές έχουν την επιλογή για εξαγωγή πληροφορίας ανάμεσα απο τρείς διαφορετικές ημερομηνίες, σήμερα, της τελευταίας εβδομάδας και συνολικά απο την αρχή. Σε περίπτωση επιλογής σημερινής ημερομηνίας εμφανίζεται ο συνολικός τζίρος του μαγαζιού μαζι με πληροφορίες για τον αριθμό των ολοκληρωμένων και μη παραγγελιών.

Μπορούν να επιλέξουν να τους εμφανιστεί πληροφορίας για το προσωπικό του μαγαζιού, ώστε να γνωρίζουν πόσες παραγγελίες έχουν εξυπηρετήσει οι ίδιοι, πόσα χρήματα έχει πληρωθεί ο καθένας τους αλλά και την γενικότερη συμμετοχή τους έχοντας αναλυτικές πληροφορίες για το καθένα.



Εικόνα 5.23 Analytics: Αναλυτική του προσωπικού του μαγαζιού

Με τη χρήση του κουμπιού «X out of Y», όπου X ο αριθμός παραγγελιών που ανήκουν στο συγκεκριμένο εργαζόμενο και Y το συνολικό νούμερο παραγγελιών για το μαγαζί, ο διαχειριστής μπορεί να δει αναλυτικά τους χρόνους της κάθε παραγγελίας που δημιούργησε ο εκάστοτε σερβιτόρος.

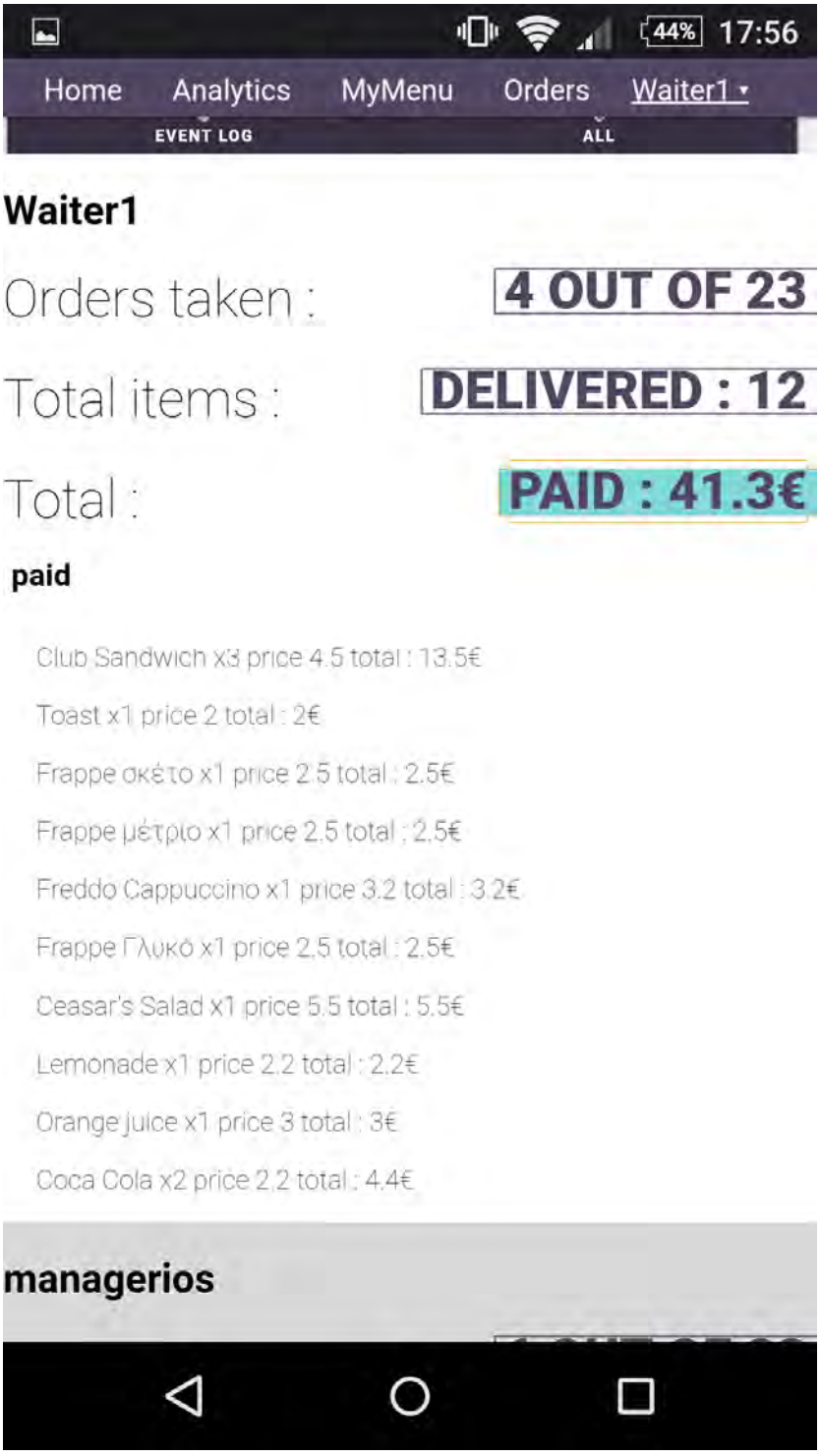
Εικόνα 5.24 Analytics: αναλυτική εμφάνιση των χρόνων κάθε παραγγελίας

Διαφορετικά στο «Delivered : X», όπου X ο συνολικός αριθμός των αντικειμένων που σέρβιρε ο εγραζόμενος, εμφανίζεται λίστα με ό,τι αντικείμενα έχει παραδόσει, μέσος χρόνος παράδοσης ενός αντικειμένου και αναλυτική λίστα με χρόνους παράδοσης του κάθε αντικειμένου σε κάθε τραπέζι.

Εικόνα 5.25 Analytics: εμφάνιση λίστας αντικειμένων παράδοσης σερβιτόρου pt1

Εικόνα 5.26 Analytics: εμφάνιση λίστας αντικειμένων παράδοσης σερβιτόρου pt2

Χρήση του κουμπιού «paid : X», όπου X ο τζίρος που έχει συλλέξει ο εκάστοτε σερβιτόρος, για την εμφάνιση αναλυτικής λίστας των αντικειμένων που έχει πληρωθεί ο συγκεκριμένος σερβιτόρος.



Εικόνα 5.27 Analytics: εμφάνιση λίστας πληρωμένων αντικειμένων από το σερβιτόρο

Τελικά, έχουν και τη δυνατότα επιλογής του Eventlog, το οποίο είναι ένα ξεχωριστό collection της βάσης δεδομένων, που αποθηκεύει κάθε κίνηση του προσωπικού του μαγαζιού μαζί με την ώρα που πραγματοποιήθηκε, για καλύτερη αντιμετώπιση λαθών.

Εικόνα 5.28 Analytics: Εμφάνιση κάθε βήματος του προσωπικού που επεξεργάζεται τη βάση δεδομένων

6. Συμπεράσματα - Επεκτάσεις

6.1. Συμπεράσματα

Κατά την ανάπτυξη της συγκεκριμένης διπλωματικής παρατηρήθηκαν πολλές από τις δυνατότητες του Meteor. Είναι πραγματικά ένα πολυεργαλείο με πάρα πολλές χρήσεις, τόσο για ανάπτυξη εφαρμογών για υπολογιστές αλλά και κινητά. Το documentation τους είναι πάρα πολύ αναλυτικό και περιεκτικό. Έχει αρκέτα tutorials και βοήθεια για τα αρχικά βήματα του καθενός. Χρησιμοποιείται από αρκετό κόσμο, με αποτέλεσμα να υπάρχει ικανοποιητικό υλικό στο ίντερνετ για οποιοδήποτε είδος απορίας. Προσφέρει πρόσβαση σε όλες τις σύχρονες τεχνολογίες για ανάπτυξη εφαρμογών παγκόσμιου ιστού, με νούμερο ένα πλεονέκτημα την αρχιτεκτονική του, που βασίζεται γύρω από τον κόσμο των μη-σχεσιακών βάσεων δεδομένων, της MongoDB που πραγματικά λύνει τα χέρια στη σωστή διαχείριση των περίπλοκων πλέον δεδομένων που απαιτούνται.

Το μόνο μειονέκτημα που ίσως πρέπει να σημειωθεί είναι το «πελάγωμα» που θα αισθανθεί κάποιος κατά τα αρχικά του βήματα στον κόσμο του Meteor και MongoDB, ειδικά αν δε γνωρίζει καλά από προγραμματισμό και Javascript αλλά και NoSQL βάσεων δεδομένων.

Όσον αφορά την συγκεκριμένη εφαρμογή σαν σύστημα παραγγελιοληψίας δεν προσφέρει περισσότερες δυνατότητες σε σχέση με την εφαρμογές που υπάρχουν ήδη στην αγορά, ίσως τη μεμονωμένη παράδοση των αντικειμένων που δεν είναι τοσο διαδεδομένη, αν όμως γίνει σωστή και όσον το δυνατόν περισσότερη αποθήκευση πληροφορίας απο το προσωπικό και τα αντικείμενα, ο διαχειριστής μπορεί να εξάγει χρήσιμες αν όχι κρίσιμες πληροφορίες για την λειτουργία του καταστήματος του ωστε να έχει την μέγιστη δυνατή απόδοση.

6.2. Επεκτάσεις

Με την πάροδο του χρόνου και τη συνέχεια ανάπτυξης της εφαρμογής, μια από τις πρώτες λειτουργίες που θα προστεθεί είναι η χρήση της εφαρμογής από τους πελάτες του κάθε μαγαζιού, και όχι μόνο το προσωπικό. Ενημέρωση του καθενός για το μενού και τα διαθέσιμα τραπέζια, αλλά και online παραγγελία ή και πληρωμή αν επιθυμούν.

Χρήση ίσως ενός προσωπικού chat για το προσωπικό του κάθε μαγαζιού για καλύτερη επικοινωνία και διαχείριση. Χρήση αλγορίθμων Machine Learning για τους διαχειριστές των μαγαζιών ώστε να μπορούν με μεγαλύτερη ευκολία να διακρίνουν την απόδοση του προσωπικου τους αλλά και των επιλογών των μενού τους (ποια πουλάνε καλύτερα και πότε)

Τελικά θα ήταν αρκετά χρήσιμο να προστεθεί η δυνατότητα δημιουργίας του χώρου του κάθε μαγαζιού εικονικά, ώστε το προσωπικό να «βλέπει» ποιο τραπέζι είναι το κάθε νούμερο για να αποφευχθούν τυχόν λάθη.

Αναφορές και βιβλιογραφία

[1] "financesonline," [Online]. Available: https://financesonline.com/top-10-restaurant-management-software-business/.

[2] A. Ndegwa, "stackpath," 31 5 2016. [Online]. Available: https://www.maxcdn.com/one/visual-glossary/web-application/.

[3] Meteor. [Online]. Available: https://www.meteor.com/.

[4] MongoDB. [Online]. Available: https://www.mongodb.com/.

[5] "Node," [Online]. Available: https://nodejs.org/en/.

[6] "JavaScript," [Online]. Available: https://www.javascript.com/.

[7] A. S. Yetayeh, "Developing Dynamic Single Page Web," 2 6 2017. [Online]. Available: https://www.theseus.fi/bitstream/handle/10024/130868/Yetayeh%20Asabeneh-Developing%20Dynamic%20Single%20Page%20WebApplications%20Using%20Meteor.pdf?sequence=1.

[8] S. Greif, "freeCodeCamp," 5 1 2017. [Online]. Available: https://medium.freecodecamp.org/the-5-things-you-need-to-know-to-understand-react-a1dbd5d114a3.

[9] l. orsini, "readwrite," 7 11 2013. [Online]. Available: https://readwrite.com/2013/11/07/what-you-need-to-know-about-nodejs/.

[10] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/MongoDB.

[11] MongoDB, "What is NoSQL?".

[12] MongoDB, "MongoDB," [Online]. Available: https://docs.mongodb.com/manual/core/databases-and-collections/.

[13] MongoDB, "MongoDB," [Online]. Available: https://docs.mongodb.com/manual/aggregation/.

[14] M. Bright, "Using MongoDB and Python," p. 38, 23 2 2016.

[15] D. Finnegan, "Senior Solution Architect, MongoDB inc.," p. 38, 25 11 2014.

[16] S. Khan, "SQL Support over MongoDB using Metadata," 2013.

[17] MongoDB, "MongoDB," [Online]. Available: https://docs.mongodb.com/manual/reference/limits/.

[18] Wikipedia, "Wikipedia," [Online]. Available: https://docs.mongodb.com/manual/reference/limits/.

[19] Mlab, "Mlab," [Online]. Available: https://docs.mlab.com/.

[20] raix, "Github," 26 11 2017. [Online]. Available: https://github.com/GroundMeteor/db.

Εικόνες που χρησιμοποιήθηκαν στην εφαρμογή:

1. https://www.freeiconspng.com/img/14168

2. https://www.colourbox.com/vector/stack-of-euro-coins-vector-icon-black-and-white-cash-illustration-outline-linear-money-icon-vector-26441507

3. https://www.istockphoto.com/illustrations/waiter-black-background?mediatype=illustration&phrase=waiter%20black%20background

Παράρτημα Α’

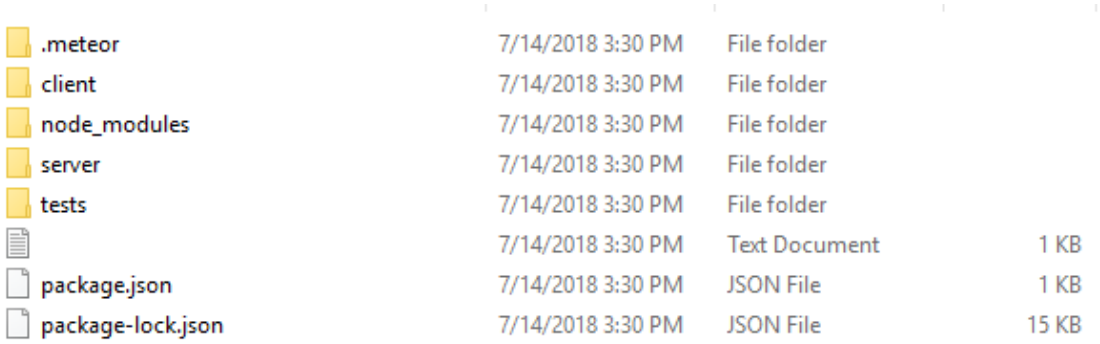
Εγκατάσταση Εφαρμογής

Προαπαιτούμενα:

- Λειτουργικό σύστημα Windows 10
- Heroku CLI (https://devcenter.heroku.com/articles/heroku-cli#download-and-install)
- Meteor για Windows (https://www.meteor.com/install)

Κατά την ολοκλήρωση εγκατάστασης τον παραπάνω εφαρμογών, αρχικά δε χρειάζεται να εγκατασταθεί ξεχωριστά η MongoDB, γιατί μπαίνει μαζί με το Meteor αφού είναι η προκαθορισμένη βάση δεδομένων του.

Με το άνοιγμα της κονσόλας (CMD) και αφού κατευθυνθεί ο χρήστης στο μονοπάτι Path που επιθυμεί με την χρήση της εντολής : **Meteor create “exampleName”** , δημιουργεί αυτόματα το Meteor μια δικιά του αρχική εφαρμογή ιστού και τα παρακάτω αρχεία στο path που επέλεξε.



Εικόνα 0.1 Αρχική δημιουργίας καταλόγων απο το Meteor

Με τη χρήση της εντολής : **Meteor add “examplePackage”** ο χρήστης μπορεί να προσθέσει πακέτα του Meteor και να τα χρησιμοποιήσει στον κώδικα του.

Όλα τα αρχεία που δημιουργήθηκαν κατά την ανάπτυξη της εφαρμογής ιστού βρίσκονται στο

https://github.com/pecharis/waiter

Αν ο χρήστης επιθυμεί και αυτός να ανεβάσει την εφαρμογή στο heroku αφού εγκαταστήσει το Heroku CLI, με τη χρήση της εντολής **Heroku login** στην κονσόλα ενώ βρίσκεται στο φάκελο του Meteor συνδέεται στον λογαριασμό του και στην συνέχεια με τις παρακάτω εντολές κάνει deploy την εφαρμογή.

```
C:\dip\dip\test1>heroku ps:scale web=1
Scaling dynos... done, now running web at 1:Free

C:\dip\dip\test1>git add .

C:\dip\dip\test1>git commit -m "final308"
[master 15b4299] final308
 3 files changed, 81 insertions(+), 45 deletions(-)

C:\dip\dip\test1>git push heroku master
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 4 threads.
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 1.36 KiB | 1.36 MiB/s, done.
Total 7 (delta 6), reused 0 (delta 0)
remote: Compressing source files... done.
remote: Building source:
remote:
```

Εικόνα 0.2 Λίστα εντολών για «ανέβασμα» εφαρμογής στο Heroku

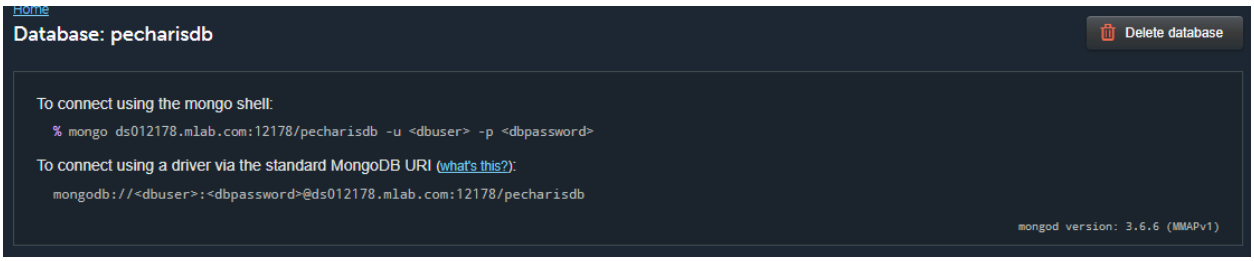
Αν ο κώδικας δεν έχει κάποιο λάθος θα εμφανιστεί το παρακάτω μήνυμα και η εφαρμογή θα είναι πλέον online στο λινκ που δίνει το Heroku.

```
remote: -----> Discovering process types
remote:        Procfile declares types     -> (none)
remote:        Default types for buildpack -> web
remote:
remote: -----> Compressing...
remote:        Done: 49M
remote: -----> Launching...
remote:        Released v30
remote:        https://waitressio.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/waitressio.git
   f801fb3..15b4299  master -> master

C:\dip\dip\test1>git add .
```

Εικόνα 0.3 Επιθυμητό αποτέλεσμα εντολών

Σε περίπτωση που ο χρήστης επιθυμεί και αυτός να χρησιμοποιήσει το mlab που διαθέτει η MongoDB, κάνοντας απλά λογαριασμό στη σελίδα τους και τις απαραίτητες ρυθμίσεις, μπορεί να χρησιμοποιήσει το σύνδεσμο που του δίνει η MongoDB.



Εικόνα 0.4 Mlab πληροφορίες για σύνδεση εφαρμογής με βάση δεδομένων

Με τη χρήση της εντολής :

set MONGO_URL=mongodb://<user>:<password>@ds012178.mlab.com:12178/pecharisdb

και πλέον θα χρησιμοποιείται η online βάση δεδομένων του mlab και όχι του τοπικού μηχανήματος.