ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σχεδιασμός και ανάπτυξη συστήματος παραγγελιών για εστιατόριο

ΠΑΥΛΑΤΟΣ ΖΑΧΑΡΙΑΣ

ΕΠΙΒΛΕΠΩΝ: Χριστοδούλου Σωτήρης, Επίκουρος Καθηγητής

ΠΑΤΡΑ 2020

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή Πάτρα, Ημερομηνία

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

Ονοματεπώνυμο, Υπογραφή

Ονοματεπώνυμο, Υπογραφή

Ονοματεπώνυμο, Υπογραφή

Υπεύθυνη Δήλωση Φοιτητή

Βεβαιώνω ότι είμαι συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τη συγκεκριμένη εργασία.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Πελοποννήσου δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα εκ μέρους του Τμήματος.

Περίληψη

Η παρούσα πτυχιακή εργασία πραγματεύεται τη μελέτη, τη σχεδίαση και την ανάπτυξη Συστήματος Ηλεκτρονικής Παραγγελιοληψίας (Point Of Sale) σε χώρους εστίασης. Σκοπός είναι να δημιουργηθεί ένα ολοκληρωμένο σύστημα, το οποίο να μπορεί να προσφέρει τη δυνατότητα κεντρικής διαχείρισης του φυσικού καταστήματος.

Οι κύριες εργασίες περιλαμβάνουν την ανάπτυξη της εφαρμογής (front-end), τη διαχείριση των παραγγελιών (back-end), της διασύνδεσης μεταξύ των δύο (API) και
μιας βάσης δεδομένων.

Ιδιαίτερη προσοχή θα δοθεί στην ευχρηστία και τη λειτουργικότητα της εφαρμογής.
Θα χρησιμοποιηθούν σύγχρονες τεχνολογίες και αρχιτεκτονικές (React Native, LARAVEL, Pusher, REST) και θα καταβληθεί προσπάθεια, ώστε η εφαρμογή να είναι σωστά
δομημένη, εύχρηστη, λειτουργική, γρήγορη και εύκολα επεκτάσιμη.

Abstract

The present thesis deals with the study, design and development of Point Of Sale Electronic Systems. The aim is to create an integrated system that can provide centralized management of the physical store.

Main tasks include application development (front-end), order management (back-end), the connection between the two (APIs) and a database.

Special attention will be given to the usability and functionality of the application.
Modern technologies, and architectures will be used (ReactNative, LARAVEL, Pusher, REST) and efforts will be made to ensure the application is correctly structured, easy to use, functional, fast and easily scalable.

Ευχαριστίες

Η παρούσα πτυχιακή εργασία πραγματοποιήθηκε με σκοπό την οργάνωση και τη μελέτη ενός πληροφοριακού συστήματος παρακολούθησης και διαχείρισης καταστημάτων εστίασης.

Σε αυτό το σημείο θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα της διπλωματικής μου εργασίας, Επίκουρο Καθηγητή, κ. Σωτήρη Χριστοδούλου, για τη σημαντική βοήθεια, καθοδήγηση και υποστήριξη που μου προσέφερε σε όλα τα στάδια αυτής της πορείας.

Τέλος, εκτιμώ το γεγονός ότι δέχτηκε την ιδέα που πρότεινα για το συγκεκριμένο πληροφοριακό σύστημα και στήριξε την προσπάθειά μου.

Περιεχόμενα

Συντομογραφίες

POS: Point Of Sale
SQL: Structured query Language
RDBMS: Relational Database Management System
RTC: Real Time Communication
EPOS: Electronic Point Of Sale
DOM: Document Object Model
HTML: HyperText Markup Language
CSS: Cascading Style Sheets
XML: Extensible Markup Language
SPA: Single Page Application
URL: Uniform Resource Locator
JSX: JavaScript XML
UI: User Interface
CLI: Command Line Interface
GUI Graphical User Interface
JSP: Java server Pages
ASP: Active server Pages
MVC: Model View Controller
SMTP: Simple Mail Transfer Protocol
TCP: Transmission Control Protocol
OSI: Open Systems Interconnection
HTTP: Hypertext Transfer Protocol
REST: Representational State Transfer
API: Application Programming Interface
JSON: JavaScript Object Notation
JWT: JSON Web Token
AJAX: Asynchronous JavaScript and XML

1 Εισαγωγή

1.1. Περιγραφή αντικειμένου της πτυχιακής εργασίας

Η ανάγκη δημιουργίας ενός κατάλληλου συστήματος ηλεκτρονικής παραγγελιοληψίας, το οποίο θα καλύπτει τις αυξανόμενες απαιτήσεις των ιδιοκτητών καταστημάτων εστίασης, οδήγησαν στη δημιουργία του εν λόγω POS.

Μία από τις σημαντικότερες ανάγκες που διαπιστώθηκαν είναι η δυνατότητα άμεσης ενημέρωσης της τιμολογιακής πολιτικής του καταστήματος και τη δυνατότητα απομακρυσμένου ελέγχου και διαχείρισης του προγράμματος και της αποδοτικότητας των υπαλλήλων. Επίσης, πολύ σημαντική είναι η δυνατότητα παρακολούθησης της κίνησης, του αριθμού των πελατών, των παραγγελιών τους και της εξυπηρέτησης τους, σε πραγματικό χρόνο, καθώς και η δυνατότητα διαχείρισης συγκεντρωτικών στοιχείων, όπως οι ημερήσιες, οι εβδομαδιαίες ή/και οι μηνιαίες εισπράξεις του καταστήματος.

Οι ανάγκες που προαναφέρθηκαν μπορεί να καλυφθούν με τη χρήση του εν λόγω συστήματος, το οποίο αναπτύχθηκε στα πλαίσια αυτής της πτυχιακής.

Συγκεκριμένα, είναι ένα διαδικτυακό πληροφοριακό σύστημα, μέσω του οποίου, οι ιδιοκτήτες των κέντρων εστίασης μπορεί να είναι ενήμεροι, για οποιαδήποτε αλλαγή γίνεται μέσω του συστήματος POS. Υπάρχει η δυνατότητα αμφίδρομης επικοινωνίας μεταξύ καταστήματος και ιδιοκτήτη μέσω του Διαδικτύου. Ο χρόνος που θα χρειάζεται για να ολοκληρωθεί μία παραγγελία, όσο και τα τυχόν λάθη που μπορεί να προκύπτουν λόγω φόρτου εργασίας, θα μειωθούν στο ελάχιστο. Οι υπάλληλοι του καταστήματος θα επιλέγουν εύκολα και γρήγορα από το σύστημα τις παραγγελίες προς παράδοση, οι οποίες θα τιμολογούνται αυτόματα.

Όσον αφορά στην υλοποίηση του Συστήματος Ηλεκτρονικής Παραγγελιοληψίας, χρησιμοποιήθηκαν ως γλώσσες προγραμματισμού οι ακόλουθες: PHP LARAVEL, SQL, React Native, RESTfull αρχιτεκτονική, ενώ η επικοινωνία των συσκευών γίνεται σε πραγματικό χρόνο μέσω των Websockets και Real time communication (RTC).

1.2 Ιστορική αναδρομή

Σε αυτό το υπό κεφάλαιο θα γίνει θα γίνει μια ιστορική αναδρομή στην εξέλιξη των παραγγελιοληψιών σε χώρους εστίασης.

1.2.1. Παραδοσιακός τρόπος παραγγελιοληψίας

Ο κλάδος μαζικής εστίασης έχει αλλάξει σε πολύ μεγάλο βαθμό σε σχέση με τα παλαιότερα χρόνια. Στο παρελθόν, όπου η κοινωνία ήταν μικρότερη και πιο κλειστή, οι ανάγκες για φαγητό ήταν περιορισμένες και καλυπτόντουσαν κυρίως μέσα από τις ίδιες τις οικογένειες και τα νοικοκυριά. Εκείνες τις εποχές ο όρος μαζική εστίαση ήταν περιορισμένος ή σχεδόν ανύπαρκτος. Οι χώροι διασκέδασης και φαγητού ήταν περισσότερο χώροι καφενείων και αναψυκτηρίων και μικρών οικογενειακών ταβερνών.

Ο τρόπος που γίνονταν οι παραγγελίες ήταν απλός: ο σερβιτόρος είχε ένα μπλοκάκι και σημείωνε χειρόγραφα τις επιλογές των πελατών από τον κατάλογο. Αυτό είχε ως αποτέλεσμα, πολλές φορές να γίνονταν παραλείψεις και λάθη. Η παραγγελία μπορούσε να χαθεί είτε να παρερμηνευθεί και έτσι να οδηγήσει στη μείωση της ποιότητας των υπηρεσιών που παρέχονται από την επιχείρηση.

1.2.2. Η Είσοδος των POS στα κέντρα εστίασης

Όλα ξεκίνησαν στα μέσα της δεκαετίας του ‘70, όταν τα POS δεν ήταν κάτι παραπάνω από τις κλασικές γραφομηχανές, οι οποίες είχαν τη δυνατότητα να κάνουν υπολογισμούς. Η IBM κυκλοφόρησε τα συστήματα IBM 3650 και 3660, τα οποία αποτελούσαν ένα σύνολο τερματικών και ήταν διασυνδεδεμένα σε έναν κεντρικό controller. Εκείνος αποτελούσε το κεντρικό σύστημα επεξεργασίας, το οποίο αναλάμβανε όλους τους υπολογισμούς. [1]

Αυτό το σύστημα έγινε ευρέως διαδεδομένο για εμπορική χρήση, καθώς το επέλεξαν πολλές επιχειρήσεις εκείνο το χρονικό διάστημα. Μάλιστα, τα πολυκαταστήματα της Dillard και τα καταστήματα της Pathmark στην Αμερική ήταν τα πρώτα που ενδιαφέρθηκαν να αποκτήσουν το συγκεκριμένο πρωτοποριακό για την εποχή σύστημα. Αυτές οι δύο εταιρείες θέλησαν να “χτυπήσουν” τον ανταγωνισμό έχοντας αυτό το τεράστιο πλεονέκτημα!

Λίγο αργότερα, η μεγαλύτερη αλυσίδα fast food στον κόσμο, McDonald’s, αποφάσισε να αποκτήσει κι εκείνη στα εστιατόρια της αυτό το σύστημα ταμειακών μηχανών που ελέγχονταν από μικροεπεξεργαστές. Αξιοσημείωτο είναι το γεγονός ότι μέχρι το 1976, οι πωλήσεις της αλυσίδας με τη βοήθεια αυτού του συστήματος εκτοξεύθηκαν. Το κέρδος “άγγιξε” τα 3 δισεκατομμύρια δολάρια, αφού οι παραγγελίες ξεπέρασαν τα 20 δισεκατομμύρια χάμπουργκερ!

Στη συνέχεια το σύστημα αυτό εξελίχθηκε από τον William Brodeck το 1974. Αυτή η συσκευή POS διέθετε ένα κουμπί για κάθε στοιχείο του μενού, παρέχοντας με αυτόν τον τρόπο μία γρήγορη και αποτελεσματική υπηρεσία παραγγελίας για τους πελάτες.

Επιπλέον, δημιουργήθηκε η σύνδεση οκτώ συσκευών με έναν υπολογιστή. Αυτό σήμαινε ότι οι τιμές, οι φόροι, οι εκπτώσεις ήταν υπό έλεγχο. Το 1992 ο Martin Goodwin και ο Bob Henry δημιούργησαν το πρώτο POS, το οποίο είχε τη δυνατότητα να “τρέξει” στην πλατφόρμα Microsoft Windows που ονομάζεται IT Retail.

Από τότε ένα ευρύ φάσμα POS εφαρμογών έχουν δημιουργηθεί σε πλατφόρμες όπως τα Windows και τα Unix. Επίσης, το κόστος αυτών των συστημάτων έχει μειωθεί. Οι βασικές απαιτήσεις που πρέπει να πληρούνται από τα σύγχρονα συστήματα POS περιλαμβάνουν: υψηλή και σταθερή ταχύτητα λειτουργίας, αξιοπιστία, ευκολία στη χρήση, εξ αποστάσεως υποστήριξη, χαμηλό κόστος και λειτουργικότητα. [2]

---

1. Πηγή: Point of Sale / https://en.wikipedia.org/wiki/Point_of_sale
2. Πηγή: http://www.cessoftware.com/blog/fascinating-history-pos-systems/

1.2.3. Η εξέλιξη των ηλεκτρονικών παραγγελιοληψιών σε χώρους εστίασης

Στη σύγχρονη εποχή οι συνθήκες ανταγωνισμού ανάμεσα στις επιχειρήσεις που δραστηριοποιούνται στον χώρο της μαζικής εστίασης έκανε επιτακτική την ανάγκη του εκσυγχρονισμού των επιχειρήσεων με την καλύτερη δυνατή μηχανογράφηση και χρήση των νέων δυνατοτήτων της τεχνολογίας.

Το πληροφοριακό σύστημα ορίζεται ως ένα σύνολο από οντότητες, όπως ανθρώπινο δυναμικό, μηχανές (Η/Υ, εκτυπωτές, τηλέφωνα κ.α.) που συνεργάζονται για την επίτευξη ενός συγκεκριμένου στόχου. Ο στόχος αυτός είναι και ο λόγος ύπαρξης του συστήματος. Τα πληροφοριακά συστήματα αρχικά υιοθετήθηκαν από μεγάλες εταιρίες, όμως με την πάροδο του χρόνου άρχισαν να γίνονται αναγκαία για την επιβίωση και των μικρών επιχειρήσεων.

Οι απαιτήσεις των πελατών ολοένα και αυξάνονταν επιζητώντας ταχύτερες και καλύτερης ποιότητας υπηρεσίες οδηγώντας έτσι τις επιχειρήσεις στην αναζήτηση νέων μεθόδων για την κάλυψη αυτών των απαιτήσεων. Παράλληλα, οι ίδιοι οι επιχειρηματίες επιζητούν καλύτερες λύσεις για τη μείωση του λειτουργικού κόστους και των δαπανών της επιχείρησης τους.

Στον χώρο της εστίασης οι βασικές διαδικασίες, όπως για παράδειγμα η εξυπηρέτηση των πελατών, περιελάμβαναν αρχικά τον προφορικό λόγο, στη συνέχεια τον γραπτό και τέλος την εξέλιξή τους σε πληροφοριακά συστήματα τύπου POS (Point Of Sale).

Με την πάροδο των χρόνων ήρθε να προστεθεί στον κόσμο της τεχνολογίας ένα νέο εργαλείο, άκρως καινοτόμο, το cloud-based computing, περνώντας έτσι σε μία νέα εποχή. Η είσοδος του cloud (σύννεφο) άλλαξε εντελώς τα δεδομένα στο σύγχρονο τεχνολογικό περιβάλλον.

Το cloud computing είναι η διάθεση υπολογιστικών πόρων μέσω διαδικτύου (π.χ. servers, apps κλπ), από κεντρικά συστήματα που βρίσκονται απομακρυσμένα από τον τελικό χρήστη, τα οποία τον εξυπηρετούν αυτοματοποιώντας διαδικασίες, παρέχοντας ευκολίες και ευελιξία σύνδεσης.

Η έννοια του Υπολογιστικού Νέφους (Cloud Computing) πρωτοεμφανίστηκε τη δεκαετία του 1950 σε εκπαιδευτικά ινστιτούτα και εταιρείες, και η χρήση του πραγματοποιούνταν από κεντρικούς υπολογιστές μεγάλων υπολογιστικών και αποθηκευτικών δυνατοτήτων.[3]

---

3. Πηγή: Kalliopi, Kanaki (2015-04-27). ΥΠΟΛΟΓΙΣΤΙΚΟ ΝΕΦΟΣ.

Οι χρήστες είχαν πρόσβαση σε αυτούς τους υπολογιστές μέσω τερματικών (dumb terminals), οι οποίοι δεν είχαν ούτε υπολογιστική ισχύ ούτε αποθηκευτικές ικανότητες.

Ο όρος του Υπολογιστικού Νέφους έγινε ευρύτερα γνωστός τη δεκαετία του 1970, όταν η IBM και η Google αποφάσισαν να συνεργαστούν στο συγκεκριμένο τεχνολογικό πεδίο. Αρχικά η IBM παρουσίασε το λειτουργικό σύστημα εικονικών μηχανών (VM operating system), το οποίο παρείχε τη δυνατότητα να εργάζονται πολλές εικονικές μηχανές (virtual machines) στο ίδιο μηχάνημα. Κάθε εικονική μηχανή ήταν και είναι μία αυτοτελής οντότητα που εκτελεί το δικό της λειτουργικό σύστημα και παρέχει υπολογιστικούς πόρους, όπως τη μια κεντρική μονάδα επεξεργασίας, μνήμης και μονάδες εισόδου-εξόδου[4].

Τη δεκαετία του 1980, η πρώτη τακτική χρήση των προσωπικών υπολογιστών συνοδεύτηκε από την υπόσχεση ότι οι χρήστες θα ήταν σε θέση να αποφασίζουν οι ίδιοι για το υπολογιστικό τους περιβάλλον. Η εμφάνιση του Διαδικτύου και του Παγκόσμιου Ιστού, εκτόξευσε τη φήμη του Υπολογιστικού Νέφους.

Τη δεκαετία του 1990 εμφανίστηκαν τα Εικονικά Ιδιωτικά Δίκτυα (Virtual Private Networks). Μέχρι τότε, οι εταιρείες τηλεπικοινωνιών υποστήριζαν τα κυκλώματα δεδομένων σημείο προς σημείο (point-to-point data circuits).

Μέχρι το 2000, κολοσσοί της Πληροφορικής όπως η Amazon, η Microsoft και η Google, ασχολήθηκαν με την ανάπτυξη και την παροχή υπηρεσιών Υπολογιστικού Νέφους[5]. Ακολούθησαν κάποια γεγονότα που θεωρούνται ευρέως ως ορόσημα στην ιστορία του Υπολογιστικού Νέφους.

Το 2006, η Amazon παρουσίασε το Ελαστικό Υπολογιστικό Νέφος (Elastic Compute Cloud(EC2)). Μία εμπορική υπηρεσία βασισμένη στον Παγκόσμιο Ιστό, που παρείχε στον χρήστη τη δυνατότητα να υλοποιεί εφαρμογές σε νοικιασμένα-ενοικιαζόμενα μηχανήματα.

Το 2008, προωθήθηκε στην αγορά το Eucalyptus, η πρώτη πλατφόρμα ανοικτού κώδικα για ανάπτυξη ιδιωτικών σύννεφων (private clouds). Την ίδια χρονιά, η Google κυκλοφόρησε το Google App Engine, μια πλατφόρμα που υποστήριζε διάφορες υπηρεσίες του Υπολογιστικού Νέφους.

4.Πηγή: Καλλιόπη Α. Κανάκη (10 Μάιος 2016). «ΥΠΟΛΟΓΙΣΤΙΚΟ ΝΕΦΟΣ».
5.Πηγή: Qian et al. pp 626-631

Στη συνέχεια των εξελίξεων το 2011, η IBM παρουσίασε το IBM SmartCloud ενώ το 2012 κυκλοφόρησε το Oracle Cloud. Τελος, η Microsoft ανακοίνωσε επίσης ότι θα εισαγάγει το cloud computing στην επόμενη μεγάλη ενημέρωση λύσεων Dynamics ERP και θα λειτουργούν μέσω της πλατφόρμας Windows Azure[6].

6. Πηγή: «Microsoft Says to Spend 90% of R&D on Cloud Strategy»

1.3. Cloud και POS

Τα συστήματα POS που βασίζονται σε Cloud επεξεργάζονται και αποθηκεύουν δεδομένα σε απευθείας σύνδεση, σε αντίθεση με τα τοπικά δεδομένα στον υπολογιστή ή στον διακομιστή επί τόπου. Η διασύνδεση POS frontend γίνεται για χρήση σε συσκευές με δυνατότητα Internet, όπως tablet, smartphones και οθόνες αφής, ενώ το backend είναι προσβάσιμο μέσω οποιουδήποτε προγράμματος περιήγησης στο Web.

Το Cloud POS εξαλείφει την ανάγκη για διακομιστή και μετακινεί τη ροή της κυκλοφορίας στο κέντρο δεδομένων κάποιου άλλου. Συνήθως, το δίκτυο είναι ασύρματο, οπότε δεν υπάρχει ανάγκη διασύνδεσης με καλώδια.

Τα συστήματα POS που βασίζονται σε Cloud αναφέρονται μερικές φορές ως εφαρμογές POS αντί λογισμικού, δεδομένου ότι χρησιμοποιούνται συχνά σε κινητές συσκευές. Ονομάζονται επίσης EPOS (ηλεκτρονικό σημείο πώλησης) για να αντικατοπτρίζουν το γεγονός ότι το λογισμικό λειτουργεί με ηλεκτρονικό εξοπλισμό.[7]

Το Cloud POS μπορεί να είναι πραγματικά φορητό ενώ ο διαχειριστής επιχείρησης μπορεί να ελέγχει και να χρησιμοποιεί δεδομένα πωλήσεων από οπουδήποτε καθώς ενημερώνονται σε πραγματικό χρόνο στο cloud. Σημαίνει επίσης χαμηλότερο κόστος εξοπλισμού, δεδομένου ότι ένα υπάρχον tablet ή κινητό τηλέφωνο μπορεί να χρησιμοποιηθεί ως διασύνδεση POS.
Το Cloud-based POS συνήθως απαιτεί εγγραφή στο διαδίκτυο ή μέσω τηλεφώνου (ο πάροχος μπορεί να εκτελέσει έλεγχο ιστορικού) και στη συνέχεια να κατεβάσει την εφαρμογή POS που είναι έτοιμη για άμεση χρήση. [8]

Με το Cloud POS, οι έμποροι πληρώνουν μηνιαία χρέωση ενοικίασης για πρόσβαση στο λογισμικό. Το μοντέλο υπηρεσιών είναι λογισμικό ως υπηρεσία, σε αντίθεση με το Salesforce ή το Office365. Ο έμπορος απλώς χρησιμοποιεί το προϊόν. Το Cloud δεν απαιτεί μια δαπανηρή αγορά εξοπλισμού. Οι ενημερώσεις προωθούνται σε συσκευές από το cloud.
Ο προμηθευτής του cloud POS θα πρέπει να χρησιμοποιεί ανοικτές διεπαφές προγραμματισμού εφαρμογών (APIs). Αυτό επιτρέπει την εύκολη πρόσβαση σε πρόσθετα ή υπηρεσίες τρίτου μέρους, όπως η παραγγελία / πληρωμή και η παράδοση μέσω κινητού ή ηλεκτρονικού ταχυδρομείου.[9]

---

7. https://www.mobiletransaction.org/cloud-based-pos-system/
8.https://www.forbes.com/sites/forbestechcouncil/2018/05/25/what-is-cloud-point-of-sale-and-how-is-it-different-from-traditional-pos/#400eb3261180
9. https://hospitalitytech.com/2018-pos-software-trends-report

Αντίθετα, τα παλαιά συστήματα POS (δηλ. Φιλοξενούμενα επί τόπου / τοπικά) κοστίζουν συνήθως ένα τέλος εγκατάστασης και απαιτούν από έναν εκπαιδευμένο μηχανικό να ρυθμίσει εξοπλισμό, να προσαρμόσει το λογισμικό στις εκάστοτε ανάγκες και να το εγκαταστήσει στο τοπικό δίκτυο υπολογιστών. Τα επιτόπια συστήματα POS δεν μπορούν να χρησιμοποιηθούν προτού ολοκληρωθεί η εγκατάσταση υλικού και λογισμικού.

Οι λύσεις που βασίζονται στο Cloud παρέχουν απτά πλεονεκτήματα, όπως είναι το κόστος, η απλότητα, η ευελιξία και η λειτουργικότητα.

Το EPOS που βασίζεται σε cloud δημιουργείται συνήθως για συγκεκριμένους τομείς, όπως εστιατόρια ή λιανικό εμπόριο. Επιπλέον, πολλές από τις εφαρμογές POS μπορούν να ενσωματωθούν με διάφορες πλατφόρμες συνεργατών όπως το Xero και το WooCommerce, δημιουργώντας μεγάλο περιθώριο για ανάπτυξη και ευελιξία.

Τέλος, η ασφάλεια των δεδομένων που παρέχει το Cloud είναι το μεγαλύτερο πλεονέκτημα.
Όλα τα δεδομένα POS που βασίζονται σε Cloud αποθηκεύονται εκεί και έτσι είναι εξαιρετικά ασφαλή. Συνήθως, κανένα από τα δεδομένα δεν είναι αποθηκευμένο στις συσκευές, εκτός εάν η εφαρμογή POS μπορεί να λειτουργήσει προσωρινά εκτός σύνδεσης, οπότε η συσκευή αποθηκεύει τις αλλαγές μέχρι την επόμενη φορά που ο χρήστης είναι συνδεδεμένος στο διαδίκτυο. Αν χαθεί όλο το υλικό, ο χρήστης εξακολουθεί να έχει άμεση πρόσβαση στα δεδομένα πωλήσεων και στο σύστημα κατά τη σύνδεση σε νέο υπολογιστή ή κινητή συσκευή. Ο πάροχος POS εκτελεί αυτόματα όλες τις ενημερώσεις ασφαλείας που απαιτούνται στο λογισμικό.

Τα τοπικά φιλοξενούμενα συστήματα POS αποθηκεύουν δεδομένα στον φυσικό διακομιστή, συνήθως έναν σκληρό δίσκο του υπολογιστή. Αυτό σημαίνει ότι εάν συμβεί κάτι στον υπολογιστή, τα δεδομένα βρίσκονται σε κίνδυνο. Συνεπώς, απαιτείται να διατηρείται το λειτουργικό σύστημα ενημερωμένο με τις πιο πρόσφατες ενημερώσεις ασφαλείας και να υπάρχει καλή συντήρηση του εξοπλισμού.

1.4. Πλεονεκτήματα των συστημάτων παραγγελιοληψίας POS

Βασικά στοιχεία που αφορούν στα πλεονεκτήματα της χρήσης των εν λόγω συστημάτων σχετίζονται με την ακρίβεια των στοιχείων πωλήσεων καθώς και των σχετικών στατιστικών. Αυτά τα στοιχεία μπορούν να φανούν ιδιαίτερα χρήσιμα για τον σωστό προγραμματισμό παραγωγής ή εισαγωγών προμηθειών. Τα ποιοτικά οφέλη αφορούν στην παροχή υψηλότερου επιπέδου υπηρεσιών στους πελάτες, καθώς και στην καλύτερη διαχείριση του χώρου εστίασης.

Η ανάπτυξη της εφαρμογής των φορητών τερματικών αφορά τόσο την ανάλυση των αναγκών του πελάτη όσο και την ανάπτυξη λογισμικού για την επίτευξη της επικοινωνίας των συσκευών με την κεντρική μονάδα.

Όλα τα συστήματα που υπάρχουν στην αγορά και είναι ασύρματα παρέχουν τα παρακάτω πλεονεκτήματα όπως παρουσιάζονται από τους Manion & DeMicco, 2004. [10]

- Το προσωπικό μπορεί εύκολα και γρήγορα να δει την κατάσταση των τραπεζιών μέσα από τις ασύρματες ηλεκτρονικές συσκευές
- Ο σερβιτόρος κερδίζει περισσότερο χρόνο εξυπηρετώντας των πελάτη
- Υπάρχει μεγαλύτερη ακρίβεια στη λήψη της παραγγελίας
- Υπάρχει μεγαλύτερη αποτελεσματικότητα
- Βελτιώνεται η εξυπηρέτηση πελατών
- Εξαλείφεται η συμφόρηση της αποστολής παραγγελιών
- Κάθε σερβιτόρος μπορεί να ελέγχει περισσότερα τραπέζια
- Μειώνεται ο χρόνος αναμονής των σερβιτόρων
- Στα τραπέζια η ολοκλήρωση των διαδικασιών γίνεται γρηγορότερα (γρηγορότερη διαδικασία παραγγελιών, γρηγορότερη διαδικασία πληρωμών)
- Δεν υπάρχει περίπτωση να χαθούν οι παραγγελίες
- Υπάρχει η δυνατότητα εξόρυξης στατικών για να γίνεται καλύτερη διαχείριση του χώρου εστίασης. Περιλαμβάνονται αναφορές στα προϊόντα με τις καλύτερες και χειρότερες πωλήσεις καθώς και τα στατιστικά των τζίρο.

---

10. Πηγή: https://www.tandfonline.com/doi/abs/10.1300/J369v07n02_07

2. Τεχνολογίες Ανάπτυξης

Σε αυτό το κεφάλαιο αναλύονται οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη του POS.

2.1.Η γλώσσα προγραμματισμού JavaScript

Η JavaScript είναι μία scripting γλώσσα προγραμματισμού, η οποία αποτελεί μια από τις τρεις βασικές τεχνολογίες του World Wide Web (HTML, CSS και JavaScript). Χρησιμοποιείται για να προσθέσει διαδραστικές και δυναμικές δυνατότητες στις ιστοσελίδες.

Όλα ξεκίνησαν το 1995, όταν ο Marc Andeerssen, ιδρυτής της Netscape Communications, είχε το όραμα ότι ο ιστός (web) χρειάζεται έναν τρόπο να γίνει πιο δυναμικός. Οι κινούμενες εικόνες(animations), η αλληλεπίδραση και άλλες μορφές μικρών αυτοματισμών θα έπρεπε να αποτελούν μέρος του ιστού του μέλλοντος. Ως αποτέλεσμα, το web θα χρειαζόταν μια μικρή scripting γλώσσα που θα μπορούσε να αλληλεπιδράσει με το DOM (Document Object Model).

Το DOM είναι μια διεπαφή προγραμματισμού για τα HTML documents (μια σελίδα για παράδειγμα αποτελεί ένα document). Αντιπροσωπεύει τη σελίδα σε μορφή αντικειμένου, έτσι ώστε τα προγράμματα να μπορούν εύκολα να αλλάξουν τη δομή και το περιεχόμενο του. Επίσης, και αυτή ήταν μια σημαντική στρατηγική απόφαση εκείνη την εποχή, αυτή η γλώσσα δεν θα έπρεπε να προσανατολίζεται σε προγραμματιστές με εμπειρία στο χώρο της τεχνολογίας και ανάπτυξης λογισμικού. Αντιθέτως, η γλώσσα για το διαδίκτυο θα έπρεπε να εξυπηρετεί έναν διαφορετικό τύπο κοινού: τους σχεδιαστές (designers).

Εκείνη την εποχή, ο ιστός ήταν στατικός(static). Η HTML ήταν ακόμη αρκετά νέα και αρκετά απλή στην εκμάθηση για τους μη προγραμματιστές, άρα, και οι δυνατότητες που θα έκαναν το web πιο δυναμικό θα έπρεπε να ήταν προσβάσιμες και εύκολες σε μη προγραμματιστές. Έτσι γεννήθηκε η ιδέα της Mocha (η πρώτη ονομασία της JavaScript).

Για αυτό τον σκοπό ο Brendan Eich, πατέρας της JavaScript, είχε προσληφθεί από την Netscape Communications για να αναπτύξει μια γλώσσα που θα τηρούσε τις απαιτήσεις. Ωστόσο, η Netscape Communications συνομιλούσε ταυτόχρονα και με την Sun Microsystems για να κλείσει μια συμφωνία ώστε να καταστήσει την Java διαθέσιμη στα προγράμματα περιήγησης ιστού (browsers). Παρόλα αυτά, υπήρχε ένα πρόβλημα. Η Java ήταν απλά πάρα πολύ μεγάλη, πολύ σύνθετη για αυτό τον ρόλο, δε θα μπορούσε ποτέ να χρησιμοποιηθεί εύκολα από συγγραφείς, ερασιτέχνες, σχεδιαστές. Έτσι, η ιδέα ήταν να καταστεί η Java διαθέσιμη για έμπειρους επαγγελματίες, ενώ η Mocha θα χρησιμοποιούνταν για μικρά scripts. Με άλλα λόγια, η Mocha προοριζόταν να είναι ο scripting companion της Java, κατά τρόπο ανάλογο τη σχέσης μεταξύ C/C ++ και Visual Basic στην πλατφόρμα των Windows.

Το πρωτότυπο της Mocha ενσωματώθηκε στον Netscape Communicator τον Μάιο του 1995 και σε σύντομο χρονικό διάστημα μετονομάστηκε σε LiveScript. Το Δεκέμβριο του 1995, η Netscape Communications και η Sun έκλεισαν τη συμφωνία η LiveScript να μετονομαζόταν σε JavaScript, η οποία θα παρουσιαζόταν ως scripting γλώσσα για μικρές client-side εργασίες στον περιηγητή ιστού (browser), ενώ η Java θα προωθούνταν ως το κατάλληλο επαγγελματικό εργαλείο για την ανάπτυξη πλούσιων web εφαρμογών. [11]

11. Πηγή: Auth0, “A Brief History of JavaScript”, https://auth0.com/blog/a-brief-history-of-javascript/

2.1.2. Η εξέλιξη της JavaScript

Από τις πρώτες ημέρες, η JavaScript έκανε σημαντική διαφορά στην εμπειρία των χρηστών, με αποτέλεσμα όλα τα ανταγωνιστικά προγράμματα περιήγησης ιστού (browsers) να μην έχουν άλλη επιλογή παρά να βρουν μια λύση, μια λειτουργική υλοποίηση της JavaScript. Επί του παρόντος, και για πολύ μεγάλο χρονικό διάστημα, τα πρότυπα του web δεν ήταν ισχυρά. Έτσι, η Microsoft υλοποίησε τη δική της έκδοση JavaScript, που ονομαζόταν JScript. Κρατώντας τη "Java" εκτός ονόματος απέφυγε πιθανά ζητήματα εμπορικών σημάτων (trademarks). Ωστόσο, η JScript ήταν διαφορετική σε κάτι περισσότερο από το όνομα. Οι μικρές διαφορές στην εφαρμογή, ιδίως σε σχέση με ορισμένες λειτουργίες του DOM, προκάλεσαν προβλήματα που γίνονται αισθητά ακόμα και σήμερα.

Η πρώτη μεγάλη αλλαγή για τη JavaScript μετά την έκδοση της είχε τη μορφή τυποποίησης ECMA. Η ECMA (European Computer Manufacturers Association, Ένωση Ευρωπαϊκών Κατασκευαστών Ηλεκτρονικών Υπολογιστών) είναι μια βιομηχανική ένωση που ιδρύθηκε το 1961 και αφορούσε αποκλειστικά την τυποποίηση των συστημάτων πληροφοριών και επικοινωνιών. Η τυποποίηση ήταν ένα σημαντικό βήμα για μια τέτοια νέα γλώσσα, και είχε ως αποτέλεσμα να ανοίξει σε ένα ευρύτερο κοινό και να δώσει σε άλλους επαγγελματίες φωνή στην εξέλιξη της γλώσσας. Για λόγους εμπορικού σήματος, η επιτροπή ECMA δεν μπόρεσε να χρησιμοποιήσει τη JavaScript ως όνομα. Οι εναλλακτικές λύσεις δεν άρεσαν σε πολλούς, οπότε μετά από συζήτηση αποφασίστηκε ότι η γλώσσα που περιγράφεται από το πρότυπο θα ονομάζεται ECMAScript. Σήμερα, η JavaScript είναι μόνο το εμπορικό όνομα για την ECMAScript.

Η ECMAScript 3 κυκλοφόρησε τον Δεκέμβριο του 1999 και οι πρώτες μεγάλες αλλαγές στη γλώσσα είδαν το φως.
Αυτή η έκδοση έφερε:

- Κανονικές εκφράσεις (Regular expressions)
- Το βρόχο do-while
- Εξαιρέσεις (Exceptions)
- Περισσότερες ενσωματωμένες συναρτήσεις (built-in functions) για πίνακες (arrays) και αλφαριθμητικά (strings)
- Τους τελεστές in και instanceof
- Πολύ καλύτερη διαχείριση λαθών

Αυτή η έκδοση της ECMAScript εξαπλώθηκε ευρέως. Υποστηρίχθηκε από όλα τα μεγάλα προγράμματα περιήγησης εκείνης της εποχής και συνέχισε να υποστηρίζεται για πολλά χρόνια. Αυτό έκανε την ECMAScript 3 την βάση για πολλές βιβλιοθήκες, ακόμα και όταν εκδόθηκαν νεότερες εκδόσεις του προτύπου. Επίσης, το AJAX (Asynchronous JavaScript and

XML) ήταν μια τεχνική που γεννήθηκε στα χρόνια της ECMAScript 3. Αν και δεν ήταν μέρος του προτύπου, η Microsoft υλοποίησε ορισμένες επεκτάσεις της JavaScript για τον Internet Explorer. Αυτή η λειτουργία επέτρεψε σε ένα πρόγραμμα περιήγησης να εκτελεί ένα ασύγχρονο αίτημα HTTP σε έναν server, επιτρέποντας έτσι στις σελίδες να ενημερώνονται σε πραγματικό χρόνο.

Τα επόμενα χρόνια δεν ήταν καλά για την ανάπτυξη της JavaScript. Μόλις ξεκίνησαν οι εργασίες για την ECMAScript 4, άρχισαν να εμφανίζονται έντονες διαφορές στην επιτροπή. Υπήρξε μια ομάδα ανθρώπων που πίστευαν ότι η JavaScript χρειάζονταν χαρακτηριστικά που θα την έκαναν μια ισχυρότερη γλώσσα για την ανάπτυξη εφαρμογών μεγάλης κλίμακας. Αυτή η ομάδα πρότεινε πολλά χαρακτηριστικά μεγάλης εμβέλειας και αλλαγών. Αρκετοί όμως θεωρούσαν ότι αυτή δεν θα ήταν η κατάλληλη εξέλιξη για τη JavaScript. Η έλλειψη συναίνεσης και η πολυπλοκότητα ορισμένων προτεινόμενων χαρακτηριστικών καθυστερούσε συνεχώς τη έκδοση της ECMAScript 4 ώσπου το 2003 σταμάτησε τελείως η ανάπτυξη της. Το 2005, ο αντίκτυπος του AJAX πυροδότησε και πάλι το ενδιαφέρον για μια νέα έκδοση της JavaScript με αποτέλεσμα να αρχίσει εκ νέου η ανάπτυξη της. Τα χρόνια περνούσαν και το σύνολο των δυνατοτήτων της συνεχώς αυξανόταν, με κάποια από τα σημαντικότερα χαρακτηριστικά της να είναι:

- Κλάσεις
- Διεπαφές (Interfaces)
- Namespaces
- Πακέτα (Packages)
- Πολλαπλές μεθόδους (Multi methods)
- Επαναλήπτες (Iterators)
- Παραγωγοί (Generators)
- Σωστότερο πεδίο εφαρμογής (Proper block scoping)

Παρόλα αυτά, από το 2008 και μετά, η κοινότητα επικεντρώθηκε στην ανάπτυξη της ECMAScript 3.1. Η ECMAScript 4 απορρίφθηκε. Η ECMAScript 3.1 ολοκληρώθηκε και εκδόθηκε το 2009. Η ECMAScript 4 ακόμη και αν δεν εκδόθηκε ποτέ επίσημα, αναγνωρίστηκε ως ειδική παραλλαγή της ECMAScript. Το αποτέλεσμα ήταν η απόφαση της επιτροπής να μετονομάσει την ECMAScript 3.1 σε ECMAScript 5 για να αποφευχθεί η σύγχυση. Η ECMAScript 5 έγινε μια από τις πλέον υποστηριζόμενες εκδόσεις της JavaScript και υποστηρίχθηκε εξ ολοκλήρου από τα τότε μεγάλα προγράμματα περιήγησης ιστού (browsers), όπως ο Firefox 4 (2011), ο Chrome 19 (2012), ο Safari 6 (2012), ο Opera 12.10 (2012) και ο Internet Explorer 10 (2012). Η ECMAScript 5 ήταν μια μέτρια αναβάθμιση της ECMAScript 3 και περιλάμβανε:

- Οι μέθοδοι Getter/setter
- Νέες μεθόδους για αντικείμενα
- Νέες μεθόδους για πίνακες

- Νέες μεθόδους για ημερομηνίες
- Σύνδεση συναρτήσεων (Function binding)
- JSON
- Αμετάβλητα καθολικά αντικείμενα (Immutable global objects)

Η ECMAScript 6, που μετονομάστηκε αργότερα σε ECMAScript 2015, σχεδιάστηκε για να φέρει μεγάλες αλλαγές. Ορισμένα από τα μεγάλα χαρακτηριστικά της ECMAScript 4 εφαρμόστηκαν σε αυτή την έκδοση, ωστόσο, εφαρμόστηκαν με διαφορετική νοοτροπία. Αυτή τη φορά στην επιτροπή επικρατούσε ενότητα, με αποτέλεσμα σχεδόν κάθε αλλαγή που είχε συζητηθεί στο παρελθόν και απαιτούσε συντακτικές αλλαγές να υλοποιηθεί σε αυτήν την έκδοση. Πολλοί προμηθευτές προγραμμάτων περιήγησης ιστού άρχισαν να εργάζονταν στην υλοποίηση των χαρακτηριστικών της ECMAScript 2015, αλλά οι αλλαγές ήταν πολλές με αποτέλεσμα να χρειαστεί αρκετός χρόνος για να υλοποιηθούν. Ακόμα και σήμερα, οι browsers δεν έχουν πλήρη κάλυψη της ECMAScript 2015. Μια σύντομη περίληψη των πιο σημαντικών νέων στοιχείων είναι:

- Νέοι τρόποι δήλωσης μεταβλητών με χρήση του let και του const
- Συναρτήσεις βέλους (Arrow functions)
- Κλάσεις
- Πρότυπα αλφαριθμητικά (Template strings)
- Υποσχέσεις (Promises)
- Προεπιλεγμένοι παράμετροι (default arguments) για τις συναρτήσεις (functions)
- Το Rest/Spread συντακτικό
- Το συντακτικό ενοτήτων (Module syntax)
- Νέα collections (Set, Map)
- Αντανάκλαση (Reflection)

2.2. ReactJS

Ένα από τα μεγαλύτερα προβλήματα της ανάπτυξης ιστοσελίδων είναι ο χειρισμός του DOM. Ο άμεσος χειρισμός του DOM όσον αφορά τους πόρους του συστήματος είναι ακριβός. Η λήψη δεδομένων μιας JavaScript εφαρμογής και η απεικόνιση τους στο πρόγραμμα περιήγησης ιστού (browser) είναι μια δαπανηρή και αργή λειτουργία. Η ReactJS είναι η πρώτη βιβλιοθήκη που καταλήγει σε ένα προφανές συμπέρασμα: η βελτιστοποίηση του χειρισμού του DOM οδηγεί στη δημιουργία μιας γρήγορης βιβλιοθήκης και σε αυτή την περίπτωση σε μια βιβλιοθήκη που προσφέρει τη δυνατότητα ανάπτυξης ιστοσελίδων με χρήση λίγου κώδικα.

Η ReactJS δημιουργήθηκε από το Facebook, και χρησιμοποιείται για τη δημιουργία σύνθετων, reactive και γρήγορων ιστοσελίδων. Ενθαρρύνει τη δημιουργία

επαναχρησιμοποιήσιμων κομματιών κώδικα (components) που παρουσιάζουν δεδομένα τα οποία αλλάζουν συχνά και έχουν ως απαίτηση τις πολλές και συχνές ανανεώσεις του DOM.

Η ReactJS είναι μια βιβλιοθήκη που ακολουθεί την αρχιτεκτονική της μιας και μόνο σελίδας ανά website (Single Page Application, SPA). Αυτό επιτυγχάνεται με την δυναμική ανανέωση, και όχι επαναφόρτωση, της σελίδας που αλληλεπιδρά με τον χρήστη, με την χρήση AJAX.

Τα Single Page Application websites πλέον δε χωρίζονται σε αυτόνομες σελίδες τις οποίες ζητάμε με ένα URL από τον server, ο οποίος μας τις «σερβίρει» ανακατευθύνοντας μας σε αυτές. Αντιθέτως τα URLs έχουν πλέον τον ρόλο των διαδρομών (routes). Τα routes θα μπορούσαν να χαρακτηριστούν ως υποσελίδες της μοναδικής σελίδας του website. Όταν ζητήσουμε ένα URL, το περιεχόμενο της σελίδας ανανεώνεται δυναμικά και πλέον χωρίς επαναφόρτωση δείχνει τα περιεχόμενα τα οποία έχουμε συνδέσει με το αντίστοιχο route. Η ReactJS υλοποιεί την αρχιτεκτονική των Single Page Applications αποθηκεύοντας την κατάσταση (state) της ιστοσελίδας σε μια δική της εσωτερική δενδρική μορφή (εικονικός DOM), και χειρίζεται όλες τις οπτικές αλλαγές ανανεώνοντας τα περιεχόμενα της στο πρόγραμμα περιήγησης ιστού (πραγματικός DOM), μόνο όταν έχει αλλάξει η κατάσταση (state) της. Η ReactJS πηγαίνει τα πράγματα ένα βήμα παραπέρα γιατί γνωρίζει πότε έχει ή δεν έχει αλλάξει η κατάσταση ή συγκεκριμένα στοιχεία της ιστοσελίδας, με αποτέλεσμα η ανανέωση του DOM να γίνεται αυτόματα και μόνο για τα στοιχεία που έχουν αλλάξει. Ο αλγόριθμος που χρησιμοποιεί για να κατανοεί πότε άλλαξε το δέντρο κατάστασης έχει πολυπλοκότητα O(n), χρήζοντάς την ιδανική για μεγάλες εφαρμογές Αυτό το καταφέρνει χρησιμοποιώντας ένα hash map των στοιχείων του DOM, με μοναδικά κλειδιά.

Η ReactJS χρησιμοποιεί την JSX, έναν preprocessor που προσθέτει τη δυνατότητα χρήσης XML συντακτικού στην JavaScript. Δεν είναι υποχρεωτική η χρήση της JSX, αλλά προτείνεται επειδή κάνει πιο εύκολη και καθαρή την ανάπτυξη ιστοσελίδων.

```
const Message = () => {
  return (
    <div>
      Hello World
    </div>
  )
}
```

Συντακτικό ReactJS/JSX

Η ReactJS προτείνει το σπάσιμο του κώδικα σε μικρά διακριτά κομμάτια, τα οποία έχουν μια και μόνο συγκεκριμένη λειτουργία, έτσι ώστε να είναι σαφή, καθαρά και να μπορούν να επαναχρησιμοποιηθούν όσο το δυνατό περισσότερο. Αυτά τα κομμάτια κώδικα τα ονομάζει

components και μπορούν να χρησιμοποιηθούν σαν HTML tags, με αποτέλεσμα ένα component να μπορεί να περιέχει μέσα του περισσότερα components.

```
const TableHeader = () => {
    return (
        <thead>
            <tr>
                <th> This is a Header</th>
            </tr>
        </thead>
    )
}
```

Component για table header

```
const TableContents = () => {
    return (
        <tbody>
            <tr>
                <td> Cell 1</td>
                <td> Cell 2</td>
                <td> Cell 3</td>
            </tr>
        </tbody>
    )
}
```

Component για table body

```
const Table = () => {
    return (
        <table>
            <TableHeader/>
            <TableContents/>
        </table>
    )
}
```

Component ενός table με χρήση components για το header και το body

2.3. React Native

Η React Native είναι ένα JavaScript framework για τη σύνταξη πραγματικών, εγγενώς μετατρέψιμων κινητών εφαρμογών για iOS και Android. Βασίζεται στη JavaScript βιβλιοθήκη της React, του Facebook για την ανάπτυξη εφαρμογών. Όμως αντί να στοχεύει στο πρόγραμμα περιήγησης, στοχεύει στα κινητά.

Με άλλα λόγια, επιτρέπει στους προγραμματιστές να γράφουν εφαρμογές για κινητά που μοιάζουν και εχουν την αίσθηση μιας πραγματικά "μητρικής" (native) εφαρμογής, όλα αυτά από την άνεση μιας γνωστής βιβλιοθήκης JavaScript. Ακομα επειδή το μεγαλύτερο μέρος του κώδικα που γράφεται μπορεί να μοιραστεί μεταξύ των πλατφορμών, η React Native καθιστά εύκολη την ταυτόχρονη ανάπτυξη εφαρμογών τόσο για Android όσο και για iOS.

Παρόμοια με την React για τον ιστό, οι React Native εφαρμογές είναι γραμμένες με ένα μείγμα JavaScript και XML, γνωστή ως JSX.

Στη συνέχεια, η React Native μεσω μιας "γέφυρας" επικαλείται τα μητρικα (native) API για rendering σε Objective-C (για iOS) ή Java (για Android).

Έτσι, η παραγόμενη εφαρμογή κάνει χρήση πραγματικών στοιχείων κινητού UI και όχι webview, έτσι μοιάζουν και έχουν την αίσθηση μιας οποιασδήποτε “μητρικής”(native) εφαρμογής για κινητά.

Η React Native επίσης έχει πρόσβαση στο API της πλατφόρμας, γι’ αυτό μπορεί να έχει πρόσβαση σε λειτουργίες, όπως η κάμερα του τηλεφώνου ή η τοποθεσία του χρήστη.
Ο πυρήνας της React Native υποστηρίζει τη δημιουργία εφαρμογών για κινητά iOS και Android. Η κοινοτητα (community) της React Native παρέχει επίσης υποστήριξη για Windows,Ubuntu, το διαδίκτυο και πολλά άλλα. Μπορεί πραγματικά να χρησιμοποιηθει η React Native για την κατασκευή κινητών εφαρμογών έτοιμων προς παραγωγή.
Κάποιες από τις εφαρμογές γραμμένες σε React Native ειναι οι ακολουθες: Facebook, Airbnb, Instagram, Walmart και Baidu.

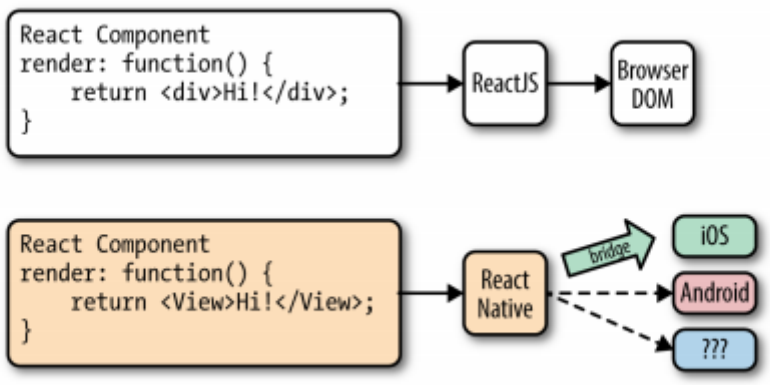
Ο κύκλος επικαιροποίησης της React Native είναι ο ίδιος με την React: όταν αλλάζει η κατάσταση (state), η React Native ανανεώνει την εμφάνιση. Η μεγάλη διαφορά μεταξύ της React Native και της React στον browser είναι ότι η React Native το κάνει αυτό με τις βιβλιοθηκες UI της πλατφόρμας, όπου τρέχει αντί να χρησιμοποιείται σήμανση HTML και CSS.

Στο πλαίσιο της React στο διαδίκτυο, οι περισσότεροι προγραμματιστές σκέφτονται το εικονικό DOM ως στοιχείο για την βελτιστοποίηση της απόδοσης. Το Virtual DOM έχει

ασφαλώς οφέλη απόδοσης, αλλά η πραγματική του δυνατότητα έγκειται στη δύναμη της αφαιρετικότητας του.

Τοποθετώντας ένα καθαρό στρώμα αφαιρετικότητας μεταξύ του κώδικα του προγραμματιστή και της πραγματικής ερμηνείας ανοίγει πολλές ενδιαφέρουσες δυνατότητες. Τι θα μπορούσε να συμβεί αν η React θα ερμηνεύσει (render) σε κάτι διαφορετικό από το DOM του προγράμματος περιήγησης;

Εξάλλου, η React ήδη "καταλαβαίνει" τι θα έπρεπε να ερμηνεύσει η εφαρμογή.
Πράγματι, έτσι λειτουργεί η React Native, όπως φαίνεται στο παρακάτω σχήμα. Αντί να ερμηνεύει στο DOM του προγράμματος περιήγησης, η React Native επικαλείται το API της Objective-C για απόδοση σε iOS ή Java APIs για να προβάλλεται σε Android. Αυτό ξεχωριζει την React Native από άλλες επιλογές ανάπτυξης cross-platform εφαρμογών, οι οποίες συχνά καταλήγουν σε απόδοση διαδικτυακής προβολής (webview). [12]



Όλα αυτά είναι εφικτά λόγω της γέφυρας, η οποία παρέχει η React με μια διεπαφή στα εγγενή στοιχεία UI της πλατφόρμας υποδοχής.

Τα στοιχεία (Components) επιστρέφουν την JSX από τη δική τους render function, η οποία περιγράφει τον τρόπο εμφάνισης. Με την React για τον ιστό, αυτό μεταφράζεται απευθείας στο DOM του προγράμματος περιήγησης.

---

12. Πηγή: Learning React Native by Bonnie Eisenman (2018)

Με την React Native, η JSX μεταφράζεται για να ταιριάζει με την πλατφόρμα υποδοχής, έτσι ώστε το <View> στοιχείο να μπορεί να γίνει UIView που είναι ειδικά για το iOS.
Ο πυρήνας της React Native υποστηρίζει iOS και Android. Λόγω της αφαίρεσης του αφαιρετικού στρώματος που παρέχεται από το Virtual DOM, η React Native μπορεί να στοχεύσει σε άλλες πλατφόρμες αν κάποιος αναπτύξει την αντίστοιχη γέφυρα.
Για παράδειγμα υπάρχει μια κοινότητα προγραμματιστών, η οποία αναπτύσσει την γέφυρα για Windows και Ubuntu.

Η γέφυρα μεταφράζει JavaScript calls και καλεί το API της εκάστοτε πλατφόρμας και τα αντίστοιχα στοιχεία UI (δηλ. Objective-C ή Java, κατά περίπτωση).
Επειδή η React Native δεν τρέχει στο κύριο νήμα του UI, μπορεί να εκτελέσει ασύγχρονες κλήσεις χωρίς να επηρεάζουν την απόδοση και κατά συνέπεια την εμπειρία του χρήστη.

Όταν γράφουμε React για το web, χρησιμοποιούμε κατά κανόνα HTML στοιχεία (, <p>, , <a>, κλπ.).

Με την React Native, όλα αυτά τα στοιχεία αντικαθίστανται από συγκεκριμένα στοιχεία της πλατφόρμας React Native. Το πιο βασικό είναι το cross-platform <View>, ένα απλό και ευέλικτο στοιχείο UI που μπορεί να θεωρηθεί ως ανάλογο με το . Για παράδειγμα, στο iOS, το στοιχείο <View> εμφανίζεται σε UIView, ενώ στο Android μετατρέπεται σε <View>
.

React	React Native
	<View>
	<Text>
,	<FlastList>, child items
<img>	<Image>

2.4 Η γλώσσα προγραμματισμού PHP

Η PHP είναι μια ευρέως χρησιμοποιούμενη γλώσσα προγραμματισμού κατάλληλη για τη δημιουργία δυναμικών ιστοσελίδων. Η γλώσσα αυτή εκτελείται στον server και όχι στον browser, όπως για παράδειγμα η JavaScript. [13]

Αντίθετα από μια συνηθισμένη HTML σελίδα η PHP δεν στέλνεται άμεσα σε έναν πελάτη αλλά αναλύεται και μετά στέλνεται το παραγόμενο αποτέλεσμα. Οι δυνατότητες που μας δίνει είναι απεριόριστες, όπως παραδείγματος χάρη ότι μπορεί να θέση ερωτήματα σε βάσεις δεδομένων, να δημιουργεί εικόνες, να διαβάζει και να γράφει αρχεία, να συνδέεται με απομακρυσμένους υπολογιστές κ.ά.

Δημιουργήθηκε το 1995 από τον RasmusLerdolf ως μια συλλογή από Perl scripts πού τα χρησιμοποιούσε στην προσωπική του σελίδα και η αρχική της ονομασία ήταν PHP/FI από το Forms Interpreter. Το 1997 η PHP/FI έφτασε στην έκδοση 2.0, βασιζόμενη αυτή τη φορά στη γλώσσα C και αριθμώντας περισσότερους από 50.000 ιστότοπους που τη χρησιμοποιούσαν, ενώ αργότερα την ίδια χρονιά οι Andi Gutmans και Zeev Suraski ξανα έγραψαν τη γλώσσα από την αρχή, βασιζόμενοι όμως αρκετά στην PHP/FI 2.0. Στη συνέχεια, οι Zeev και Andi δημιούργησαν την εταιρεία Zend (από τα αρχικά των ονομάτων τους), η οποία συνεχίζει μέχρι και σήμερα την ανάπτυξη και εξέλιξη της γλώσσας PHP.

---

13. Πηγή: PHP - http://www.php.net

Υπάρχουν τρεις κύριοι τομείς που χρησιμοποιείται ένα PHP script.

- Server-side scripting. Αυτό είναι το πιο παραδοσιακό και το κύριο πεδίο για την PHP. Χρειάζονται τρία πράγματα για να δουλέψει αυτό. Τον PHP μεταγλωττιστή (CGI ή server module), ένα webserver (εξυπηρετητή σελίδων) και ένα web browser ("φυλλομετρητή"). Πρέπει να τρέξει ο webserver, με μια συνδεδεμένη εγκατάσταση της PHP. Υπάρχει η δυνατότητα να προσπελαστούν τα αποτελέσματα του PHP προγράμματος με ένα web browser, βλέποντας την σελίδα PHP μέσα από τον server.

- Command line scripting. Υπάρχει η δυνατότητα δημιουργίας ενός PHP script για να τρέχει χωρίς server ή browser. Χρειάζεται μόνο τον PHP μεταγλωττιστή για να την χρησιμοποιηθεί με αυτό τον τρόπο. Αυτός ο τύπος είναι ιδανικός για script που εκτελούνται συχνά με τη χρήση της cron (σε Unix ή Linux) ή με τον Task Scheduler (στα Windows). Αυτά τα script μπορούν επίσης να χρησιμοποιηθούν για απλές εργασίες επεξεργασίες κειμένου.

- Εγγραφή client-side GUI εφαρμογών (Γραφικά περιβάλλοντα χρηστών). Η PHP ίσως να μην είναι η πιο καλή γλώσσα για να γράψει κανείς παραθυρικές εφαρμογές, αλλά αν ξέρετε PHP πολύ καλά και θέλετε να χρησιμοποιήσετε κάποια προχωρημένα χαρακτηριστικά της PHP στις clientside εφαρμογές σας, μπορείτε επίσης να χρησιμοποιήσετε το PHP-GTK για αυτού του είδους τα προγράμματα. Έχετε επίσης τη δυνατότητα να γράφετε cross-platform εφαρμογές με αυτό τον τρόπο. Το PHP-GTK είναι μια επέκταση της PHP και δεν συμπεριλαμβάνεται στην κύρια διανομή. [14]

Τα πλεονεκτήματα της έναντι άλλων γλωσσών όπως Perl, Microsoft Active Server Pages (ASP), Java Server Pages (JSP) είναι τα εξής:
:

- Υψηλή απόδοση
- Διασυνδέσεις με πολλά διαφορετικά συστήματα βάσεων δεδομένων
- Ενσωματωμένες βιβλιοθήκες για πολλές συνηθισμένες Web διαδικασίες
- Χαμηλό κόστος
- Ευκολία μάθησης και χρήσης
- Μεταφερσιμότητα
- Διαθεσιμότητα του κώδικα προέλευσης

---

14. Πηγή: Wikipedia (2020). PHP. Retrieved, from http://el.wikipedia.org/wiki/PHP

2.5 Το Framework LARAVEL

Το Laravel είναι ένα web application framework ανοιχτού κώδικα γραμμένο σε PHP. Δημιουργήθηκε από τον Taylor Otwell και έχει ως στόχο την ανάπτυξη web εφαρμογών που ακολουθούν MVC (Model – View – Controller) αρχιτεκτονική λογισμικού. Μερικά από τα χαρακτηριστικά του είναι το modular packaging system με αυτόνομο dependency manager, η μεγάλη ποικιλία στους τρόπους πρόσβασης σε σχεσιακές βάσεις, αλλά και οι έτοιμες λειτουργίες που διευκολύνουν την ανάπτυξη και συντήρηση εφαρμογών. Η πρώτη beta έκδοση ανακοινώθηκε στις 9 Ιουνίου 2011, ενώ σήμερα αποτελεί ένα από τα πιο γνωστά PHP frameworks μαζί με τα Symfony2, Nette, CodeIgniter και Cake.

Το Laravel επιχειρεί να απλουστεύσει την ανάπτυξη διευκολύνοντας τα κοινά καθήκοντα που χρησιμοποιούνται στην πλειοψηφία των έργων ιστού όπως το authentication, routing, sessions, και caching. Σκοπός του Laravel είναι να καταστήσει τη διαδικασία ανάπτυξης μιας εφαρμογής ευχάριστη για τον προγραμματιστή χωρίς να θυσιάζει τη λειτουργικότητα της εφαρμογής. [15]

---

15. Πηγή: Laravel Framework Documentation - https://laravel.com/docs/

Οι τρόποι με τους οποίους το επιτυγχάνει αυτό είναι οι εξής:

- **Blade Templates**.Το Laravel έχει δική του μηχανή προτύπων που ονομάζεται Blade Templating Engine. Το πλεονέκτημα που διαθέτει η Blade Templating Engine είναι ότι μπορούμε να γράψουμε απλό PHP κώδικα, όπου χαρακτηρίζεται από την κομψή και εύκολη στο διάβασμα σύνταξή του.

- **Composer Ready**. Ο Composer είναι ένα εργαλείο μέσω του οποίου μπορούμε εύκολα να εγκαταστήσουμε στην εφαρμογή μας πακέτα από τρίτους (third-party packages).

- **Services** Η Laravel περιλαμβάνει ένα μεγάλο πλήθος έτοιμων services προς χρήση, όπως: Authentication, Authorization, Artisan Console, Billing, Cache, Collections, Elixir, Encryption, Errors & Logging, Events, Filesystem / Cloud Storage, Hashing, Helpers, Localization, Mail, Package Development, Pagination, Queues, Redis, Session, SSH Tasks, Task Scheduling, Testing, Validation. Παρακάτω περιγράφονται μερικά από τα κύρια services που χρησιμοποιήθηκαν.

- **Authentication** Η Laravel καθιστά την υλοποίηση ενός συστήματος πιστοποίησης πολύ απλή υπόθεση. Για την ακριβεία, σχεδόν τα πάντα παρέχονται έτοιμα. Τα configuration αρχεία βρίσκονται στον φάκελο config/auth.php και περιέχουν πολλές επιλογές για τη διαμόρφωση της συμπεριφοράς του service πιστοποίησης.

- **Artisan Console** Artisan είναι το όνομα του command-line interface που περιέχεται στη Laravel. Παρέχει ένα πλήθος χρήσιμων εντολών για χρήση κατά τη διάρκεια της ανάπτυξης της εφαρμογής. Στηρίζεται στο Symfony Console. 4.5.3 Mail Η Laravel παρέχει ένα απλό API πάνω στη γνωστή SwiftMailer βιβλιοθήκη. Παρέχει drivers για SMTP, Mailgun, Mandrill, Amazon SES καθώς και της συναρτήσεις mail και sendmail της PHP, παρέχοντας έτσι, σχετικά εύκολα, τη δυνατότητα για αποστολή email είτε τοπικά είτε πάνω σε ένα επιλεγμένο cloud service.

- **Query Builder** Ο query builder παρέχει ένα εύχρηστο interface για τη δημιουργία και εκτέλεση queries στη βάση. Μπορεί να χρησιμοποιηθεί για να να υλοποιήσει τις περισσότερες λειτουργίες της εφαρμογής που σχετίζονται με τη βάση και δουλεύει για όλα τα υποστηριζόμενα συστήματα βάσεων.

- **Eloquent ORM**. Η Eloquent περιλαμβάνεται στη Laravel και παρέχει μια απλή υλοποίηση του ActiveRecord για την αλληλεπίδραση με μία βάση. Κάθε πίνακας στη βάση έχει ένα αντίστοιχο Model που χρησιμοποιείται για την αλληλεπίδραση με τη βάση

- **Migrations.** Τα migrations παρέχουν κάποιου είδους έλεγχο έκδοσης για τη βάση, επιτρέποντας σε μια ομάδα να τροποποιήσει και να μοιραστεί εύκολα το σχήμα αυτής. Τα migrations, τυπικά συσχετίζονται με το schema builder της Laravel με σκοπό την εύκολη κατασκευή του σχήματος της βάσης.

- **Database Seeding.** Η Laravel παρέχει έναν απλό τρόπο για τη δημιουργία test δεδομένων στη βάση χρησιμοποιώντας seeding κλάσεις. Αυτές οι κλάσεις είναι αποθηκευμένες στο φάκελο database/seeds.

Οπως προαναφέρθηκε το Laravel είναι ένα MVC framework.
Το MVC (Model-View-Controller) είναι ένα αρχιτεκτονικό μοντέλο λογισμικού για την ανάπτυξη ιστοσελίδων. Είναι επίσης ένα από τα πιο συχνά χρησιμοποιούμενα βιομηχανικά πρότυπα για την δημιουργία κλιμακωτών και επεκτάσιμων έργων.

Η MVC αρχιτεκτονική είναι πολύ διάσημη καθώς απομονώνει τη λογική της εφαρμογής από το επίπεδο της διεπαφής του χρήστη. Σκοπός του είναι να χωρίζει μια εφαρμογή σε τρία βασικά λογικά στοιχεία: το μοντέλο, την προβολή και τον ελεγκτή.
Κάθε ένα από αυτά είναι κατασκευασμένα για να χειρίζονται συγκεκριμένες πτυχές ανάπτυξης μιας εφαρμογής. Αυτή η λογική βοηθάει τον προγραμματιστή στη σωστή οργάνωση του κώδικα και κάνει πιο εύκολη την συντήρησή του.

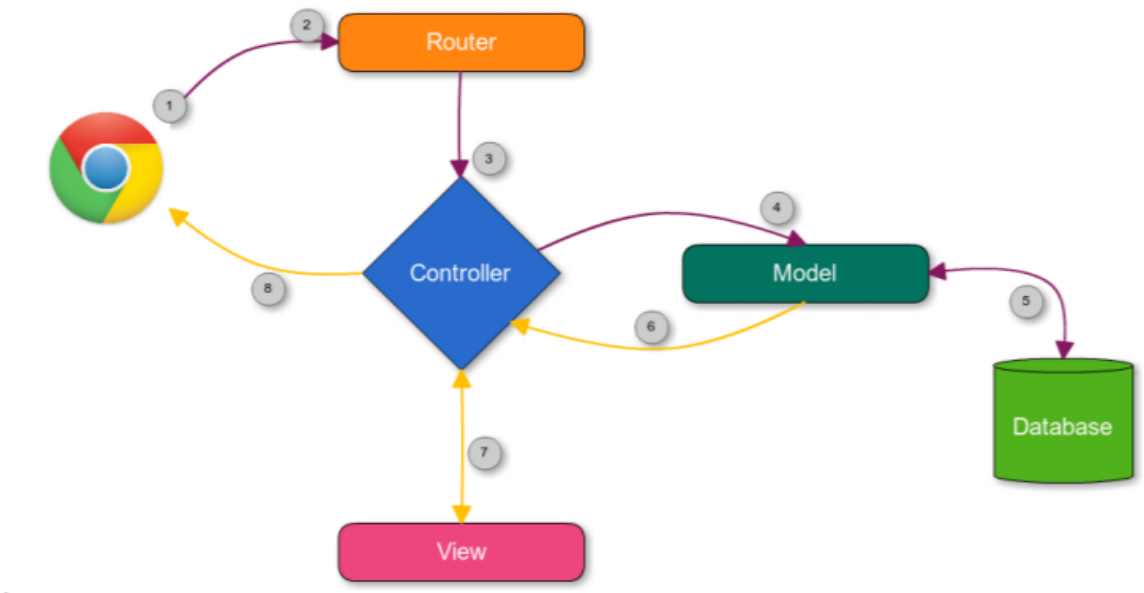
[8] **Το μοντέλο (Model)** είναι οντότητες με τη μορφή δεδομένων στην ιστοσελίδα και ο τομέας στον οποίο είναι χτισμένο το λογισμικό μας. Τα μοντέλα βασίζονται σε πραγματικά στοιχεία όπως ένα πρόσωπο, ένας τραπεζικός λογαριασμός ή ένα προϊόν. Για παράδειγμα, σε ένα ηλεκτρονικό κατάστημα, τα προϊόντα, οι προσφορές και ο χρήστης είναι τα μοντέλα. Αυτά συνδέονται με πίνακες βάσεων δεδομένων και χειρίζονται τις σχέσεις και τις λειτουργίες. Ένα προϊόν μπορεί να έχει κάποιες προσφορές και εκπτώσεις. Τα μοντέλα είναι περισσότερα από αποθηκευμένα δεδομένα. Το μοντέλο προϊόντος πρέπει να έχει ορισμένους περιορισμούς (αριθμός στοιχείων, κόστος, προσφορές υπό όρους). Ένα πρότυπο εκπληρώνει όλες τις προσδοκίες μιας οντότητας.

Η προβολή (View) είναι η οπτική αναπαράσταση ενός μοντέλου και αφορά το περιεχόμενο (σελίδες HTML) που προβάλλεται στον χρήστη. Για παράδειγμα, όταν επισκέπτεστε ένα ηλεκτρονικό κατάστημα, σας παρέχονται λίστες με στοιχεία (προϊόντα), προσφορές, καλαθιού αγορών κλπ. Όλο αυτό το περιεχόμενο είναι δεδομένα (HTML) για εμφάνιση στο πρόγραμμα περιήγησης. Αυτό τα δεδομένα είναι γνωστά ως προβολή στο MVC.

Ο ελεγκτής (Controller) είναι η γέφυρα μεταξύ μοντέλου και προβολής. Διαχειρίζεται το αίτημα του χρήστη με λογική. Για παράδειγμα, σε ένα ηλεκτρονικό κατάστημα, αναζήτηση και φιλτράρισμα προϊόντων, ενημέρωση του καλαθιού, χειρισμός του ελέγχου ταυτότητας χρήστη και ολοκλήρωση της πληρωμής είναι οι μερικοί ελεγκτές. Στο MVC, το περιεχόμενο

HTML και η βάση δεδομένων δεν είναι άμεσα προσβάσιμα για τον εξωτερικό κόσμο. Αυτά εξυπηρετούνται μέσω ελεγκτών. [16]

Δρομολογητής (Router) Η δρομολόγηση είναι η δυνατότητα χαρτογράφησης αιτημάτων URL σε συγκεκριμένα routes. Είναι απλά ένας τρόπος δημιουργίας μιας διεύθυνσης URL.
Στο Laravel, όλα τα αιτήματα χαρτογραφούνται με τη βοήθεια των routes. Η βασική δρομολόγηση μεταφέρει το αίτημα στους αντίστοιχους ελεγκτές (Controllers)



16. Πηγή: Matt Stauffer (2019) Laravel: Up & Running: A Framework for Building Modern PHP Apps

2.6 Websockets

Το WebSocket είναι ένα πρωτόκολλο επικοινωνίας υπολογιστών, το οποίο παρέχει κανάλια πλήρους αμφίδρομης επικοινωνίας μέσω μίας TCP σύνδεσης. Έγινε πρότυπο από τον οργανισμό IETF με τον κωδικό RFC 6455 το 2011.

Το πρωτόκολλο WebSocket υποστηρίζεται σχεδόν σε όλους τους κύριους φυλλομετρητές, όπως οι Google Chrome, Microsoft Edge, Internet Explorer, Firefox, Safari και Opera. Το WebSocket απαιτεί επίσης τις εφαρμογές κινητών στον διακομιστή να το υποστηρίζουν.

Το WebSocket είναι διαφορετικό πρωτόκολλο από το HTTP. Και τα 2 πρωτόκολλα βρίσκονται στο 7ο επίπεδο του Μοντέλου αναφοράς OSI και έτσι εξαρτώνται από το TCP στο επίπεδο 4. Παρ' ότι είναι διαφορετικά, το RFC 6455 δηλώνει ότι το WebSocket "είναι σχεδιασμένο να λειτουργεί πάνω από τις πόρτες 80 και 443 του HTTP αλλά και να υποστηρίζει HTTP proxies και μεσάζοντες" κάνοντάς το έτσι συμβατό με το πρωτόκολλο HTTP. Για να επιτευχθεί η συμβατότητα με το HTTP, η χειραψία του WebSocket χρησιμοποιεί το HTTP Upgrade header για να αλλάξει το HTTP πρωτόκολλο σε WebSocket.

Το WebSocket είναι μία μοναδική σύνδεση υποδοχής (socket connection) ή τερματική σύνδεση,που επιτρέπει ταυτόχρονη και αμφίδρομη (full-duplex, bi-directional) επικοινωνία.

Με το πρωτόκολλο WebSocket, το HTTP αίτημα, μετατρέπεται σε ένα μοναδικό αίτημα, για το άνοιγμα μια σύνδεσης υποδοχής, χρησιμοποιώντας την ίδια σύνδεση τόσο από τον πελάτη προς τον εξυπηρετητή, όσο και από τον εξυπηρετητή προς τον πελάτη.

Με την τεχνολογία WebSocket, μειώνεται σημαντικά η καθυστέρηση επικοινωνίας και ανταλλαγής δεδομένων, διότι από την στιγμή που εγκατασταθεί μια σύνδεση WebSocket, ο εξυπηρετητής μπορεί και στέλνει πίσω στον πελάτη δεδομένα, όσο αυτά είναι διαθέσιμα.

Σε αντίθεση με την τεχνική Polling, με το WebSocket γίνεται ένα μοναδικό αίτημα, ενώ στη συνέχεια, ο εξυπηρετητής δεν χρειάζεται να περιμένει αίτημα από τον πελάτη. Παρόμοια, ο πελάτης μπορεί να στέλνει μηνύματα στον εξυπηρετητή ανά πάσα στιγμή.

Αυτό το μοντέλο αποστολής ενός μοναδικού αιτήματος μειώνει σημαντικά την καθυστέρηση, αντί της τεχνικής Polling, με την οποία στέλνεται ένα αίτημα ανά τακτά χρονικά διαστήματα, ανεξάρτητα των δεδομένων που είναι διαθέσιμα.

Κάθε σύνδεση WebSocket, ξεκινάει με ένα αίτημα HTTP. Αυτό το αίτημα, είναι όπως κάθε άλλο αίτημα HTTP, με τη διαφορά ότι περιλαμβάνει μια επιπλέον ειδική κεφαλίδα - «Upgrade». Η κεφαλίδα «Upgrade» υποδεικνύει, ότι ο πελάτης επιθυμεί να αναβαθμίσει τη

σύνδεση χρησιμοποιώντας διαφορετικό πρωτόκολλο. Σε αυτή την περίπτωση, αυτό το διαφορετικό πρωτόκολλο είναι το WebSocket. [17]

Σε αυτό σημείο, παρουσιάζεται ένα παράδειγμα εκκίνησης σύνδεσης (handshake)

Client request:

GET **/chat HTTP**/1.1

Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
Origin: http://example.com

Server response:

HTTP/1.1 101 **Switching Protocols**
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: chat

Για να ολοκληρωθεί επιτυχώς μια WebSocket χειραψία ώστε να εγκαθιδρυθεί η σύνδεση, θα πρέπει ο εξυπηρετητής να απαντήσει με ένα υπολογισμένο κλειδί. Μια τέτοια απόκριση θα σημαίνει ότι ο εξυπηρετητής καταλαβαίνει πλήρως το WebSocket πρωτόκολλο. Η λειτουργία απόκρισης στην πλευρά του εξυπηρετητή, παίρνει την τιμή της κεφαλίδας «SecWebSocket-Key» που έστειλε ο πελάτης με το αίτημα του και επιστρέφει την υπολογιζόμενη τιμή που ο πελάτης περιμένει στην κεφαλίδα απόκρισης «Sec-WebSocket-Accept». Στη διαδικασία για εγκαθίδρυση σύνδεσης WebSocket και στον υπολογισμό του κλειδιού απόκρισης, το πρωτόκολλο WebSocket, βασίζεται στις κεφαλίδες με το πρόθεμα sec, οι οποίες ορίζονται στο πρωτόκολλο μορφοποίησης RFC 6455 (Fette & Melnikov, 2011).

---

17. Πηγή: websocket Vanessa Wang, Frank Salim, and Peter Moskovits - The Definitive Guide to HTML5 WebSocket – Apress 2013 - http://www.websocket.org/book.html

Όσο μια σύνδεση WebSocket είναι ανοικτή, ο πελάτης και ο εξυπηρετητής μπορούν να στέλνουν μηνύματα ο ένας στον άλλο, ανά πάσα στιγμή. Κάθε μήνυμα WebSocket, έχει έναν κωδικό εντολής, διευκρινίζοντας τον τύπο του ωφέλιμου φορτίου ή πληροφορίας (payload) του μηνύματος. Ο κωδικός εντολής αποτελείται από τα τέσσερα τελευταία bits του πρώτο byte της κεφαλίδας του πλαισίου. Οι κωδικοί εντολών έχουν μια αριθμητική τιμή, όπως περιγράφονται παρακάτω.

Κωδικός εντολής	**Τύπος ωφέλιμου φορτίου του μηνύματος**	**Περιγραφή**
1	Text	Ο τύπος δεδομένων του μηνύματος είναι κείμενο.
2	Binary	Ο τύπος δεδομένων του μηνύματος είναι δυαδικός (δεδομένα σε δυαδική αναπαράσταση)
8	Close	Ο πελάτης ή ο εξυπηρετητής στέλνει μια χειραψία κλεισίματος στον εξυπηρετητή ή στον πελάτη.
9	Ping	Ο πελάτης ή ο εξυπηρετητής στέλνει ένα αίτημα ping στον εξυπηρετητή ή στον πελάτη.
10	Pong	Ο πελάτης ή ο εξυπηρετητής στέλνει ένα αίτημα pong στον εξυπηρετητή ή στον πελάτη.

2.7. REST αρχιτεκτονική

Προκειμένου οι υπολογιστές να συνδεθούν μεταξύ τους μέσω ενός δικτύου και να μπορέσουν να μιλήσουν στην ίδια γλώσσα, εκατοντάδες μέθοδοι έχουν δημιουργηθεί με το πέρασμα των χρόνων. Κάθε μέθοδος ή πρωτόκολλο υλοποιούσε αυτή τη σύνδεση με διαφορετικό τρόπο με αποτέλεσμα να μην ήταν εύκολη η επικοινωνία μεταξύ διαφορετικών συστημάτων. Για να λυθεί αυτό το πρόβλημα, το 2000 ο Roy Fielding δημιούργησε το REpresentational state transfer (REST).

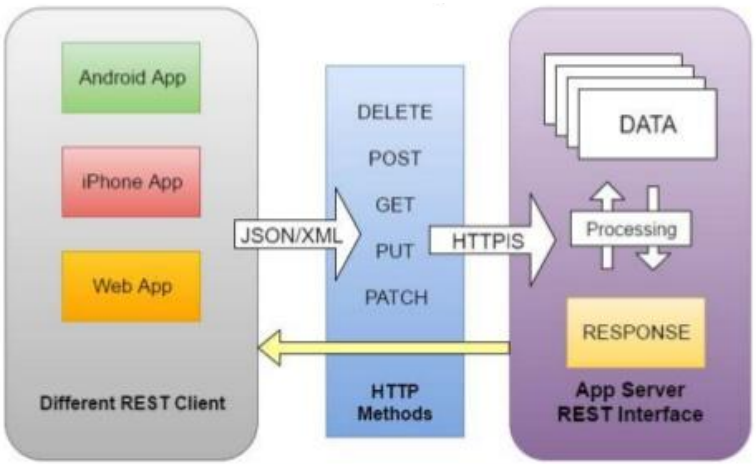
Το REST είναι ένα αρχιτεκτονικό μοντέλο για την παροχή προτύπων επικοινωνίας μεταξύ συστημάτων υπολογιστών στο διαδίκτυο, διευκολύνοντας τα συστήματα να επικοινωνούν μεταξύ τους μέσω του ίδιου μηχανισμού που χρησιμοποιεί το web, αποφεύγοντας πολλές δυσκολίες που είχαν άλλα μεγαλύτερα και πολύπλοκα πρωτόκολλα.

Η αρχή για το πώς δουλεύει το REST είναι παρόμοια με το πώς μια ιστοσελίδα και τα δεδομένα της γίνονται διαθέσιμα στο πρόγραμμα περιήγησης ιστού (browser). Η ιστοσελίδα εμφανίζεται στο browser μέσω μιας διεύθυνσης URL. Ο browser ζητά τα δεδομένα της ιστοσελίδας μέσω μίας αίτησης στο αντίστοιχο URL, λειτουργώντας ως client, και ο server του τα επιστρέφει με μία απάντηση. Άρα, τα δεδομένα βρίσκονται σε κάποια διεύθυνση την οποία ένας υπολογιστής θα χρησιμοποιούσε είτε για να ζητήσει δεδομένα είτε ακόμα και για να στείλει δεδομένα.

Για αυτές τις ενέργειες χρησιμοποιούνται οι αιτήσεις HTTP GET και HTTP POST.
Επίσης, το REST υποστηρίζει και άλλες αιτήσεις, όπως το PUT, DELETE,
OPTIONS, HEAD, TRACE, CONNECT. Αλλά οι πραγματικά πιο σημαντικές αιτήσεις
είναι το GET και το POST.

Το GET μπορεί να χρησιμοποιηθεί για την ανάκτηση οποιουδήποτε είδους δεδομένων παρέχει ένας server και το POST για τη δημιουργία. Με αυτές τις απλές αιτήσεις που ακόμα και ένας απλός browser χρησιμοποιεί καθημερινά, συστήματα τελείως διαφορετικά μεταξύ τους μπορούν να επικοινωνούν με αιτήσεις και απαντήσεις απλά και γρήγορα.

Στο αρχιτεκτονικό μοντέλο του REST, η υλοποίηση του client και η υλοποίηση του server μπορεί να γίνει ανεξάρτητα, χωρίς να γνωρίζει ο ένας για τον άλλο. Αυτό σημαίνει ότι ο κώδικας στην πλευρά του client μπορεί να αλλάξει ανά πάσα στιγμή χωρίς να επηρεαστεί η λειτουργία του server και ο κώδικας στην πλευρά του server μπορεί να αλλάξει χωρίς να επηρεαστεί η λειτουργία του client. Εφόσον κάθε πλευρά γνωρίζει τη μορφή των μηνυμάτων που αποστέλλονται στο άλλο, μπορούν να διατηρηθούν ως μονάδες (modular) και χωριστά. Διαχωρίζοντας τις ευθύνες των clients από τις ευθύνες των servers, βελτιώνουμε την ευελιξία της διασύνδεσης μεταξύ των συστημάτων και βελτιώνουμε την δυνατότητα κλιμάκωσης των servers.



Επίσης, τα συστήματα που ακολουθούν την αρχιτεκτονική REST δεν αποθηκεύουν πληροφορίες για τη κατάσταση προηγούμενων ανταλλαγών μνημάτων μεταξύ client και server (stateless), πράγμα που σημαίνει ότι ο server δεν χρειάζεται να γνωρίζει τίποτα σχετικά με την κατάσταση στην οποία βρίσκεται ο client και το αντίστροφο. Με αυτόν τον

τρόπο, τόσο ο server όσο και ο client μπορούν να καταλάβουν οποιοδήποτε μήνυμα λαμβάνουν, ακόμη και χωρίς να βλέπουν προηγούμενα μηνύματα. Αυτός ο περιορισμός επιβάλλεται μέσω της χρήσης πόρων(URLs) και όχι εντολών(commands). Επειδή τα συστήματα REST αλληλεπιδρούν μέσω τυποποιημένων λειτουργιών, δεν βασίζονται σε υλοποίηση διεπαφών(interfaces). Αυτοί οι περιορισμοί βοηθούν να επιτευχθεί αξιοπιστία, γρήγορη απόδοση και δυνατότητα κλιμάκωσης, αφού τα κομμάτια (components) που αποτελούν τους client και πόσο μάλλον τους servers μπορούν να διαχειρίζονται, να ενημερώνονται και να επαναχρησιμοποιούνται χωρίς να επηρεάζουν το σύστημα στο σύνολό του, ακόμα και κατά τη λειτουργία του συστήματος. Οι συνθήκες που παρέχει το REST είναι ιδανικές για την ανάπτυξη Application
Programming Interfaces (API's).

Ο ποιο συνιθισμενος τυπος API's ειναι της μορφης JSON (JavaScript Object Notation) όπου είναι βασισμένο σε ένα υποσύνολο της γλώσσα προγραμματισμού JavaScript και είναι ένα πρότυπο κειμένου που χρησιμοποιείται για αποθήκευση και ανταλλαγή δεδομένων. Είναι ελαφρύ, απλό, εύκολα κατανοητό από τους ανθρώπους και δεν εξαρτάται από καμιά γλώσσα προγραμματισμού, πράγματα που το έχουν κάνει ευρέως διαδεδομένο. [18]

18. Πηγή: Erik Wilde, Casare Pautasso, REST: From Research to Practice, Springer, 2011.

2.8. MySQL

MySQL είναι ένα Σύστημα Διαχείρισης Σχεσιακής Βάσης Δεδομένων (RDBMS) που αλληλεπιδρά με τον SQL (Structured Query Language), ένα εργαλείο που χρησιμοποιείται για την επεξεργασία δεδομένων και την πρόσβαση σε μια βάση δεδομένων. Είναι ένα πολύ γρήγορο και δυνατό, σύστημα διαχείρισης βάσεων δεδομένων. Μια βάση δεδομένων επιτρέπει αποθήκευση, αναζήτηση, ταξινόμηση και ανάκληση δεδομένων. Ο MySQL διακομιστής ελέγχει την πρόσβαση στα δεδομένα, για να μπορούν να δουλεύουν πολλοί χρήστες ταυτόχρονα, για να παρέχει γρήγορη πρόσβαση και να διασφαλίζει ότι μόνο πιστοποιημένοι χρήστες μπορούν να έχουν πρόσβαση. Συνεπώς, ο MySQL είναι ένας πολυνηματικός διακομιστής πολλαπλών χρηστών.

Ο MySQL server είναι ο διαχειριστής του συστήματος της βάσης δεδομένων.
Χειρίζεται όλες τις εντολές της βάσης. Για να δεχθεί εντολές ο MySQL server πρέπει να τρέχει και να περιμένει αιτήσεις. Γι' αυτό τον λόγο ο MySQL server συνήθως ρυθμίζεται να ξεκινά όταν ο υπολογιστής που τον φιλοξενεί ξεκινά. Όλη η αλληλεπίδραση με τη βάση δεδομένων γίνεται στέλνοντας μήνυμα στον MySQL server.

Η επικοινωνία γίνεται με τη χρήση της SQL (Structured Query Language), η οποία είναι μια γλώσσα που καταλαβαίνει πολλά συστήματα διαχείρισης βάσεων δεδομένων (RDBMS). Η PHP δεν καταλαβαίνει την SQL και αυτό επειδή δεν χρειάζεται να καταλαβαίνει. Η PHP απλά δημιουργεί μια σύνδεση με τον MySQL server για να στείλει ένα μήνυμα SQL. Ο MySQL server ερμηνεύει το μήνυμα SQL και ακολουθεί τις οδηγίες. Έπειτα στέλνει απάντηση στέλνοντας ένα μήνυμα δηλώνοντας την κατάσταση του και τι ενέργειες έκανε ή αναφέρει κάποιο λάθος στην περίπτωση που δεν μπόρεσε να καταλάβει και να ακολουθήσει τις οδηγίες.[19]

---

19. Πηγή: Luke Welling and Laura Thomson (2001), PHP and MySQL Web Development, Sams Publishing

2.9. Apache

Ο Apache HTTP γνωστός και απλά σαν Apache είναι ένας εξυπηρετητής του παγκόσμιου ιστού (web). Όποτε ένας χρήστης επισκέπτεται ένα ιστότοπο το πρόγραμμα πλοήγησης (browser) επικοινωνεί με έναν διακομιστή (server) μέσω του πρωτοκόλλου HTTP, ο οποίος παράγει τις ιστοσελίδες και τις αποστέλλει στο πρόγραμμα πλοήγησης. Ο Apache είναι ένας από τους δημοφιλέστερους εξυπηρετητές ιστού, εν μέρει γιατί λειτουργεί σε διάφορες πλατφόρμες όπως τα Windows, το Linux, το Unix και το Mac OS X. Κυκλοφόρησε υπό την άδεια λογισμικού Apache και είναι λογισμικό ανοιχτού κώδικα. Συντηρείται από μια κοινότητα ανοιχτού κώδικα με επιτήρηση από το Ίδρυμα Λογισμικού Apache (Apache Software Foundation).

Ο Apache χρησιμοποιείται και σε τοπικά δίκτυα σαν διακομιστής συνεργαζόμενος με συστήματα διαχείρισης Βάσης Δεδομένων π.χ. Oracle, MySQL.

Η πρώτη του έκδοση, γνωστή ως NCSA HTTPd, δημιουργήθηκε από τον Robert McCool και κυκλοφόρησε το 1993. Θεωρείται ότι έπαιξε σημαντικό ρόλο στην αρχική επέκταση του παγκόσμιου ιστού. Ήταν η πρώτη βιώσιμη εναλλακτική επιλογή που παρουσιάστηκε απέναντι στον εξυπηρετητή http της εταιρείας Netscape και από τότε έχει εξελιχθεί στο σημείο να ανταγωνίζεται άλλους εξυπηρετητές βασισμένους στο Unix σε λειτουργικότητα και απόδοση. Από το 1996 ήταν από τους πιο δημοφιλείς όμως από τον Μάρτιο του 2006 έχει μειωθεί το ποσοστό της εγκατάστασής του κυρίως από τον Microsoft Internet Information Services και την πλατφόρμα .NET. Τον Οκτώβριο του 2007 το μερίδιο του ήταν 47.73% από όλους τους ιστότοπους. Τον Μάρτιο του 2017, το 49,48% του συνόλου των καταχωρισμένων Ελληνικών τομέων χρησιμοποιούσε το Apache. [20]

20. Πηγή: https://en.wikipedia.org/wiki/Apache_HTTP_Server

3. Ανάπτυξη και Παρουσίαση Εφαρμογής

Σε αυτό το κεφάλαιο θα περιγραφεί η δομή της εφαρμογής σε αρχιτεκτονικό, παρουσιαστικό αλλά και σε λειτουργικό επίπεδο, καθώς και ο τρόπος εκτέλεσης όλων των λειτουργιών αυτού.

3.1.Αρχιτεκτονική δομή εφαρμογής

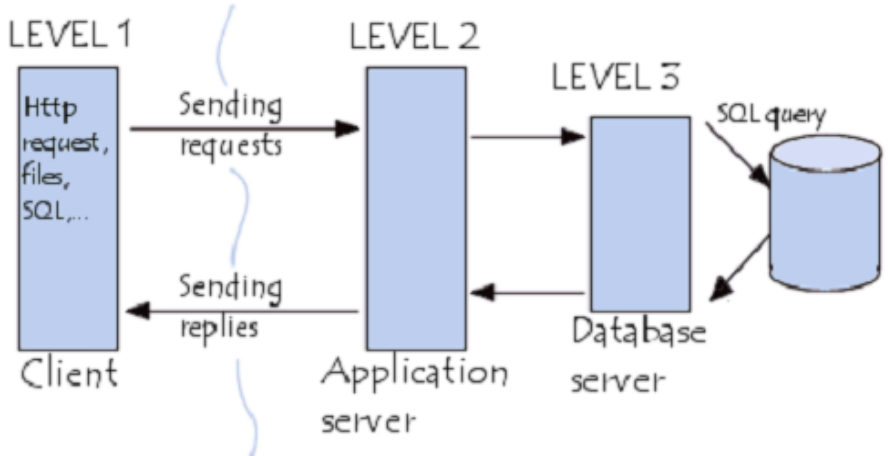
Η εφαρμογή είναι βασισμένη στην αρχιτεκτονική τριών επιπέδων (3-Tier architecture).
Το πρώτο επίπεδο (επίπεδο παρουσίασης, presentation tier) αποτελείται από την ιστοσελίδα, η οποία λειτουργεί ως η διεπαφή προς τον χρήστη.

Η εφαρμογή έχει αναπτυχθεί χρησιμοποιώντας React Native για Android ως βασική πλατφόρμα. Πιο συγκεκριμένα, έχει γίνει χρήση του Context API για state management έναντι του Redux, όλη η CSS είναι custom καθώς δεν έχει χρησιμοποιηθεί κανένα CSS frameworks.

Το δεύτερο (επίπεδο επιχειρησιακής λογικής, business logic tier) αποτελείται από τον REST API server. Ο REST API server έχει αναπτυχθεί χρησιμοποιώντας το Laravel Framework και πιο συγκεκριμένα, έχει γίνει χρήση της γλώσσας PHP, όπου τρέχει σε Apache.

Το τρίτο και τελευταίο επίπεδο (επίπεδο δεδομένων, data tier) αποτελείται από τη βάση δεδομένων και τον κώδικα που επικοινωνεί και διαχειρίζεται αυτή. Ως βάση δεδομένων έχει χρησιμοποιηθεί MySQL.

Η επικοινωνία μεταξύ των επιπέδων γίνεται μόνο μεταξύ γειτονικών επιπέδων. Δηλαδή, το πρώτο επίπεδο (ιστοσελίδα) επικοινωνεί μόνο με το δεύτερο επίπεδο (REST API server), το δεύτερο επίπεδο ως ενδιάμεσο επίπεδο επικοινωνεί και με το πρώτο επίπεδο αλλά και με το τρίτο επίπεδο (κώδικας διαχείρισης βάσης δεδομένων/βάση δεδομένων) και το τρίτο επίπεδο επικοινωνεί μόνο με το δεύτερο επίπεδο.



Εικόνα 6: Απεικόνιση αρχιτεκτονικής τριών επιπέδων

Η αρχιτεκτονική τριών επιπέδων επιλέχθηκε επειδή δίνει αρκετά πλεονεκτήματα και ιδιαίτερη ευελιξία κατά την υλοποίηση αλλά και κατά την επέκταση της εφαρμογής, λόγω της έλλειψης άμεσης σύνδεσης (loose coupling) μεταξύ των επιπέδων.
Αναλυτικότερα, μερικά από αυτά τα πλεονεκτήματα είναι:

- Το λογισμικό που αποτελεί το κάθε επίπεδο μπορεί να γραφτεί σε τελείως διαφορετικές γλώσσες προγραμματισμού, αλλά η επικοινωνία των επιπέδων να πραγματοποιείται χωρίς προβλήματα. Στην περίπτωση αυτής της εφαρμογής, χρησιμοποιήθηκε η React Native στο επίπεδο παρουσίασης (Εφαρμογή), το framework Laravel και η γλώσσα PHP στο επίπεδο επιχειρησιακής λογικής (REST API server) και ο MySQL Server στο επίπεδο δεδομένων.

- Το λογισμικό που αποτελεί το κάθε επίπεδο μπορεί να αναπτυχθεί ξεχωριστά, σε διαφορετικές χρονικές περιόδους ή και παράλληλα, ακόμη και από διαφορετικούς προγραμματιστές.

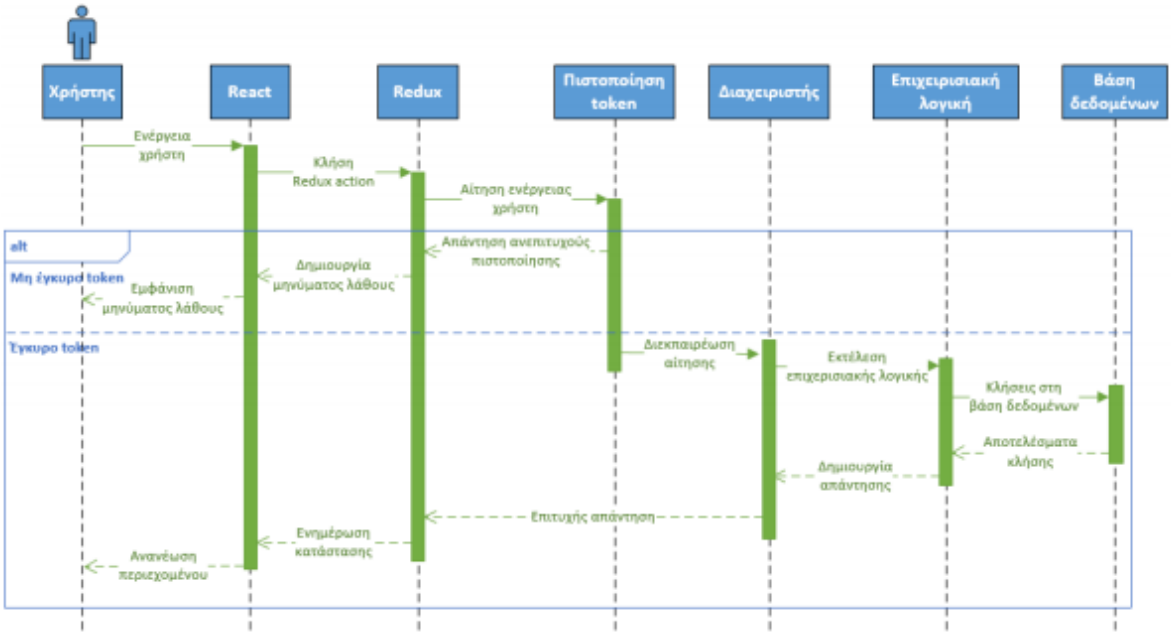
- Στην περίπτωση που χρειάζεται να αλλάξει κάποια λειτουργικότητα (functionality), για παράδειγμα στο επίπεδο επιχειρησιακής λογικής (όπως ο τρόπος υπολογισμού του υπολοίπου ενός λογαριασμού), η αλλαγή θα γίνει μόνο εκεί, και δε θα χρειαστεί να γίνει καμία απολύτως ενέργεια ή αλλαγή στα υπόλοιπα επίπεδα.

Για κάθε ενέργεια του χρήστη στο επίπεδο διεπαφής (εφαρμογή), ακολουθείται η εξής ροή: Μέσω ενός React Native κομματιού κώδικα (component) το
οποίο ανήκει σε μια από τις διαδρομές (routes) που είναι «σπασμένη» η ιστοσελίδα,
ο χρήστης θα ζητήσει μια λειτουργία (πχ πληρωμή).
Το React Native component θα πραγματοποιήσει μία ή και περισσότερες κλήσεις (ανάλογα τη συναλλαγή) στο επίπεδο της επιχειρησιακής λογικής (REST API server) μέσω μιας ενέργειας (action) του reducer του Context στο οποίο ανήκει, πάντα στέλνοντας μαζί και το token πιστοποίησης. Το πρώτο πράγμα που θα κάνει ο REST API server ανεξαρτήτως αίτησης είναι να ελέγξει την εγκυρότητα του token, δηλαδή αν είναι σωστό ή αν έχει λήξει (η μέγιστη διάρκεια ζωής των token έχουν οριστεί σε 10 λεπτά), ώστε να αποφασίσει αν θα επιτραπεί η διεκπεραίωση της αίτησης ή όχι.

Στη συνέχεια, αν το token είναι έγκυρο, ανάλογα την αίτηση που έχει δεχτεί ο REST API server, εκτελείται η κατάλληλη επιχειρησιακή λογική, γίνονται οι απαραίτητες κλήσεις προς το επίπεδο δεδομένων (βάση δεδομένων) με τη χρήση του Laravel Framework και αν όλα έχουν πάει καλά ετοιμάζεται η κατάλληλη HTTP απάντηση επιτυχίας, διαφορετικά ετοιμάζεται η κατάλληλη απάντηση αποτυχίας, μαζί με ένα κατάλληλο μήνυμα για τον χρήστη.

Αν το token δεν είναι έγκυρο, τότε ετοιμάζεται η HTTP απάντηση μη έγκυρης πιστοποίησης.

Έπειτα, αποστέλλεται η απάντηση στο επίπεδο διεπαφής (Εφαρμογής), η οποία λαμβάνεται από το action του Context, το οποίο εν ακολουθία ανανεώνει την κατάσταση (state) του reducer που πραγματοποίησε την αίτηση, μέσω του component της React.
Τέλος, τα περιεχόμενα του component από το οποίο ξεκίνησε όλη η διαδικασία και οποιουδήποτε άλλου συμμετέχει στη συναλλαγή ανανεώνονται αυτόματα λόγω της αλλαγής της κατάστασης του reducer και ο χρήστης βλέπει το αποτέλεσμα στην οθόνη του.



Εικόνα 7: Διάγραμμα ακολουθίας ενέργειας χρήστη

Αυτή η διαδικασία πραγματοποιείται ανεξαιρέτως για όλες τις λειτουργίες της εφαρμογής, από τη μικρότερη μέχρι και τη μεγαλύτερη.

3.2.REST API Server

Όλη η λειτουργικότητα του Server έχει χωριστεί σε «λειτουργικές περιοχές», URLs οι οποίες προκύψανε από την τμηματοποίηση και ομαδοποίηση όλων των λειτουργιών σύμφωνα με την οντότητα την οποία αφορούν. Οι λειτουργικές περιοχές που αναγνωρίστηκαν με αυτό τον τρόπο είναι οι εξής:

- login
- logout
- refresh
- register
- categories
- images
- invoices

- orders
- printers
- products
- tables
- users

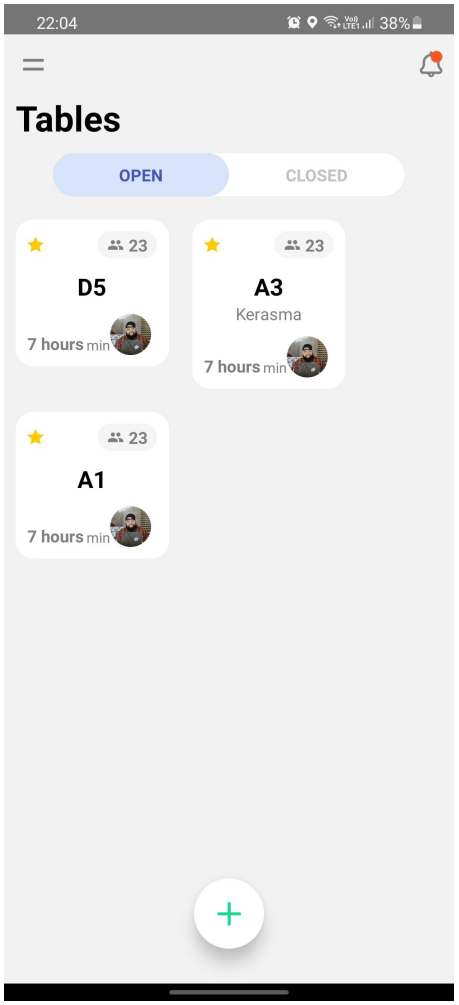
Για κάθε μια από αυτές τις λειτουργικές περιοχές, έχει φτιαχτεί και ένας διαχειριστής (controller) ο οποίος περιέχει τα υλοποιημένα σημεία εισαγωγής (endpoints) της κάθε λειτουργικής περιοχής, που γίνονται διαθέσιμα μέσω του REST API server, πίσω από τα οποία είναι υλοποιημένη όλη η επιχειρησιακή λογική.

Λειτουργική περιοχή	Τύπος HTTP κλήσης	Περιγραφή κλήσης/σημείο εισαγωγής	Παράμετροι
api/auth/login	POST	Πιστοποίηση χρήστη	username και κωδικό χρήστη
api/auth/logout	POST	Αποσύνδεση χρήστη	JWT
api/auth/refresh	POST	Ανανέωσή κλειδιού	JWT
api/auth/register	POST	Εγγραφή χρήστη	json
api/categories	GET	κατηγορίες προϊόντων	-
api/categories	POST	δημιουργία κατηγορίας	json
api/categories/{id}	GET	συγκεκριμένη κατηγορία	id κατηγορίας
api/categories/{id}	DELETE	διαγραφή κατηγορίας	id κατηγορίας
api/categories/{id}	PUT	ενημέρωσή κατηγορίας	id κατηγορίας
api/images	POST	μεταφόρτωσή εικόνας	-
api/images/{file}	GET	προβολή εικόνας	filename
api/invoices	GET	προβολή όλων των χρεώσεων	-
api/invoices/{id}	GET	προβολή χρέωσης	id χρεώσεις
api/orders	GET	προβολή όλων των παραγγελιών	-
api/orders/{id}	GET	προβολή παραγγελία	id παραγγελίας
api/printer	POST	Εγγραφή νέου printer	json
api/printer/{id}	GET	Προβολή printer	id printer

api/printer/{id}	PUT	Ενημέρωσή printer	id printer
api/printer/{id}	DELETE	Διαγραφή printer	id printer
api/printers	GET	Προβολή όλων των printers	-
api/products	GET	Προβολή όλων των προϊόντων	-
api/products	POST	Εγγραφή προϊόντος	json
api/products/{id}	GET	Προβολή προϊόντος	id προϊόντος
api/products/{id}	DELETE	Διαγραφή προϊόντος	id προϊόντος
api/products/{id}	PUT	Ενημέρωση προϊόντος	id προϊόντος
api/tables	POST	Εγγραφή Τραπεζίου	json
api/tables	GET	Προβολή όλων των Τραπεζίων	-
api/tables/closed	GET	Προβολή ανοικτών τραπεζιών	-
api/tables/open	GET	Προβολή κλειστών τραπεζίων	-
api/tables/{id}	PATCH	Ενημέρωσή τραπεζίου	id
api/tables/{id}	DELETE	Διαγραφή τραπεζίου	id τραπεζίου
api/tables/{id}	GET	Προβολή τραπεζίου	id τραπεζίου
api/tables/{id}	POST	Εγγραφή τραπεζίου	id τραπεζίου (json)
api/tables/{id}/invoices	POST	Εγγραφή χρεώσεις στο τραπέζι	id τραπεζίου (json)
api/tables/{id}/orders	POST	Εγγραφή παραγγελίας στο τραπέζι	id τραπεζίου (json)
api/tables/{id}/orders	GET	Εγγραφή παραγγελιών του τραπεζίου	id τραπεζίου
api/tables/{id}/state	POST	Εγγραφή Σταδίου του τραπεζίου	id τραπεζίου (json)
api/users	POST	Εγγραφή χρήστη	json
api/users	PATCH	Ενημέρωσή χρήστη	json

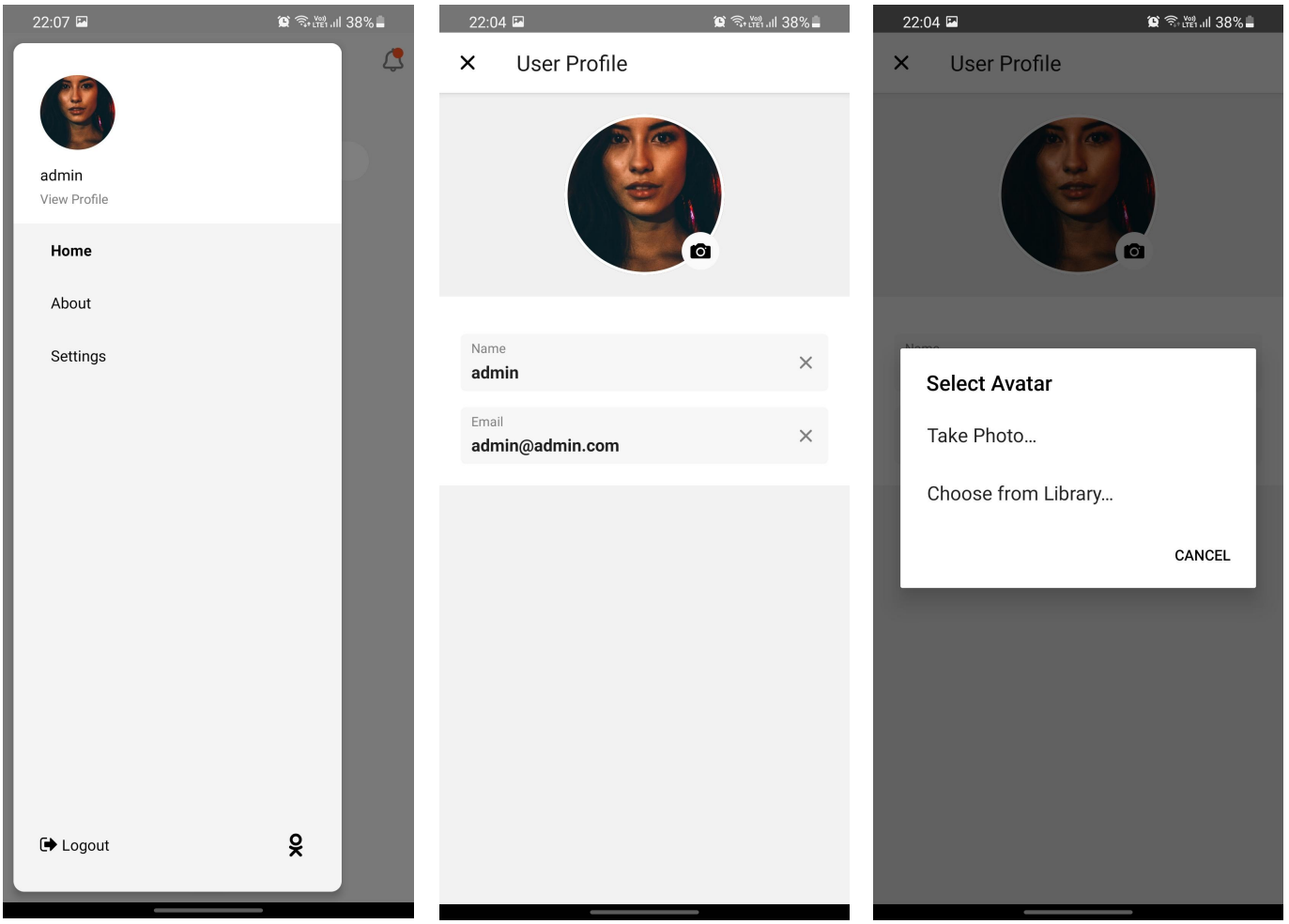
3.3. Δομή Εφαρμογής

Η εφαρμογή έχει υλοποιηθεί σε React Native και αποτελείται από την αρχική οθόνη όπου απεικονίζονται τα τραπέζια τα οποία βρίσκονται εν ενεργεία αυτήν την στιγμή καθώς και τα τραπέζια τα οποία έχουν κλείσει (έχουν ολοκληρωθεί). Η μετάβαση γίνεται μέσω μιας απλής κίνησης του χεριού Ένα πλήκτρο με το οποία μεταβαίνει στην οθόνη για την δημιουργία ενός νέου τραπεζίου καθώς και το πλήκτρο του κεντρικού menu.

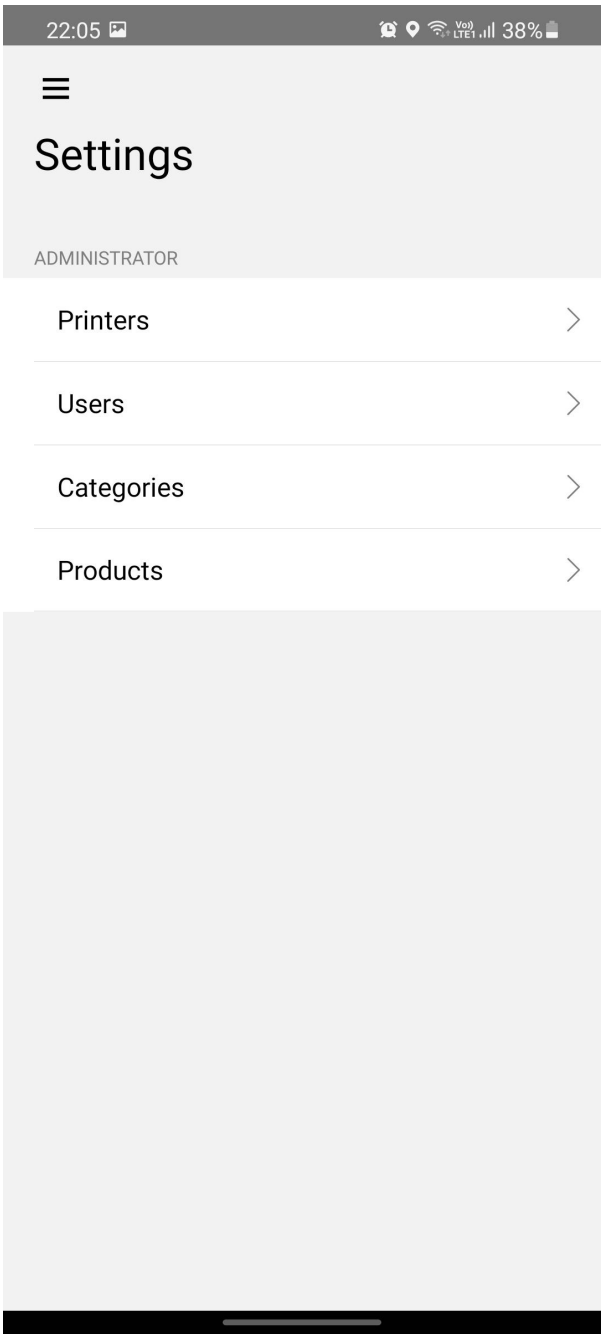


Να σημειωθεί ότι η επικοινωνία γίνεται μέσω websocket σε πραγματικό χρόνο, που σημαίνει πως οποιοσδήποτε χρήστης της εφαρμογής κάνει οποιαδήποτε αλλαγή, αυτή θα εμφανιστεί κατευθείαν σε όλες τις υπόλοιπες συσκευές, χωρίς να χρειαστεί οποιαδήποτε ενέργεια από τους χρήστες. Αυτό είναι πάρα πολύ σημαντικό στο να βρίσκονται όλες οι συσκευές insync με τα δεδομένα του server και όλες οι συσκευές να βρίσκονται στο ίδιο state. Αυτό είναι το μεγαλύτερο πλεονέκτημα αυτής της εφαρμογής. Εκμηδενίζονται τα λάθη στο φυσικό κόσμο τα οποία οφείλονται σε μεταχρονισμένα δεδομένα. πχ Έστω ότι ο σερβιτόρος κλείνει ένα τραπέζι, αυτό χωρίς επικοινωνία πραγματικού χρόνου θα είχε ως αποτέλεσμα οι υπόλοιπες συσκευές να μην έχουν ενημερωθεί και θα δείχνουν το τραπέζι ακόμα ανοιχτό μέχρι ως ότου κάνουν αίτημα στον server να τους σταλούν τα νέα δεδομένα όπου θα φέρουν και τις αλλαγές του τραπεζίου.

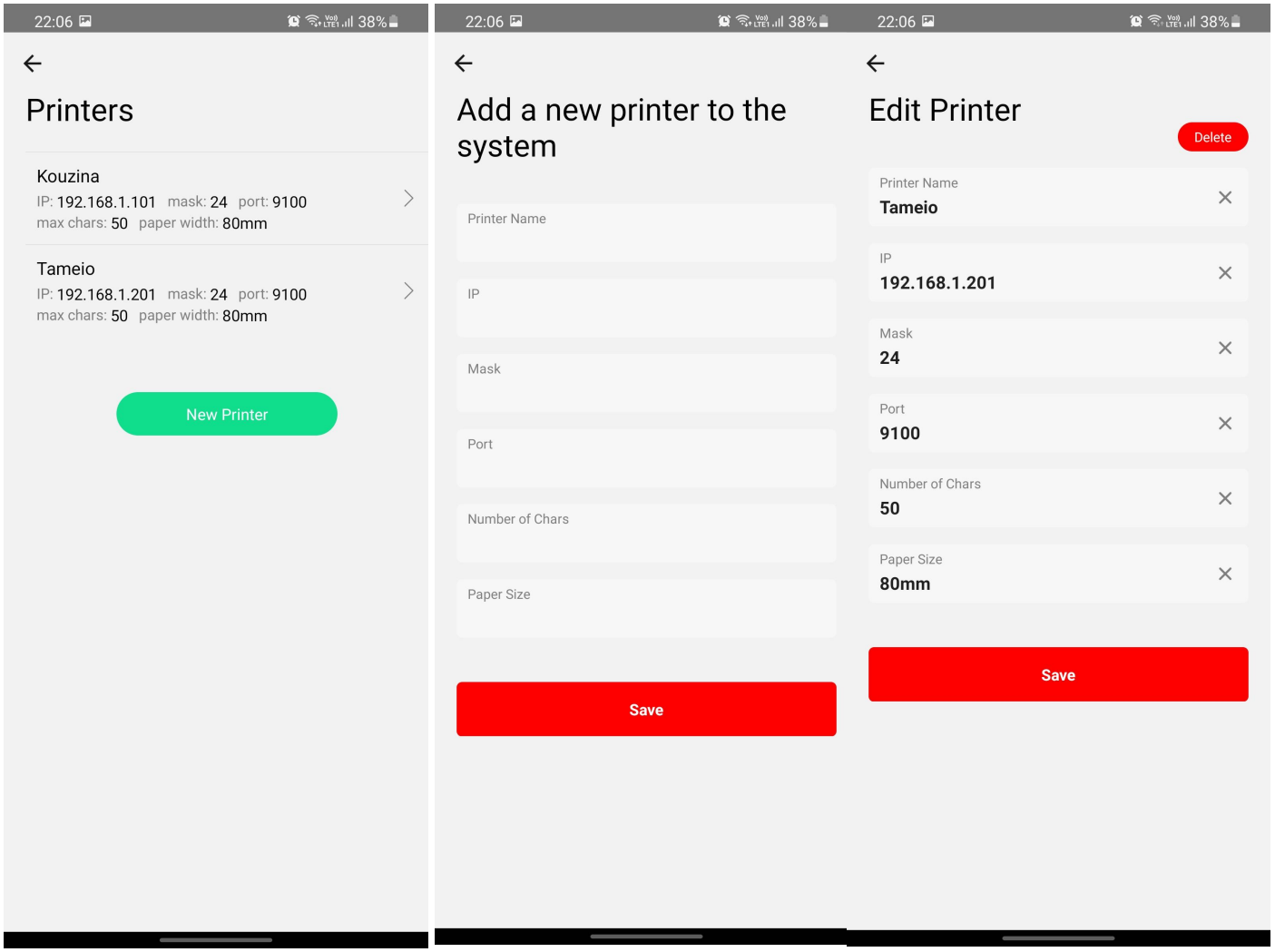
Το Component των τραπεζίων φέρουν επιπλέων πληροφορίες πέραν του ονόματος του τραπεζίου όπως την ώρα που έχει μεσολαβήσει από τη δημιουργία του τραπεζίου, ώστε να γνωρίζει το προσωπικό τον χρόνο που έχει περάσει. Αναγράφεται ακόμα το πλήθος των ατόμων που κάθονται στο τραπέζι, σχόλιο για το συγκεκριμένο τραπέζι καθώς και τον σερβιτόρο ο οποίος έκανε τη δημιουργία του τραπεζίου.



Πατώντας στο κεντρικό μενού εμφανίζεται ένας drawer με την επιλογή για τη μετάβαση στις ρυθμίσεις. Καθώς και στο προφίλ του χρήστη όπου μπορεί να αλλάξει τη φωτογραφία που απεικονίζεται επιλέγοντας την από τη συλλογή της συσκευής η τραβώντας μια νέα φωτογραφία εκείνη την στιγμή, ακόμα να αλλάξει το όνομα του και το email του.



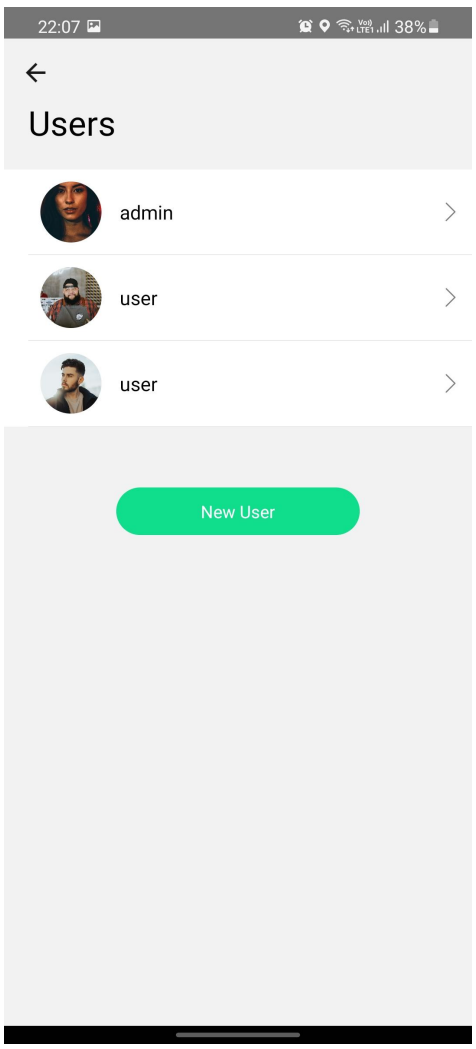
Στην σελίδα των ρυθμίσεων υπάρχουν οι επιλογές για τη ρύθμισή των εκτυπωτών οπού θα εκτυπώνονται οι παραγγελίες.
Εισαγωγή χρηστών, κατηγοριών καθώς και προϊόντων.

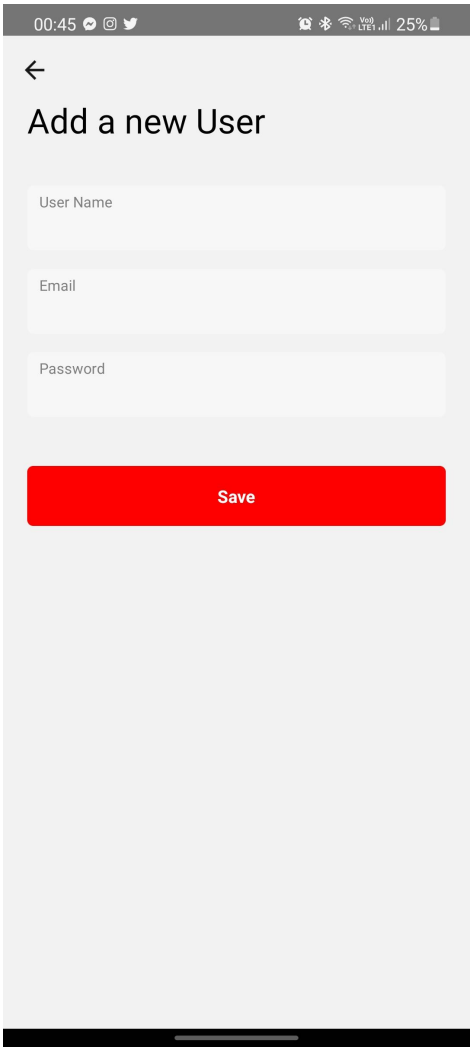


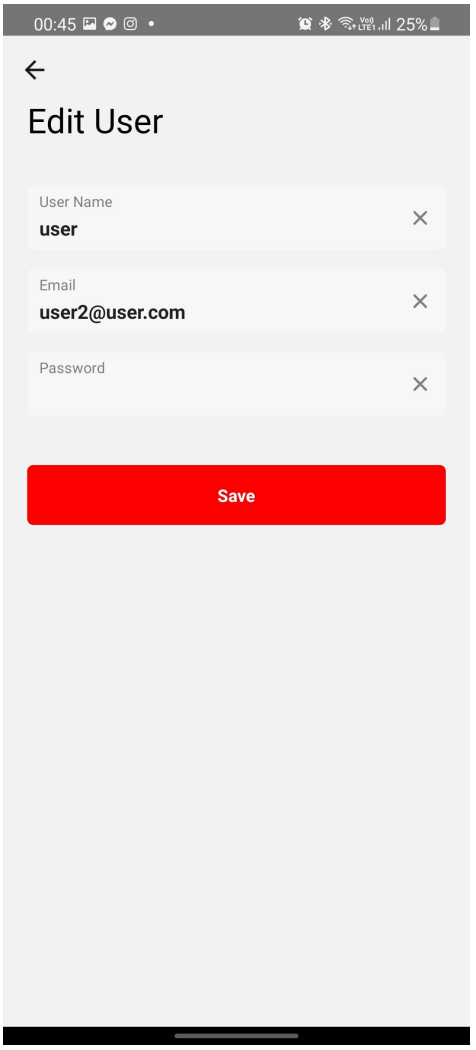
Επιλέγοντάς στις ρυθμίσεις την επιλογή των εκτυπωτών, μεταβαίνει στη σελίδα όπου εμφανίζονται όλοι οι εκτυπωτές καθώς και η επιλογή για δημιουργία νέου.
Πατώντας στους Ίδη υπάρχων εκτυπωτές μεταβαίνει στις σελίδα της τροποποίησής του εκάστοτε εκτυπωτή, όπου υπάρχει και η επιλογή της διαγραφής του.
Πατώντας στην εισαγωγή νέου εκτυπωτή μεταβαίνει στη σελίδα για την εισαγωγή νέου, όπου συμπληρώνονται τα πεδία.

- Όνομά
- IP εκτυπωτή που βρίσκεται στο τοπικό δίκτυο
- Port επικοινωνίας
- Μέγιστο πλήθος χαρακτήρων (σε πλάτος)
- Πλάτος του χαρτιού σε mm

Οι τροποποίησες επικυρώνονται με το πάτημα του save.



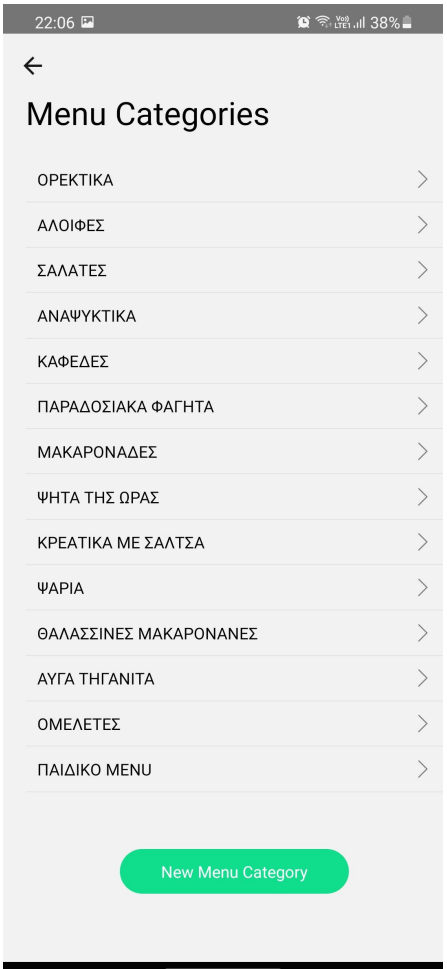


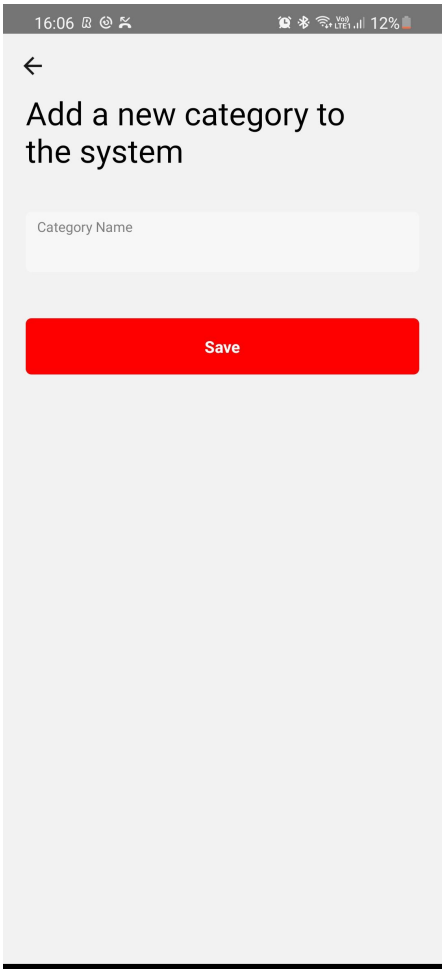


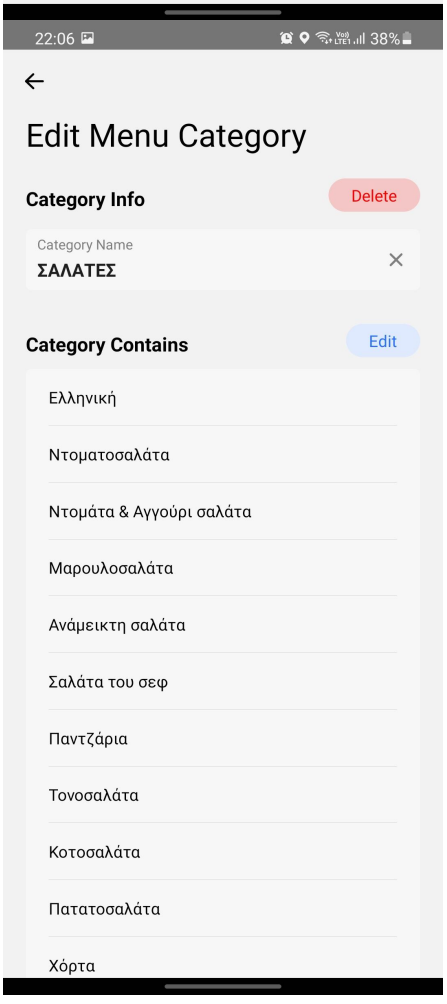
Επιλέγοντάς στις ρυθμίσεις την επιλογή των χρηστών, μεταβαίνει στη σελίδα όπου εμφανίζονται όλοι οι χρήστες καθώς και η επιλογή για δημιουργία νέου.
Πατώντας στους ήδη υπάρχων χρήστες μεταβαίνει στις σελίδα της τροποποίησής του εκάστοτε χρήστη.
Πατώντας στην εισαγωγή νέου χρήστη μεταβαίνει στη σελίδα για την εισαγωγή νέου, όπου συμπληρώνονται τα πεδία.

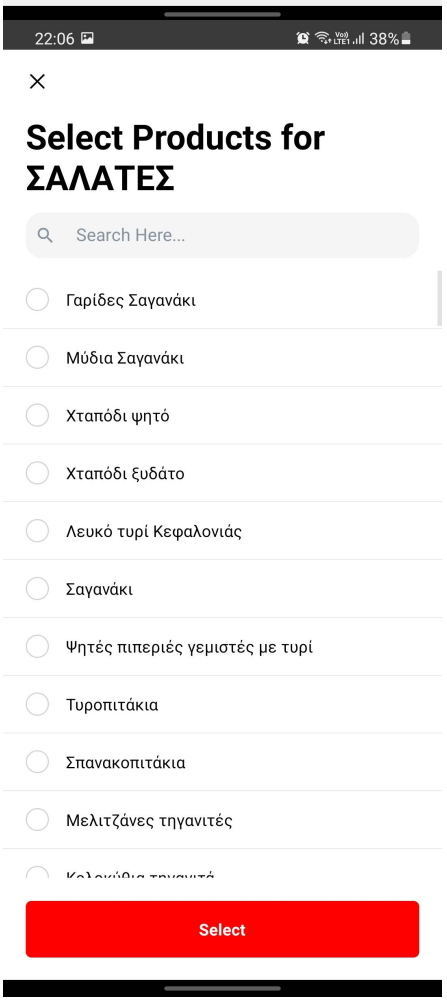
- Όνομά χρήστη
- email
- κωδικό

Οι τροποποίησής επικυρώνονται με το πάτημα του save.


22:06
38%
Menu Categories
ΟΡΕΚΤΙΚΑ
ΑΛΟΙΦΕΣ
ΣΑΛΑΤΕΣ
ΑΝΑΨΥΚΤΙΚΑ
ΚΑΦΕΔΕΣ
ΠΑΡΑΔΟΣΙΑΚΑ ΦΑΓΗΤΑ
ΜΑΚΑΡΟΝΑΔΕΣ
ΨΗΤΑ ΤΗΣ ΩΡΑΣ
ΚΡΕΑΤΙΚΑ ΜΕ ΣΑΛΤΣΑ
ΨΑΡΙΑ
ΘΑΛΑΣΣΙΝΕΣ ΜΑΚΑΡΟΝΑΝΕΣ
ΑΥΓΑ ΤΗΓΑΝΙΤΑ
ΟΜΕΛΕΤΕΣ
ΠΑΙΔΙΚΟ MENU
New Menu Category


16:06
12%
Add a new category to the system
Category Name
Save


22:06
38%
Edit Menu Category
Category Info
Delete
Category Name
ΣΑΛΑΤΕΣ
Category Contains
Edit
Ελληνική
Ντοματοσαλάτα
Ντομάτα & Αγγούρι σαλάτα
Μαρουλοσαλάτα
Ανάμεικτη σαλάτα
Σαλάτα του σεφ
Παντζάρια
Τονοσαλάτα
Κοτοσαλάτα
Πατατοσαλάτα
Χόρτα

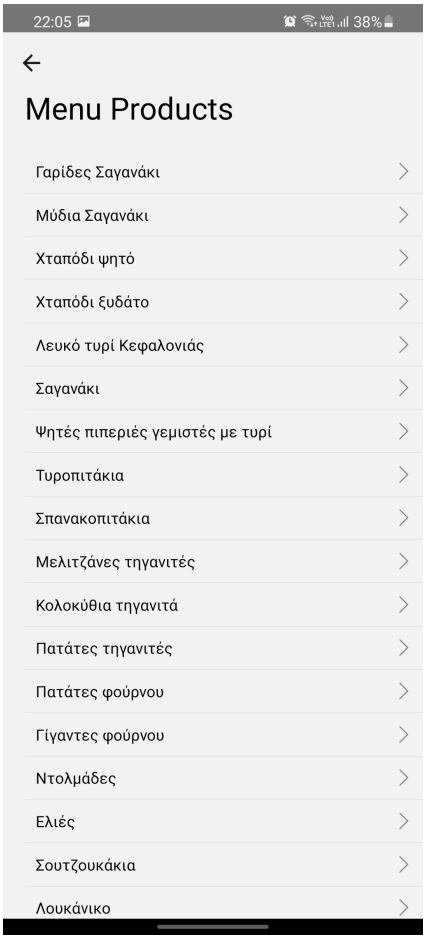

22:06
38%
Select Products for ΣΑΛΑΤΕΣ
Search Here...
Γαρίδες Σαγανάκι
Μύδια Σαγανάκι
Χταπόδι ψητό
Χταπόδι ξυδάτο
Λευκό τυρί Κεφαλονιάς
Σαγανάκι
Ψητές πιπεριές γεμιστές με τυρί
Τυροπιτάκια
Σπανακοπιτάκια
Μελιτζάνες τηγανιτές
Select

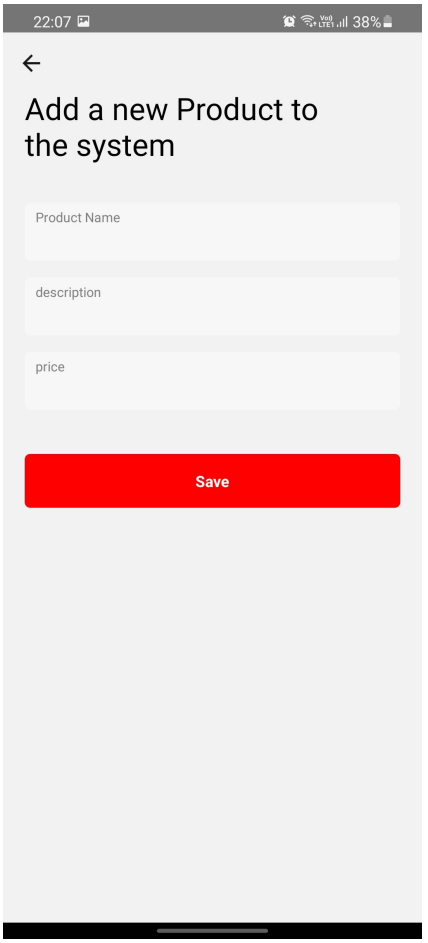
Επιλέγοντας στις ρυθμίσεις την επιλογή των κατηγοριών, μεταβαίνει στη σελίδα οπού εμφανίζονται όλες οι κατηγορίες καθώς και η επιλογή για τη δημιουργία νέας.
Πατώντας στις ήδη υπάρχων κατηγορίες μεταβαίνει στις σελίδα της τροποποίησης της εκάστοτε κατηγορίας.
Πατώντας στην εισαγωγή νέας κατηγορίας μεταβαίνει στη σελίδα για την εισαγωγή νέας, όπου συμπληρώνονται το πεδίο.

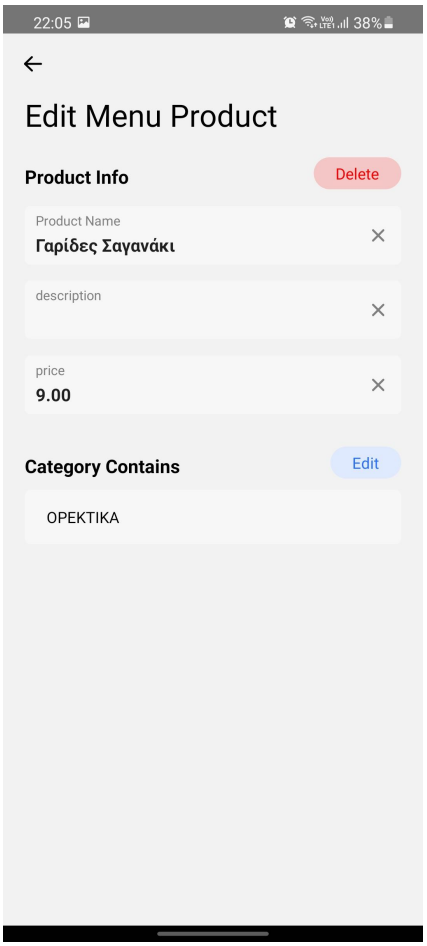
- Όνομά κατηγορίας

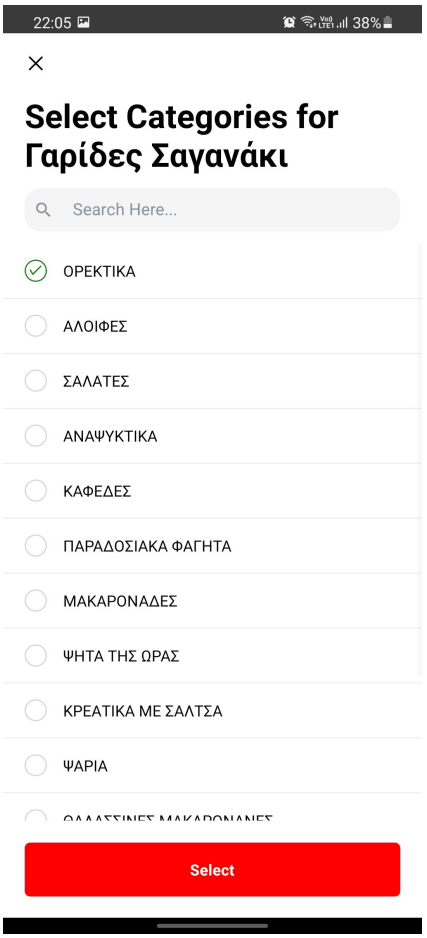
Στην τροποποίηση της κατηγορίας μπορούμε να εισάγουμε τα προϊόντα που θα φέρει η κάθε κατηγορία. Η διαδικασία επιλογής είναι πάρα πολύ, αρκεί να ελεγχθεί το προϊόν πατώντας πάνω του, επίσης υπάρχει η δυνατότητα αναζήτησης μέσω μια μπάρας όπου γράφοντάς τα πρώτα γράμματα από τη λέξη ξεκίνα να κάνει φιλτράρισμα των προϊόντων και να απεικονίζει μόνο τα σχετικά. Επομένως, σε μεγάλο πλήθος προϊόντων η αναζήτησή του γίνεται πολύ απλά, να σημειωθεί ότι γίνεται πολλαπλή επιλογή, για την αφαίρεση ενός η περισσοτέρων προϊόντων αρκεί να πατήσουμε ξανά στο επιλεγμένο για την αφαίρεση του.
Υπάρχει και η δυνατότητα διαγραφείς της κατηγορίας όπου φυσικά κατά τη διαγραφή της τα προϊόντα που φέρει δε διαγράφονται.
Ένα προϊόν μπορεί να ανήκει σε πολλές κατηγορίες.

Οι τροποποίησής επικυρώνονται με το πάτημα του save.


22:05
38%
Menu Products
Γαρίδες Σαγανάκι
Μύδια Σαγανάκι
Χταπόδι ψητό
Χταπόδι ξυδάτο
Λευκό τυρί Κεφαλονιάς
Σαγανάκι
Ψητές πιπεριές γεμιστές με τυρί
Τυροπιτάκια
Σπανακοπιτάκια
Μελιτζάνες τηγανιτές
Κολοκύθια τηγανιτά
Πατάτες τηγανιτές
Πατάτες φούρνου
Γίγαντες φούρνου
Ντολμάδες
Ελιές
Σουτζουκάκια
Λουκάνικο


22:07
38%
Add a new Product to the system
Product Name
description
price
Save


22:05
38%
Edit Menu Product
Product Info
Delete
Product Name
Γαρίδες Σαγανάκι
description
price
9.00
Category Contains
Edit
ΟΡΕΚΤΙΚΑ

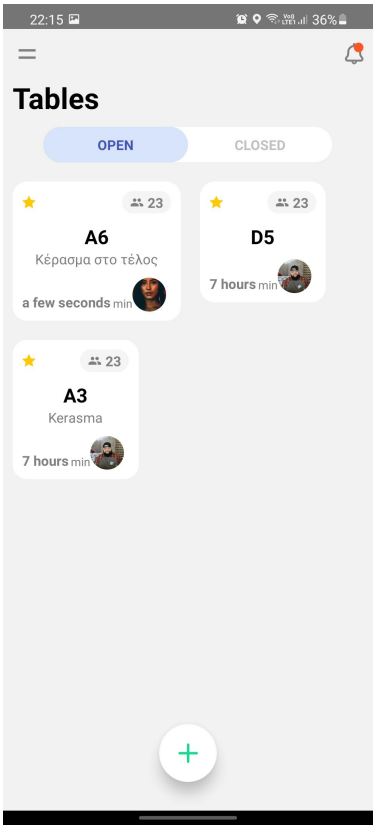

22:05
38%
Select Categories for Γαρίδες Σαγανάκι
Search Here...
ΟΡΕΚΤΙΚΑ
ΑΛΟΙΦΕΣ
ΣΑΛΑΤΕΣ
ΑΝΑΨΥΚΤΙΚΑ
ΚΑΦΕΔΕΣ
ΠΑΡΑΔΟΣΙΑΚΑ ΦΑΓΗΤΑ
ΜΑΚΑΡΟΝΑΔΕΣ
ΨΗΤΑ ΤΗΣ ΩΡΑΣ
ΚΡΕΑΤΙΚΑ ΜΕ ΣΑΛΤΣΑ
ΨΑΡΙΑ
Select

Επιλέγοντάς στις ρυθμίσεις την επιλογή των προϊόντων, μεταβαίνει στη σελίδα όπου εμφανίζονται όλα τα προϊόντα καθώς και η επιλογή για τη δημιουργία νέου
Πατώντας στα ήδη υπάρχων προϊόντα μεταβαίνει στη σελίδα της τροποποίησής του εκάστοτε προϊόντος.
Πατώντας στην εισαγωγή νέου προϊόντος μεταβαίνει στη σελίδα για την εισαγωγή, όπου συμπληρώνονται τα πεδία.

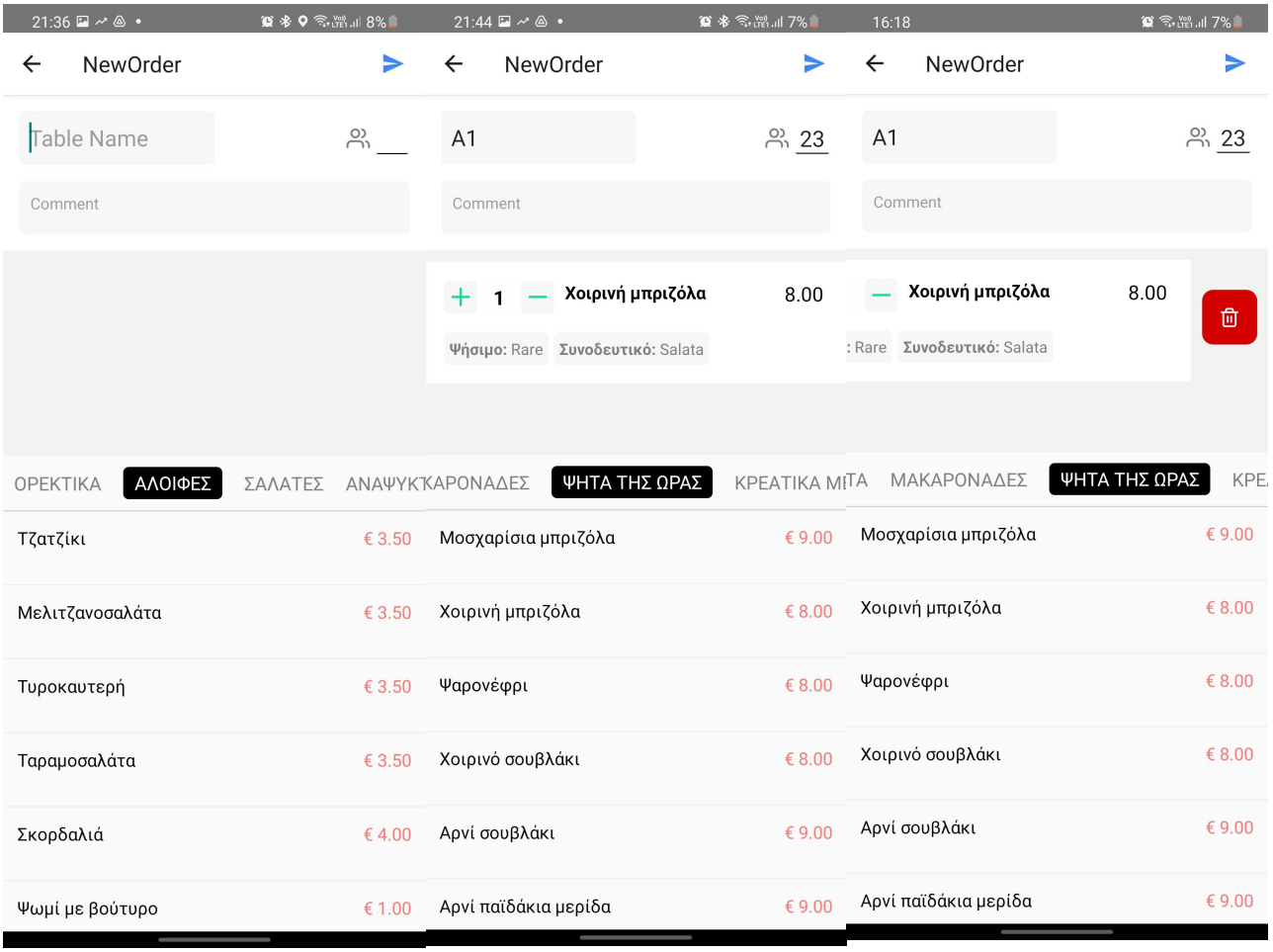
- Όνομά προϊόντος
- Περιγραφή προϊόντος
- Τιμή

Στην τροποποίηση του προϊόντος μπορούμε να εισάγουμε τις κατηγορίες που θα ανήκει το προϊόν. Η διαδικασία επιλογής είναι πάρα πολύ απλή, αρκεί να επιλεχθεί το προϊόν πατώντας πάνω του, επίσης υπάρχει η δυνατότητά αναζήτησής μέσω μια μπάρας οπού γράφοντάς τα πρώτα γράμματά από τη λέξη ξεκίνα να κάνει φιλτράρισμα των κατηγοριών και να απεικονίζει μόνο τα σχετικά. Επόμενος σε μεγάλο πλήθος κατηγοριών η αναζήτησή γίνεται πολύ άπλα, να σημειωθεί ότι γίνεται πολλαπλή επιλογή, για την αφαίρεση ενός η περισσοτέρων κατηγοριών αρκεί να πατήσουμε ξανά πάνω στην επιλεγμένη για την αφαίρεση της.
Υπάρχει και η δυνατότητα διαγραφείς του προϊόντος.
Ένα προϊόν μπορεί να ανήκει σε πολλές κατηγορίες.

Οι τροποποίησής επικυρώνονται με το πάτημα του save



Πίσω στην αρχική οθόνη επιλέγοντάς ένα από τα τραπέζια μεταβαίνει στην οθόνη προβολής των παραγγελιών του συγκεκριμένου τραπεζίου.



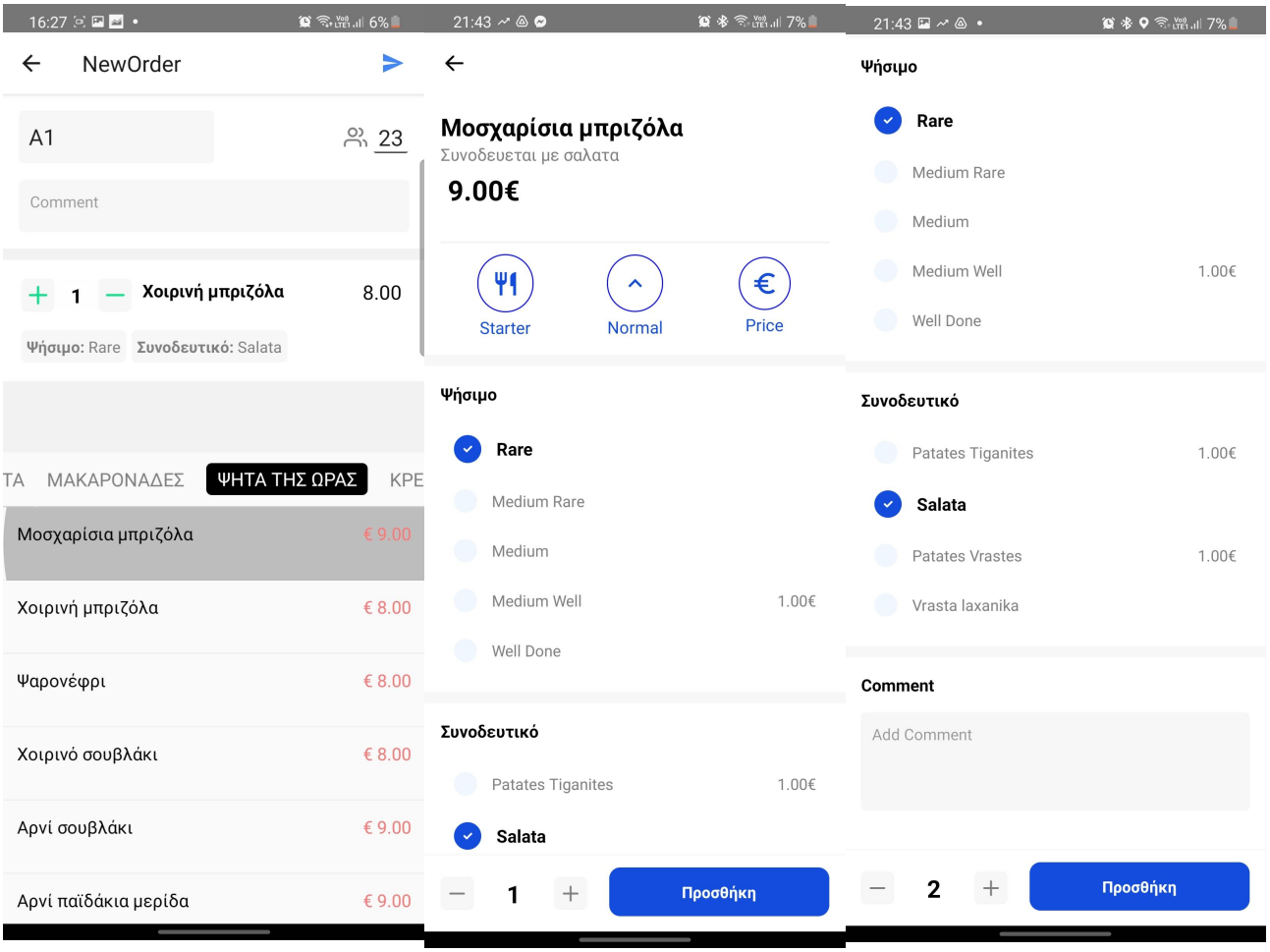
Κατά τη νέα παραγγελία ο σερβιτόρος εισάγει το όνομά του τραπεζίου το πλήθος των πελατών και τυχόν σχόλια για το τραπέζι.

Στο κάτω μισό μέρος της οθόνης βρίσκεται το menu του καταστήματος. Στο άνω μέρος βρίσκονται οι κατηγορίες που δημιουργήθηκαν από τις ρυθμίσεις και τα προϊόντα που φέρουν.

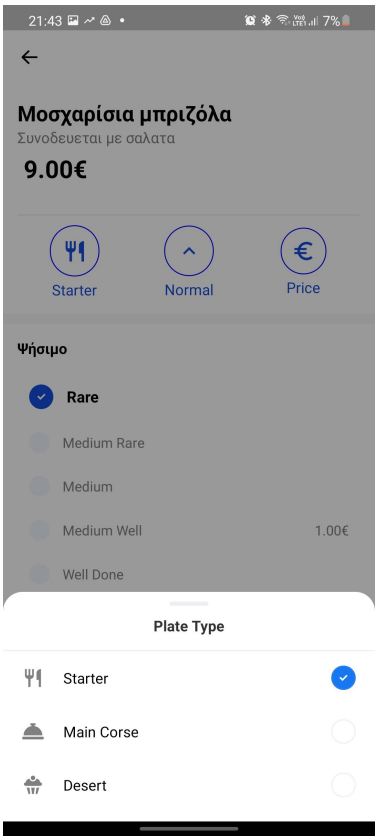
Πατώντας πάνω στο προϊόν εισάγετε στο "καλάθι" της παραγγελίας.

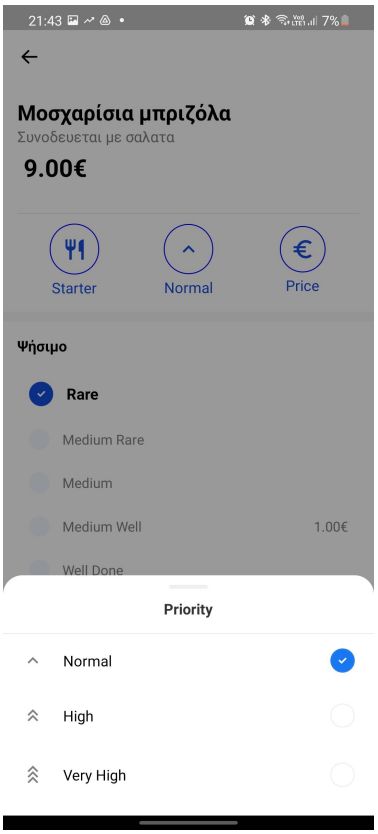
Για την αφαίρεση ενός προϊόντα αρκεί να κάνουμε swipe αριστερά και ένα κουμπί διαγραφείς εμφανίζεται οπού αφαιρεί το προϊόν από το καλάθι.

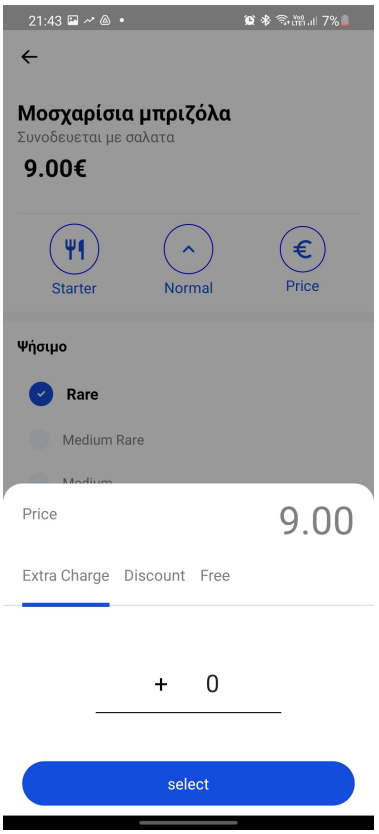
Επίσης, υπάρχει η δυνατότητα να αυξηθεί ή να μειωθεί το πλήθος των προϊόντων για γρήγορη τροποποίηση της παραγγελίας.

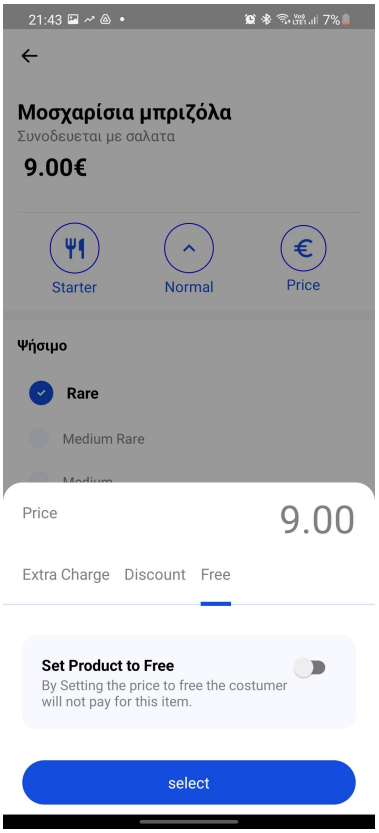


Πατώντας παρατεταμένα στο προϊόν του μενού μεταβαίνει στις επιλογές του προϊόντος.
Εδώ ο σερβιτόρος μπορεί να επιλέξει ανάμεσα στις επιλογές που δίνονται για το κάθε προϊόν.Μπορεί ακόμα ο σερβιτόρος να προσθέσει μια περιγραφή για το προϊόν πχ. όχι, αλάτι.
Υπάρχει η δυνατότητα να αυξομειώνει το πλήθος και πατώντας προσθήκη να το εισάγει στην παραγγελία.



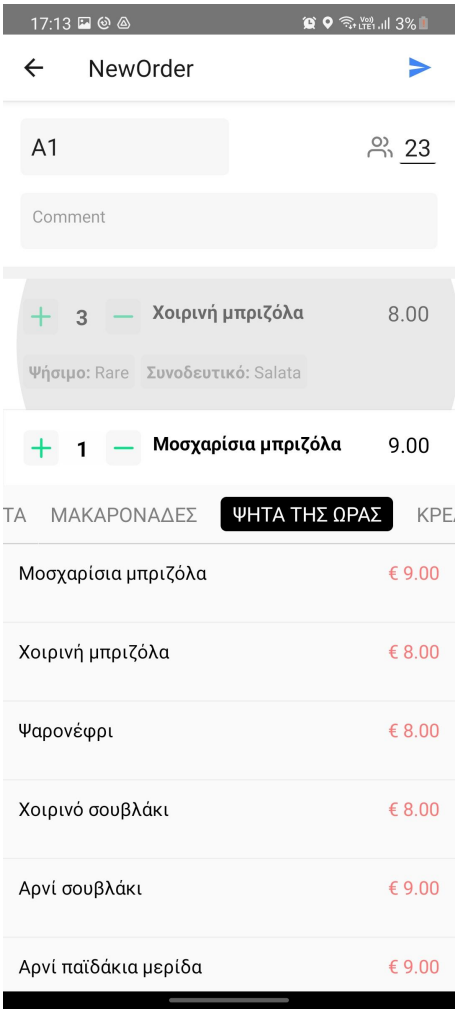


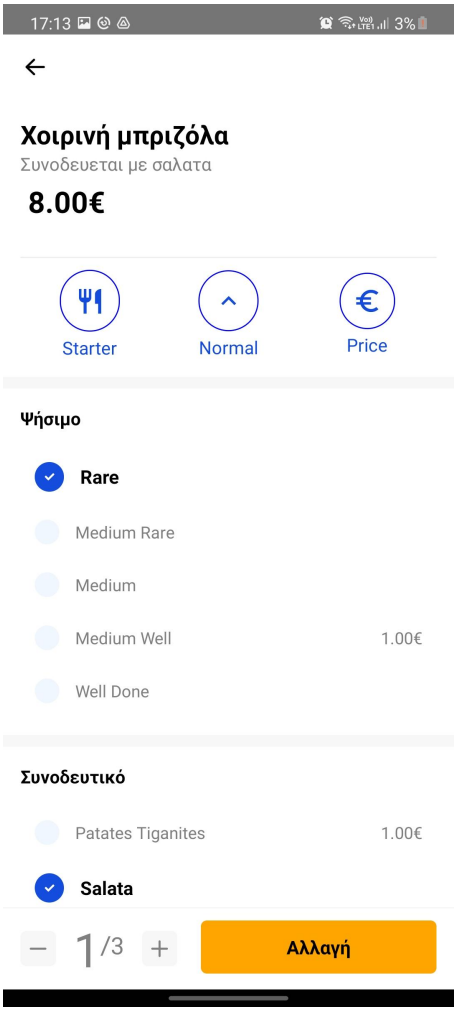


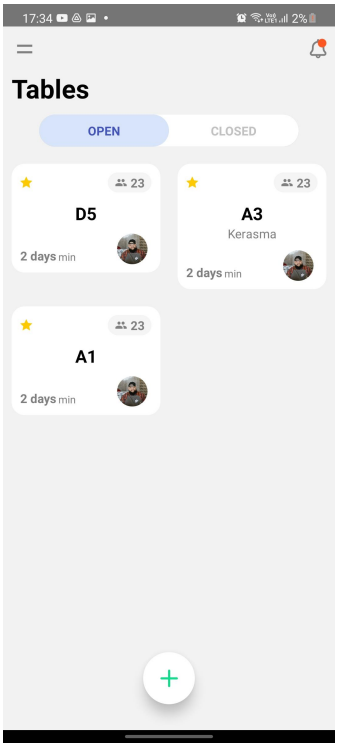


Στο πάνω μέρος βρίσκονται τρία κουμπιά. Το πρώτο επιλέγεται ο τύπος του πιάτου δλδ αν ο πελάτης το θέλει σαν κύριος, πρώτο ή επιδόρπιο. Στο δεύτερο κουμπί επιλέγεται η προτεραιότητα που θα έχει, κανονική, υψηλή, πολύ υψηλή.
Στο τρίτο καθορίζεται η τιμή, αν δλδ ζήτησή κάτι επιπλέον ο πελάτης μπορεί να χρεωθεί από τον σερβιτόρο, όπως επίσης να κάνει έκπτωση ή ακόμα και να κεράσει το πιάτο.

Τέλος, πατώντας προθήκη εισάγετε το προϊόν στην παραγγελία και μεταβαίνει στην προηγούμενη οθόνη του καλαθιού της παραγγελίας.

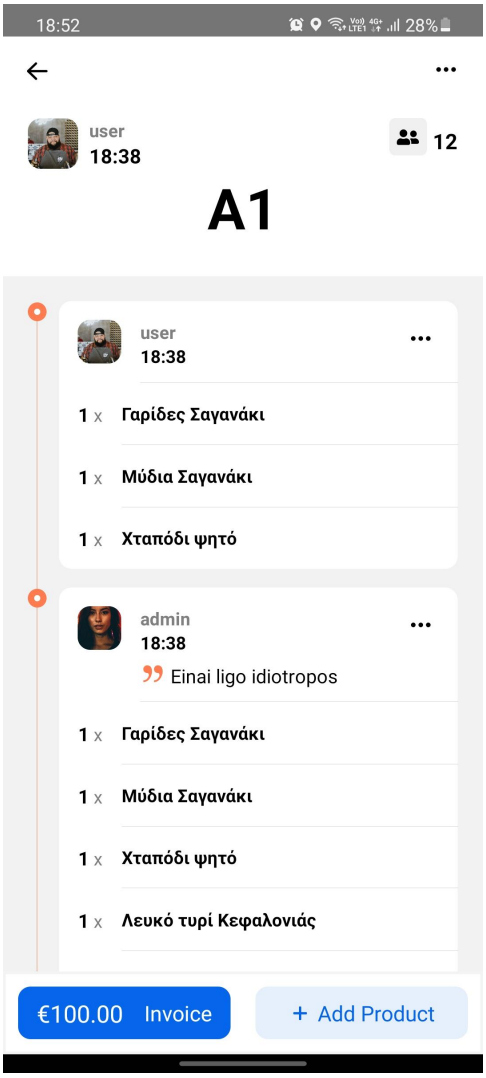


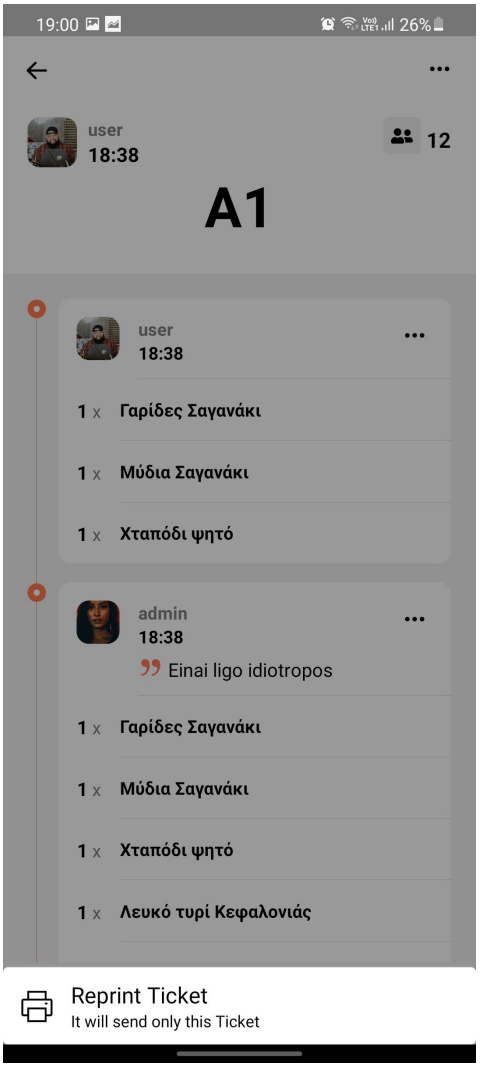




Πατώντας παρατεταμένα στο προϊόν που βρίσκεται στην παραγγελία υπάρχει η δυνατότητά τροποποίησής οπού μεταβαίνει στην οθόνη τροποποίησης αλλά στο κάτω μέρος επιλέγεις από το ήδη υπάρχον πλήθος προϊόντων σε ποια από αυτά θα γίνουν οι αλλαγές. Πολύ σημαντική λειτουργία διότι γλιτώνει πολύ χρόνο στον σερβιτόρο.

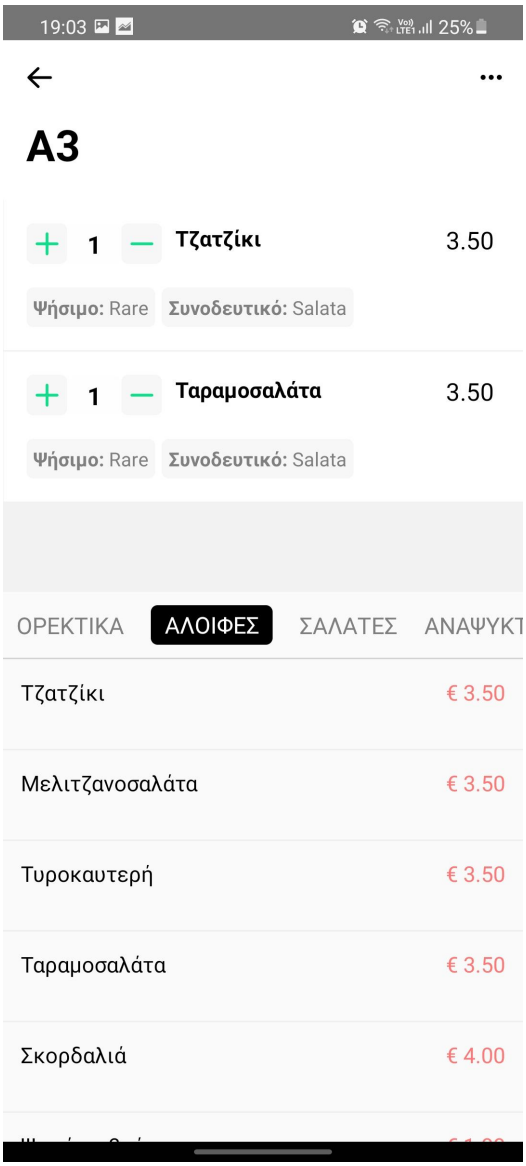
Αφού ολοκληρωθεί η παραγγελία πατώντας πάνω δεξιά η παραγγελία αποστέλλεται στον Σέρβε, εκτυπώνετε στην κουζίνα και μεταβαίνει πάλι στην αρχική οθόνη



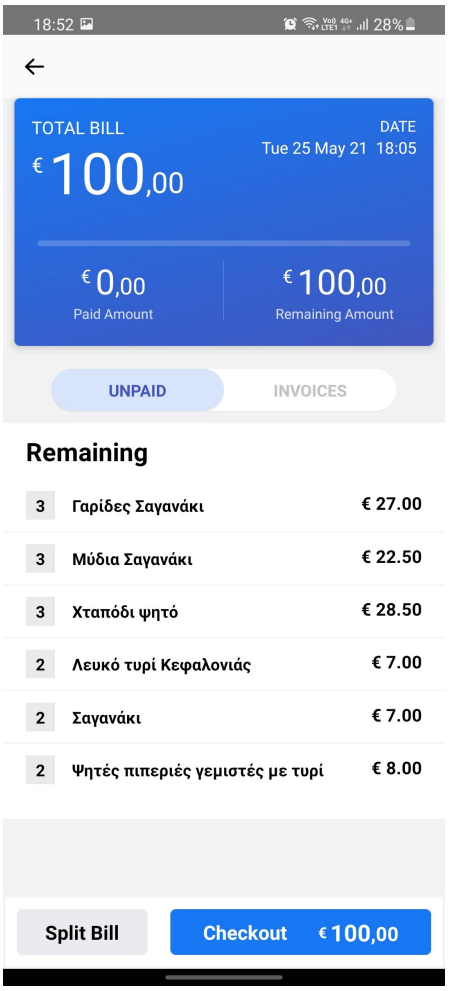


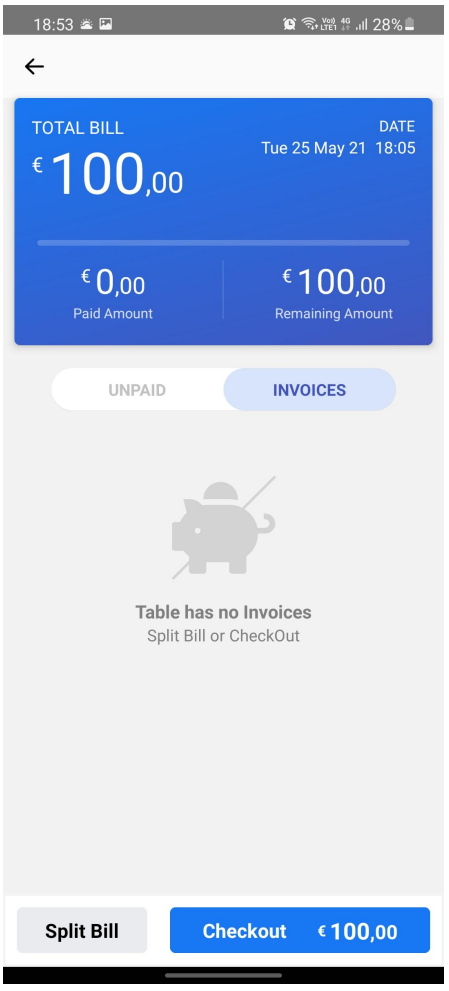
Στην οθόνη του τραπεζίου εμφανίζονται όλες οι παραγγελίες που έχουν γίνει.
Αρχικά εμφανίζεται πάνω αριστερά ο χρήστης ο οποίος "άνοιξε" το τραπέζι (η πρώτη παραγγελία) και την ώρα.
Κάθε παραγγελία που προστίθεται φέρει τον σερβιτόρο που την έκανε, τα προϊόντα καθώς και τυχόν σχόλια που μπορεί να προσθέσει ο σερβιτόρος.
Όταν αποστέλλεται μια παραγγελία εκτυπώνεται και στην κουζίνα από έναν θερμικό εκτυπωτή.
Πατώντας πάνω στις τελείες εμφανίζεται ένα pop up από το τέλος της σελίδας με animation και δίνεται η επιλογή της προεκτύπωσης.

Στο τέλος της σελίδα βρίσκονται δυο κουμπιά το αριστερό απεικονίζει το σύνολο του τραπεζίου και πατώντας το μεταβαίνει στην οθόνη για την τιμολόγηση ενώ το αριστερό κουμπί μεταβαίνει στην οθόνη για να προστεθούν extra προϊόντα στο τραπέζι κατά παραγγελία του πελάτη.

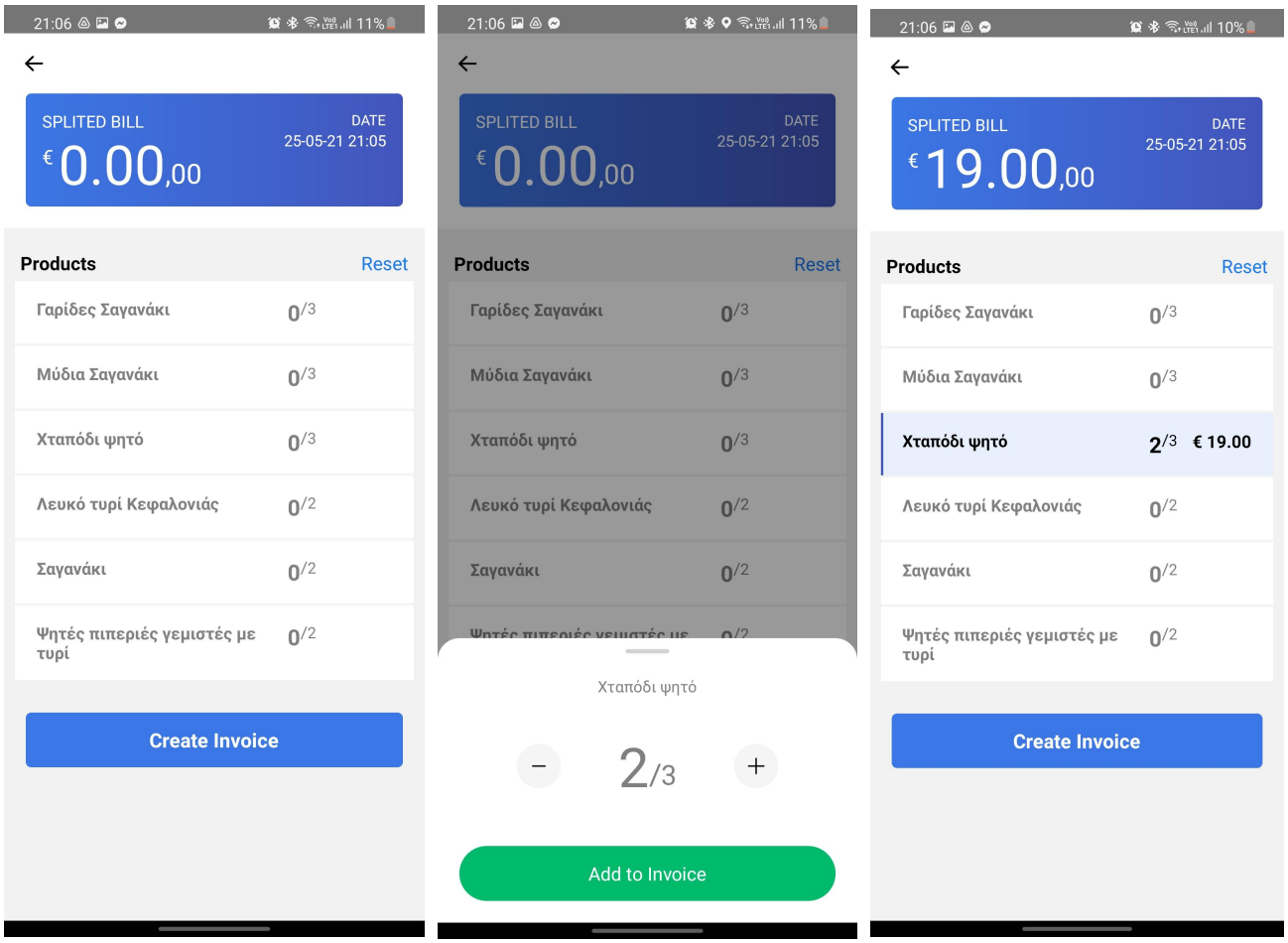


Πατώντας να προστεθεί νέα παραγγελία στο τραπέζι μεταβαίνει στην οθόνη της παραγγελία με όλες τις λειτουργίες που αναφέρθηκαν προηγούμενος. Ο σερβιτόρος παίρνει την παραγγελία και αυτή προστίθεται στο τραπέζι.

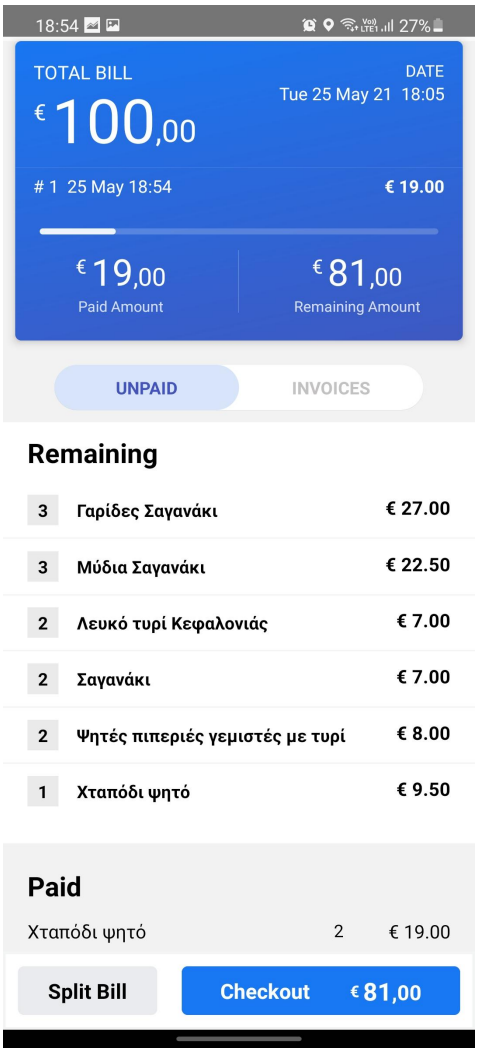


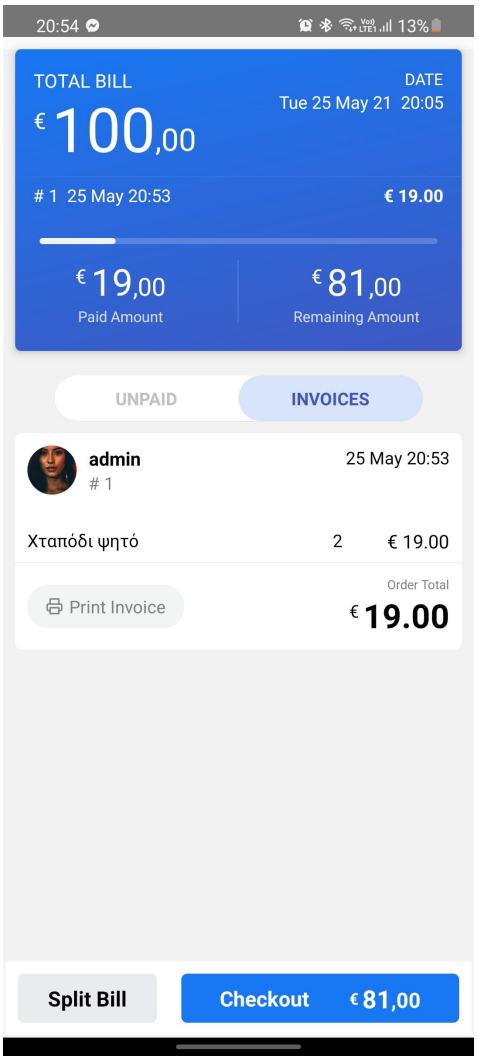


Πατώντας το κουμπί invoice μεταβαίνει την οθόνη της τιμολόγησης εμφανίζεται το σύνολο του τραπεζίου καθώς και τα προϊόντα με την τιμή τους. Κάνοντας swipe μεταβαίνει στα τιμολόγιά. Υπάρχουν δυο επιλογές, πληρωμή ολοκλήρου του λογαριασμού ή διαχωρισμό του λογαριασμού ώστε να μπορεί ο κάθε πελάτης ενώ βρίσκεται σε κοινό τραπέζι να πληρώνει τα δικά του προϊόντα.

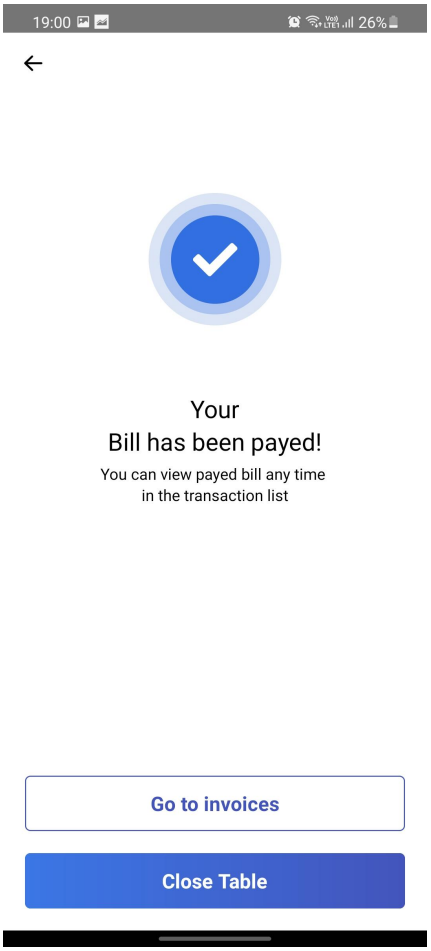


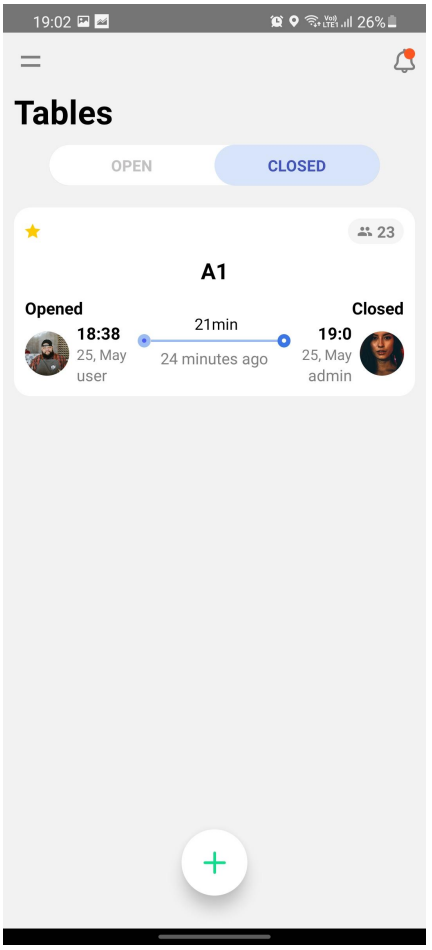
Πατώντας στον διαχωρισμό του λογαριασμού εμφανίζονται όλα τα προϊόντα τα οποία δεν έχουν πληρωθεί και πατώντας πάνω τους ένα pop up slider εμφανίζεται με την επιλογή του πλήθους του συγκεκριμένου προϊόντος που ο πελάτης θέλει να πληρώσει.
Επίσης, στο πάνω μέρος της οθόνης εμφανίζεται το ποσό που πρόκειται να πληρωθεί.
Πατώντας create invoice δημιουργείται ένα τιμολόγιο και τα προϊόντα αφαιρούνται από το τραπέζι και ταυτόχρονα επιστρέφει στην οθόνη της τιμολόγησης





Πλέον εμφανίζεται στη σύνοψη του λογαριασμού το ποσό που έχει πληρωθεί, το ποσό που απονέμεται και μια μπάρα όπου δείχνει ποσοστιαία το ποσό που έχει πληρωθεί. Επίσης, εμφανίζονται τα πληρωμένα προϊόντα και αυτά που απομένουν καθώς και στα invoices κάνοντας swipe εμφανίζεται το τι έχει πληρωθεί το ποιος σερβιτόρος πληρώθηκε και την ώρα. Υπάρχει ακόμα και η δυνατότητα προεκτύπωσης του λογαριασμού που έχει πληρωθεί.





Πατώντας στο CheckOut δημιουργείται ένα invoice όπου περιέχει όλο τον λογαριασμό ή τα προϊόντα τα οποία έχουν απομείνει σε περίπτωση που έχει προηγηθεί split bill.
Έπειτα δίνεται η δυνατότητα να κλείσει το τραπέζι ή να παραμείνει ανοιχτό αν και πληρωμένο.

Μεταβαίνει πάλι στην αρχική οθόνη όπου το τραπέζι εμφανίζεται στα κλειστά.
Αναγράφεται ο χρήστης που το έκλεισε καθώς και τον χρόνο που έμεινε ανοιχτό.

Επίλογος

Συμπερασματικά, πρέπει να τονιστεί ότι τα πληροφοριακά συστήματα και ειδικότερα το σύστημα ασύρματης παραγγελιοληψίας όσον αφορά τους χώρους μαζικής εστίασης και τις ανάγκες που καλύπτει σε μια σύγχρονη επιχείρηση τέτοιου είδους είναι ένα πολύ καλό και χρήσιμο εργαλείο που βοηθά στην οργάνωση και την καλύτερη διαχείριση.

Στόχος για τη δημιουργία του συγκεκριμένου POS σε χώρους εστίασης ήταν ολόκληρη η εφαρμογή να είναι σωστά δομημένη, εύχρηστη, λειτουργική, γρήγορη και εύκολα επεκτάσιμη. Η χρησιμοποίηση της αρχιτεκτονικής τριών επιπέδων υπήρξε σημείο κλειδί, διότι έδωσε την ελευθερία χρησιμοποίησης διαφορετικών τεχνολογιών σε κάθε επίπεδο, με αποτέλεσμα την επιλογή των βέλτιστων εργαλείων για την υλοποίηση του κάθε επιπέδου. Η React Native σε επίπεδο παρουσίασης έκανε την εφαρμογή εξαιρετικά γρήγορη, ενώ σε επίπεδο κώδικα βοήθησε πάρα πολύ στη σωστή δόμηση και επαναχρησιμοποίηση πολλών κομματιών κώδικα. Η υλοποίηση της επιχειρησιακής λογικής στη μορφή ενός REST API και της εφαρμογής των κανόνων της συγκεκριμένης αρχιτεκτονικής είχε ως αποτέλεσμα οι λειτουργίες να εκτελούνται γρήγορα και ασύγχρονα, δίνοντας στον χρήστη τη βέλτιστη εμπειρία χρήσης του POS. Τέλος, η ανάπτυξη της εφαρμογής με χρήση της αρχιτεκτονικής τριών επιπέδων, έδωσε τη δυνατότητα στην εφαρμογή να είναι εξαιρετικά επεκτάσιμη και σε δυνατότητες, αλλά και σε ευκολία επέκτασης.

Οι δυσκολίες που υπερνικήθηκαν και οι ώρες που δαπανήθηκαν, κατέληξαν σε ένα λειτουργικό μεν τελικό αποτέλεσμα, το οποίο όμως σίγουρα επιδέχεται βελτιώσεις και χρήζει περαιτέρω ανάπτυξης.

Τέλος, αξίζει να σημειωθεί ότι πρώτο ρόλο πάντα παίζει ο άνθρωπος που χειρίζεται όλα αυτά τα συστήματα και η διάθεσή του να χρησιμοποιήσει όλα αυτά τα μέσα προς το κοινό καλό ολόκληρης της επιχείρησης μαζικής εστίασης.

Βιβλιογραφία

[1]: https://en.wikipedia.org/wiki/Point_of_sale

[2]: www.cessoftware.com/blog/fascinating-history-pos-systems/

[3]Kalliopi, Kanaki (2015-04-27). ΥΠΟΛΟΓΙΣΤΙΚΟ ΝΕΦΟΣ.

[4] Καλλιόπη Α. Κανάκη (10 Μάιος 2016). «ΥΠΟΛΟΓΙΣΤΙΚΟ ΝΕΦΟΣ».

[5] Qian et al. pp 626-631

[6] «Microsoft Says to Spend 90% of R&D on Cloud Strategy». Αρχειοθετήθηκε από το πρωτότυπο στις 18 Οκτωβρίου 2013.

[7] https://www.mobiletransaction.org/cloud-based-pos-system/

[8]https://www.forbes.com/sites/forbestechcouncil/2018/05/25/what-is-cloud-point-of-sale-and-how-is-it-different-from-traditional-pos/#400eb3261180

[9]https://hospitalitytech.com/2018-pos-software-trends-report

[10] https://www.tandfonline.com/doi/abs/10.1300/J369v07n02_07

[11] The New Stack, “JavaScript’s History and How it Led To ReactJS”, https: /the newstack.io/javascripts-history-and-how-it-led-to-reactjs

[12] Learning React Native by Bonnie Eisenman (2018)

[13] PHP - http://www.php.net

[14] Wikipedia (2020). PHP. Retrieved, from http://el.wikipedia.org/wiki/PHP

[15] Laravel Framework Documentation - https://laravel.com/docs/

[16] Matt Stauffer (2019) Laravel: Up & Running: A Framework for Building Modern PHP Apps

[17] websocket Vanessa Wang, Frank Salim, and Peter Moskovits - The Definitive Guide to HTML5 WebSocket – Apress 2013 - http://www.websocket.org/book.html

[18] Erik Wilde, Casare Pautasso, REST: From Research to Practice, Springer, 2011.

[19] Luke Welling and Laura Thomson (2001), PHP and MySQL Web Development, Sams Publishing

[20] https://en.wikipedia.org/wiki/Apache_HTTP_Server