ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Τμήμα Μηχανικών Πληροφορικής ΑΤΕΙΘ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη ολοκληρωμένου συστήματος Point of Sale για μαγαζιά μαζικής εστίασης

Των φοιτητών

Δέδε Αναστάσιου 07 / 3238

Καραπιδάκη Παύλου 07 / 3265

Επιβλέπων Καθηγητής

Κεραμόπουλος Ευκλείδης

Θεσσαλονίκη 2014

ΠΡΟΛΟΓΟΣ

Στις μέρες μας, τα θέματα οργάνωσης και ελέγχου μιας επιχείρησης αποτελούν για τους περισσότερους επιχειρηματίες βασική προτεραιότητα, με αποτέλεσμα να γίνεται κατανοητός ο λόγος για τον οποίο κάποιος θα ξοδέψει χρήματα για έναν τέτοιο σκοπό. Οι Point of Sale εφαρμογές που κυκλοφορούν στην αγορά σκοπό έχουν να διευκολύνουν τη ζωή όλων σε μια επιχείρηση εστίασης, είτε αυτοί είναι διαχειριστές είτε απλοί υπάλληλοι.

Η υλοποίηση από τη μεριά του προγραμματιστή παρ' όλα αυτά είναι ακόμα πιο απαιτητική, όσο περισσότερη άνεση προσφέρει στον εκάστοτε πελάτη. Σε αρκετές περιπτώσεις, τα ολοκληρωμένα Point of Sale συστήματα απαιτούν την προσήλωση μεγάλου αριθμού προγραμματιστών με γνώσεις σε πολλούς διαφορετικούς τομείς προγραμματισμού και τεχνολογίες. Το κόστος για αυτό το λόγο μπορεί πολλές φορές να ξεπεράσει τα όρια ενός μέσου επιχειρηματία όταν στο παιχνίδι μπαίνει και η έννοια της συντήρησης, της αποσφαλμάτωσης, και της παραμετροποίησης του προγράμματος Client.

Η δική μας προσέγγιση στο θέμα του POS ορίζεται στα πλαίσια μιας πτυχιακής εργασίας, με σκοπό την υλοποίηση ενός συστήματος POS με τον καλύτερο δυνατό αποτέλεσμα. Οι γνώσεις οι οποίες θα καλύψουν την υλοποίηση αυτού του προγράμματος, δεν προϋπήρχαν εξ ολοκλήρου κατά την έναρξη συγγραφής γραμμών κώδικα σε μια λευκή οθόνη, αλλά αποκτήθηκαν κατά τη διάρκεια του έργου, πράγμα που αποτελεί για μας το βαθύτερο σκοπό μιας πτυχιακής εργασίας.

Τέλος αξίζει να σημειωθεί ότι το παρών έγγραφο καθώς και όλος ο πηγαίος κώδικας που χρησιμοποιήθηκε για να έρθει εις πέρας αυτή η εφαρμογή αποτελεί προϊόν προσωπικής έρευνας και μελέτης, αφού μέσα από την εργασία αυτή αυξήσαμε τις γνώσεις μας πάνω στη δημιουργία εφαρμογών για Android συσκευές, βάσεις δεδομένων και PHP, συνθέτοντας διαφορετικά UI για τις ανάγκες κάθε τύπου χρήστη του Point of Sale.

ΠΕΡΙΛΗΨΗ

Στόχος αυτής της πτυχιακής εργασίας είναι η δημιουργία εφαρμογής Client σε Android, Client για την παραλαβή των παραγγελιών σε Tablet ή PC, η δημιουργία βάσης δεδομένων και το Administrator UI σε PC, συνθέτοντας ένα ολοκληρωμένο σύστημα Point Of Sale. Θα γίνει αναλυτική περιγραφή όλων των μεθόδων και των τεχνολογιών που θα χρησιμοποιηθούν με εικόνες όπου κριθεί αναγκαίο.

Θα χρησιμοποιηθεί στις περισσότερες περιπτώσεις ο όρος POS (Point Of Sale) για συντομία. Σε κάθε κεφάλαιο θα αναλύεται ένα μεγάλο κομμάτι του POS για παράδειγμα η εφαρμογή Client που θα τρέχει σε Android συσκευή για τους σερβιτόρους και το Chef UI που θα τρέχει σε PC ή Tablet, είναι δύο μεγάλα και διαφορετικά κεφάλαια.

Ένα μεγάλο κομμάτι της συγκεκριμένης πτυχιακής εργασίας είναι το Administrator UI. Αυτό συμβαίνει διότι κάθε σοβαρό POS πρέπει να έχει ένα κομμάτι και από αυτή τη πλευρά, το οποίο θα μπορεί να κάνει ευέλικτη την εφαρμογή. Οι Clients που χρησιμοποιούν τα data που θα δημιουργεί ο Administrator, είναι οι χρήστες του POS στην Android εφαρμογή, και οι σεφ από PC ή Tablet.

Βασικές γλώσσες προγραμματισμού και περιγραφής που θα αναλυθούν για να έρθει εις πέρας αυτό το έργο είναι στο μεγαλύτερο μέρος PHP, SQL, HTML, JAVASCRIPT, JQUERY, XML και JAVA. Τα βασικότερα εργαλεία με τα οποία θα υλοποιηθούν όλα αυτά να είναι τα Eclipse, Notepad++, phpMyAdmin, Apache Server και διάφορες συσκευές με λειτουργικό Android.

ABSTRACT

The aim of this thesis is to create Client - Server application on Android, a database using phpMyAdmin, Server side services to perform the necessary functions for the clients, UI for receiving orders sent by waiters on Tablet or PC, and an Administrator UI, composing an integrated Point Of Sale system. We will make a detailed description of all methods and technologies that are used even with pictures where necessary.

POS (Point Of Sale) term will be used in most cases for short. Each chapter analyzes a large piece of POS, as the application for waiters that will be running on Android devices and the Administrator UI that will be running on PC, are two large and different chapters.

A large part of this thesis is the Administrator UI. This is because any serious POS must have a piece from this side, which can make the application flexible. The Clients who use the data that are created by the Administrator are users of the Android POS application and chefs that use a PC or a Tablet.

Programming languages and description that will be analyzed in order to come out this system is mostly PHP, SQL, HTML, JAVASCRIPT, JQUERY, XML and JAVA. The basic tools with which to implement all of this are Eclipse with ADT, Notepad++, phpMyAdmin, Apache Server and various devices running Android OS.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ

1. ΚΕΦΑΛΑΙΟ ΠΡΩΤΟ

ANDROID, ΜΙΑ ΓΕΝΙΚΗ ΕΙΚΟΝΑ

ΕΙΣΑΓΩΓΗ

Στο κεφάλαιο αυτό θα εξηγήσουμε τους λόγους τους οποίους επιλέξαμε την πλατφόρμα android για την υλοποίηση του προγράμματος μας, βάση και της ιστορικής της αναδρομής μέχρι σήμερα, καθώς και τα εργαλεία που θα χρησιμοποιήσουμε για τον σκοπό αυτό. Τέλος θα δώσουμε έναν ορισμό για το τι είναι POS (Point Of Sale) και ποια η χρησιμότητα του στο χώρο εργασίας.

1.1. ΙΣΤΟΡΙΚΗ ΑΝΑΔΡΟΜΗ

Το Android είναι ένα λειτουργικό σύστημα ανοιχτού κώδικα, βασισμένο στο Linux, για φορητές συσκευές όπως smartphones και tablets. Η Google, έχοντας εντοπίσει αυξημένη χρήση του internet και αναζητήσεων στον παγκόσμιο ιστό μέσω κινητών συσκευών (mobile devices) εξαγοράζει το 2005 την Android Inc με σκοπό την ανάπτυξη μιας πλατφόρμας για τέτοιου είδους συσκευές. Περίπου την ίδια περίοδο, η Apple παρουσιάζει το iPhone (2007) το οποίο κάνει χρήση κάποιων επαναστατικών καινοτομιών όπως η υποστήριξη multitouch και η ανοιχτή παγκόσμια αγορά εφαρμογών. Το Android γρήγορα προσαρμόστηκε ώστε να υιοθετήσει και να παρέχει επίσης τις προαναφερθείσες δυνατότητες αν και κατά κοινή ομολογία οι πρώτες εκδόσεις του υπολείπονταν του iPhone όσον αφορά τις υποστηριζόμενες λειτουργίες και τη συνολική εμπειρία που πρόσφερε στους χρήστες.

Το 2007 λοιπόν, τη χρονιά λανσαρίσματος του iPhone, δημιουργείται ένας οργανισμός που αποτελείται από μεγάλο αριθμό εταιρειών τηλεπικοινωνιακού εξοπλισμού καθώς και εταιρείες πληροφορικής όπως η Google, η T-Mobile, η Motorola, η Samsung, η Sony Ericsson, η Intel, η Vodafone, η Toshiba κ.α. με όνομα Open Handset Alliance (http://www.openhandsetalliance.com) και σκοπό την έρευνα και την ανάπτυξη τεχνολογιών για την παραγωγή συσκευών που θα διευκολύνουν τόσο τους παρόχους κινητής τηλεφωνίας όσο και τους κατασκευαστές κινητών τηλεφώνων αλλά και τους προγραμματιστές εφαρμογών. Τα μέλη της συμμαχίας δεσμεύτηκαν να παρέχουν τις τεχνολογίες

αυτές βάσει του μοντέλου ανοιχτού πηγαίου κώδικα Apache.

Η πρώτη 'early look' έκδοση του Android SDK δημοσιεύτηκε το Νοέμβριο του 2007, ενώ το πρώτο smartphone που έκανε χρήση λειτουργικού Android ήταν το G1 της T-Mobile. Οι συσκευές Android άρχισαν να διαδίδονται με γρήγορο ρυθμό κυρίως λόγω της δυνατότητας της πλατφόρμας να εκμεταλλεύεται το μοντέλο cloud computing αλλά και της έμφυτης υποστήριξης για συνεργασία με μία σχεσιακή βάση δεδομένων (SQLite).

Ακολούθησαν αρκετές αναβαθμισμένες εκδόσεις του Android, κάθε μία προσθέτοντας νέα χαρακτηριστικά και λειτουργίες. Για κάποιον άγνωστο λόγο κάθε έκδοση του Android φέρει και μία κωδική ονομασία ενός γλυκού εδέσματος (1.5 - Cupcake, 1.6 - Donut, 2.0 Eclair κ.α.), ενώ από τις εκδόσεις αυτές εκείνη που εισήγαγε την υποστήριξη πιο ανεπτυγμένων λειτουργιών ήταν σίγουρα η 2.0 στην οποία ενσωματώθηκε η υποστήριξη multitouch, HTML 5, text-to-speech και η δυνατότητα πιο προχωρημένων αναζητήσεων.

Κατά τη σύνταξη των σημειώσεων που έχετε στα χέρια σας, η πιο πρόσφατη έκδοση του λειτουργικού Android είναι η 4.4 (KitKat).

1.2. ΕΡΓΑΛΕΙΑ ΑΝΑΠΤΥΞΗΣ ANDROID ΕΦΑΡΜΟΓΩΝ

1.2.1. ANDROID SDK

Το Android SDK (Software Development Kit) αποτελεί μια συλλογή εργαλείων και βιβλιοθηκών που καθιστούν εφικτή την ανάπτυξη εφαρμογών στο Android. Τη στιγμή που γράφονται αυτές οι γραμμές, το SDK έχει φτάσει στην έκδοση r19 η οποία υποστηρίζει το Android 4.0.3. Το λογισμικό ανάπτυξης λοιπόν περιλαμβάνει μια μεγάλη λίστα με εργαλεία ανάπτυξης. Σε αυτά περιλαμβάνονται:

- Εργαλεία Debugging των εφαρμογών
- Βιβλιοθήκες
- Εξομοιωτής συσκευών (Android Virtual Machines)
- Documentation
- Δείγματα Κώδικα
- Tutorials

Το SDK υποστηρίζει πολλά δημοφιλή λειτουργικά συστήματα

συμπεριλαμβανομένων όλων των σύγχρονων διανομών Linux, το MAC OS X 10.4.9 και μεταγενέστερα, και τα Windows XP και τις μεταγενέστερες εκδόσεις.

Το λογισμικό ανάπτυξης αποτελείται από πακέτα τα οποία βρίσκονται αποθηκευμένα σε ένα επίσημο repo της Google, και ο προγραμματιστής μπορεί να κατεβάσει πέραν των βασικών πακέτων, και άλλα τα οποία υποστηρίζουν παλαιότερες εκδόσεις του Android, ή άλλες συσκευές εκτός κινητών συσκευών (πχ Google TV Addon).

Όσον αφορά την υποστήριξη παλαιότερων εκδόσεων του Android, το SDK κάνει εφικτή την υποστήριξη σε αυτές δίνοντας στον προγραμματιστή την δυνατότητα να στοχεύσει αυτός σε πια APIs θα απευθύνεται η εφαρμογή του. Αυτό είναι αναγκαίο λόγω του ότι πολλοί χρήστες έχουν παλαιότερες λειτουργικές συσκευές οι οποίες κυκλοφορήσαν με παλαιότερες εκδόσεις του Android (πχ 1.6 ή 2.1), και ο κατασκευαστής της συσκευής δεν έχει ή δεν πρόκειται να βγάλει αναβάθμιση για την συσκευή τους. Το πρόβλημα αυτό είναι γνωστό σαν διάσπαση του Android (Android Fragmentation).

1.2.2. ECLIPSE IDE ME ADT (ANDROID DEVELOPMENT TOOLS)

Ο προγραμματισμός στο Android βασίζεται στην γλώσσα Java και ο κάθε προγραμματιστής μπορεί να χρησιμοποιήσει έναν οποιονδήποτε text editor για να γράψει κώδικα για να επεξεργαστεί τα αρχεία *.Java και *.XML και μετέπειτα να τα κάνει compile μέσω γραμμής εντολών χρησιμοποιώντας το JDK (Java Development Kit). Ο συγκεκριμένος τρόπος ανάπτυξης δεν είναι ιδιαίτερα φιλικός στον χρήστη γι’ αυτό συνίσταται η χρήση ενός IDE (Integrated Development Environment) που να υποστηρίζει Java, όπως το Eclipse ή το Netbeans.

Η Google υποστηρίζει επίσημα το Eclipse και έχει αναπτύξει ειδικά για αυτό το ADT plugin, το οποίο παρέχει σύνδεση με το Android SDK με όλες τις δυνατότητες που περιλαμβάνει αυτό. Επίσης το plugin παρέχει σύνδεση με τον AVD Manager, για διαχείριση και εκκίνηση από το GUI του, εικονικών συσκευών Android για δοκιμές και debugging των εφαρμογών.

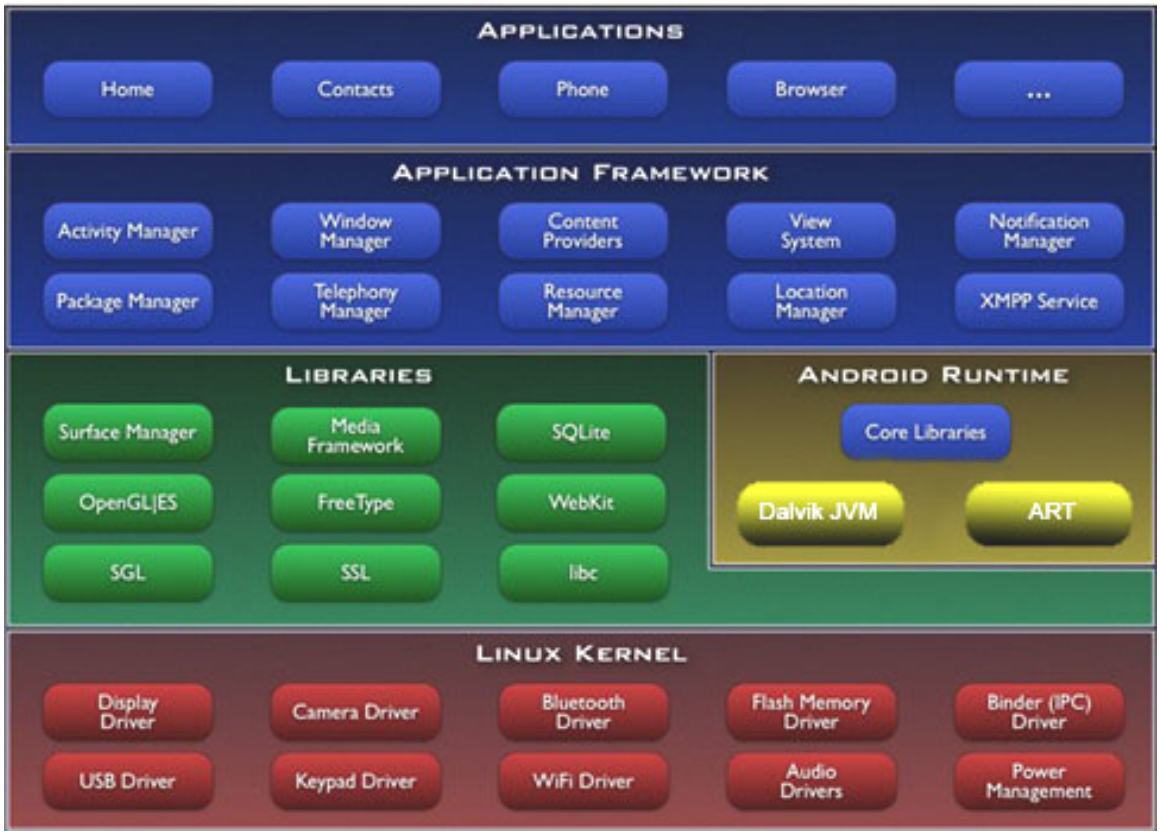
Φυσικά όπως είπαμε και παραπάνω, ο κάθε προγραμματιστής μπορεί να χρησιμοποιήσει τον Text Editor ή IDE της επιλογής του για τη δημιουργία του κώδικα και μετέπειτα να χρησιμοποιήσει τα εργαλεία JDK και Apache Ant

μέσω γραμμής εντολών για να κάνει compile την εφαρμογή του ώστε να την τεστάρει με όλες τις δυνατότητες που το παρέχει το Android SDK.

Η επιλογή ενός IDE που κάνει όλη την πολύπλοκη δουλειά για μας είναι προφανής λοιπόν. Επίσης τα περισσότερα παραδείγματα και άρθρα για το Android στηρίζονται στο γεγονός ότι η πλειονότητα των developers χρησιμοποιεί το Eclipse μαζί με το ADT plugin οπότε ξεκινάνε με αυτό σαν δεδομένο.

1.3. Η ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΟΥ ANDROID

Όπως μπορείτε να δείτε στην εικόνα 1, στον πυρήνα της πλατφόρμας Android βρίσκεται ένα Linux kernel το οποίο είναι υπεύθυνο για τη διαχείριση των device drivers, τον έλεγχο πρόσβασης στους πόρους του συστήματος, τη διαχείριση μνήμης και τις λοιπές υπηρεσίες που παρέχει ένα λειτουργικό σύστημα.



Εικόνα 1: Android Architecture

Στους device drivers συγκαταλέγονται αυτοί της οθόνης, του Wi-Fi, της κάμερας, του ήχου κ.α. Ένα επίπεδο επάνω βρίσκονται οι native βιβλιοθήκες του συστήματος που είναι γραμμένες σε C++ και περιλαμβάνουν το OpenGL, την SQLite, την Media library κ.α. Οι εφαρμογές που τρέχουν στο κινητό μπορούν να έχουν πρόσβαση στις βιβλιοθήκες αυτές μέσω της Dalvik JVM ή της ART. Όπως έχει ήδη αναφερθεί, οι εφαρμογές Android είναι γραμμένες σε Java και άρα για να τρέξουν χρειάζονται το αντίστοιχο περιβάλλον. Όπως λοιπόν για να εκτελέσουμε μία εφαρμογή σε ένα PC είναι απαραίτητο να είναι εγκατεστημένο το κατάλληλο JRE (Java Runtime Environment), για τις εφαρμογές Android τον ρόλο του JRE παίζουν οι Dalvik και ART.

1.4. ΓΙΑΤΙ ANDROID ΑΝΤΙ IPHONE

Σήμερα, η μάχη για την επικράτηση στην αγορά των smartphones μαίνεται μεταξύ δύο κολοσσών, του iPhone και του Android. Όπως αναφέρθηκε ήδη, το iPhone δείχνει να έχει ακόμη το προβάδισμα και οι κάτοχοί του διατείνονται πως προσφέρει σαφώς ανώτερη εμπειρία στο χρήστη και μεγαλύτερη ποικιλία εφαρμογών. Ενδεχομένως κάτι τέτοιο να ισχύει, μιας και το iPhone ήταν αυτό που κυκλοφόρησε πρώτο εισάγοντας παράλληλα καινοτόμες τεχνολογίες στην αγορά. Εδώ δεν θα μας απασχολήσουν συγκρίσεις για το ποιο από τα δύο είναι καλύτερο ή πιο δημοφιλές. Μια τέτοια συζήτηση θα ήταν ανούσια, άλλωστε εμπεριέχει και το προσωπικό γούστο του καθενός. Αντίθετα, στο πλαίσιο που κινούμαστε είναι σαφώς πιο ουσιώδες το να εστιάσουμε στο ποια είναι τα θετικά και αρνητικά της κάθε πλατφόρμας, με γνώμονα την ανάπτυξη εφαρμογών. Ένα από τα πιο δυνατά σημεία του Android είναι πως η ανάπτυξη εφαρμογών γίνεται σχεδόν αποκλειστικά στη Java, ίσως την πιο δημοφιλή, πιο ολοκληρωμένη και καλύτερα δομημένη γλώσσα που υπάρχει σήμερα. Επιπλέον, η υλοποίηση Android εφαρμογών μπορεί να γίνει με μικρό κόστος από την πλευρά του προγραμματιστή. Ο υποψήφιος Android developer θα πρέπει να διαθέτει έναν προσωπικό υπολογιστή με οποιοδήποτε λειτουργικό σύστημα, ένα κινητό Android για το σωστό τεστάρισμα των εφαρμογών του και τέλος έναν λογαριασμό developer στο Android Market, για τον οποίο απαιτείται η εφ' άπαξ καταβολή ενός τιμήματος της τάξης των 25$ (ισχύον τίμημα κατά τη σύνταξη των σημειώσεων).

Από την πλευρά του iPhone, η ανάπτυξη γίνεται σε Objective C, μια γλώσσα που αποτελεί υπερσύνολο της C. Κινδυνεύοντας να κατηγορηθώ για εύνοια υπέρ της Java, νομίζω πως δε θα διαφωνήσει κανείς πως όσον αφορά τη σύνταξη κώδικα σε C και Java, δεν υπάρχει κανένα περιθώριο σύγκρισης! Αντίστοιχα, το κόστος για να είναι σε θέση κάποιος να υλοποιήσει εφαρμογές iPhone και να τις διαθέσει στην αγορά είναι σαφώς μεγαλύτερο. Απαιτείται ένας υπολογιστής Mac, ένα iPhone για το τεστάρισμα των εφαρμογών του και ένας λογαριασμός developer, για τον οποίο θα πρέπει να καταβάλλει το ποσό των 99$ κάθε χρόνο (ισχύουσα τιμή κατά τη σύνταξη των σημειώσεων). Συγκρίνοντας τώρα τις τιμές των συσκευών, μία νέα συσκευή iPhone κοστίζει κατά μέσο όρο 450€ (ανάλογα με την έκδοση) ενώ είναι δυνατόν να αποκτήσετε μια αξιοπρεπέστατη συσκευή Android με λιγότερα από 200€. Συμψηφίζοντας όλα τα παραπάνω, είναι σαφές πως όσον αφορά τα κόστη αλλά και τις τεχνολογίες που εμπλέκονται για την υλοποίηση εφαρμογών, το Android έχει σαφές προβάδισμα. Από την άλλη πλευρά, υπάρχει η αίσθηση πως οι χρήστες iPhone είναι περισσότερο διατεθειμένοι να πληρώσουν για να αγοράσουν μια εφαρμογή από ότι οι χρήστες Android. Αυτό οφείλεται στο γεγονός πως το iPhone καλώς ή κακώς θεωρείται στις μέρες μας ως status symbol, ένα gadget δηλαδή που πολλοί από τους κατόχους του το έχουν με σκοπό τον εντυπωσιασμό. Τέτοιοι χρήστες δεν είναι επιφυλακτικοί στο να πληρώσουν και να κατεβάσουν μια ανούσια εφαρμογή απλά για να τη δείξουν στους φίλους τους και αυτή η συμπεριφορά λειτουργεί προς όφελος των developers. Αντίθετα, το Android βασίζεται όπως θα δούμε και στη συνέχεια στο Linux και έχει έναν αμιγώς open source χαρακτήρα. Πολλοί από τους χρήστες του Android ασπάζονται αυτή την open source φιλοσοφία και δεν είναι διατεθειμένοι να πληρώσουν για μία εφαρμογή, ή για να το θέσουμε σε πιο σωστή βάση, θα αγόραζαν μια εφαρμογή μόνο αν είναι εγγυημένα χρήσιμη και προσεγμένη.

Ένα ακόμη πλεονέκτημα για τους developers iPhone εφαρμογών αποτελεί το γεγονός πως το iPhone είναι ένα. Διαθέτει μία οθόνη συγκεκριμένων διαστάσεων, ένα δεδομένο interface διάδρασης με τον χρήστη και συγκεκριμένο σετ από αισθητήρες, άρα οι developers της συγκεκριμένης πλατφόρμας δεν θα χρειαστεί ποτέ να "πονοκεφαλιάσουν" κατά την ανάπτυξη μιας εφαρμογής και να γράψουν τον κατάλληλο κώδικα για να μπορεί αυτή να

τρέχει χωρίς προβλήματα σε συσκευές με διαφορετικά χαρακτηριστικά. Αντίθετα, όσον αφορά το Android υπάρχει πληθώρα συσκευών με διαφορετικά χαρακτηριστικά όπως οθόνες διαφορετικών διαστάσεων και αναλύσεων, διαφορετικά σετ αισθητήρων, διαφορετικά interfaces διάδρασης κ.α. τα οποία οι developers θα πρέπει να λαμβάνουν υπ' όψιν κατά την ανάπτυξη των εφαρμογών τους ώστε αυτές να τρέχουν απροβλημάτιστα και με ομοιόμορφο τρόπο σε όλες τις συσκευές. Αυτό προϋποθέτει έξτρα δουλειά και πρόνοια από την πλευρά του προγραμματιστή.

1.5. ΟΡΙΣΜΟΣ ΚΑΙ ΧΡΗΣΙΜΟΤΗΤΑ ΤΟΥ Point Of Sale (POS)

Με τον όρο Point of Sale ή σημείο πώλησης προσδιορίζεται το σημείο όπου σε ένα κατάστημα λιανικής ολοκληρώνεται μια συναλλαγή. Είναι το σημείο στο οποίο ένας πελάτης κάνει μια πληρωμή στον έμπορο, σε αντάλλαγμα για αγαθά ή υπηρεσίες. Στο σημείο πώλησης, ο πωλητής θα υπολογίσει το ποσό που οφείλεται από τον πελάτη και παρέχοντας του επιλογές για την πληρωμή. Ο έμπορος θα πρέπει κανονικά εκδώσει απόδειξη για τη συναλλαγή.

Το σύγχρονο σημείο πώλησης συχνά αναφέρεται ως το σημείο της υπηρεσίας, διότι δεν είναι μόνο ένα σημείο πώλησης, αλλά και ένα σημείο της επιστροφής ή της παραγγελίας του πελάτη. Επιπλέον περιλαμβάνει προηγμένα χαρακτηριστικά για να ανταποκριθεί σε διαφορετικές ανάγκες, όπως τη διαχείριση αποθεμάτων, οικονομικά, αποθήκευση, κλπ, όλα χτισμένα στο λογισμικό POS. Πριν από την σύγχρονη τεχνολογία POS, όλες αυτές οι λειτουργίες πραγματοποιούνταν ανεξάρτητα και απαιτούσαν την χειροκίνητη εκ νέου πληκτρολόγηση των πληροφοριών, η οποία μπορούσε να οδηγήσει σε σφάλματα εισόδου. Πόσο μάλλον όταν τα έγγραφα ήταν χειρόγραφα και απαιτούσαν ένα έξτρα άτομο σε μια μικρή επιχείρηση για να βάλει σε τάξη τη «γραφειοκρατία».

Στη συγκεκριμένη πτυχιακή εργασία θα παρουσιάσουμε την υλοποίηση σε λειτουργικό Android ενός Point of Sale με σκοπό τη διευκόλυνση στις παραγγελιοληψίες σε μαγαζιά μαζικής εστίασης. Θα αναλυθούν μέθοδοι που χρησιμοποιήθηκαν για την ολοκλήρωση της εφαρμογής και θα εξηγηθούν λεπτομερώς οι λόγοι χρήσης της πλειονότητας των εντολών.

Θα χρησιμοποιηθούν για αυτό το σκοπό όλες οι τεχνολογίες που προαναφέρθηκαν όπως το Android SDK, το ADT plug-in για το Eclipse, δυο συσκευές Android (κινητά τηλέφωνα) NEXUS 4 και 5, για το debugging του προγράμματος που θα αναπτυχθεί, Tablet για τις ανάγκες του Chef UI της εφαρμογής, ένα PC στο όποιο θα τρέχει ένας Apache Server για την επικοινωνία όλων των clients με τη βάση δεδομένων και ένα ασύρματο router για τη δικτύωση και επικοινωνία των συσκευών.

ΓΝΩΣΤΑ POS 2014

Τα 10 ποιοτικότερα κατά σειρά Point of Sale προγράμματα για το 2014 είναι τα εξής [15]:

1. **AmberPOS** (http://www.softwareadvice.com/retail/pacific-amber-technologies-amberpos-profile/)
2. **LightSpeed (**http://www.softwareadvice.com/retail/lightspeed-profile/**)**
3. **NCR Counterpoint POS and Retail Management (**http://www.softwareadvice.com/retail/radiant-systems-counterpoint-profile/**)**
4. **iVend Retail (**http://www.softwareadvice.com/retail/citixsys-ivend-retail-profile/**)**
5. **GiftLogic (**http://www.softwareadvice.com/retail/giftlogic-profile/**)**
6. **TouchBistro (**http://www.softwareadvice.com/retail/touchbistro-profile/**)**
7. **COMCASH Retail ERP (**http://www.softwareadvice.com/retail/comcash-profile/**)**
8. **AIMsi (**http://www.softwareadvice.com/retail/aimsi-profile/**)**
9. **Gotmerchant.com (**http://www.softwareadvice.com/retail/gotmerchant-profile/**)**
10. **cloudseeder POS (**http://www.softwareadvice.com/retail/cloudseeder-pos-profile/**)**

ΕΠΙΛΟΓΟΣ

Γίνεται κατανοητό ότι επιλέξαμε το λειτουργικό σύστημα Android για τις ανάγκες της πτυχιακής εργασίας μας διότι είναι κάτι γνώριμο στο ευρύ κοινό με αποτέλεσμα να έχουμε τη δυνατότητα να αντλήσουμε πληροφορίες από παντού στο διαδίκτυο για όλες μας τις ανάγκες. Επίσης τα εργαλεία που χρησιμοποιούνται για το συγκεκριμένο σκοπό είναι εύκολα στο χειρισμό και διανέμονται δωρεάν. Τέλος ένα POS μπορεί να έχει πολλές εφαρμογές σε πολλές διαφορετικές επιχειρήσεις και να καλύψει ανάγκες όχι μόνο λειτουργικές αλλά και οργανωτικές.

2. ΚΕΦΑΛΑΙΟ ΔΕΥΤΕΡΟ

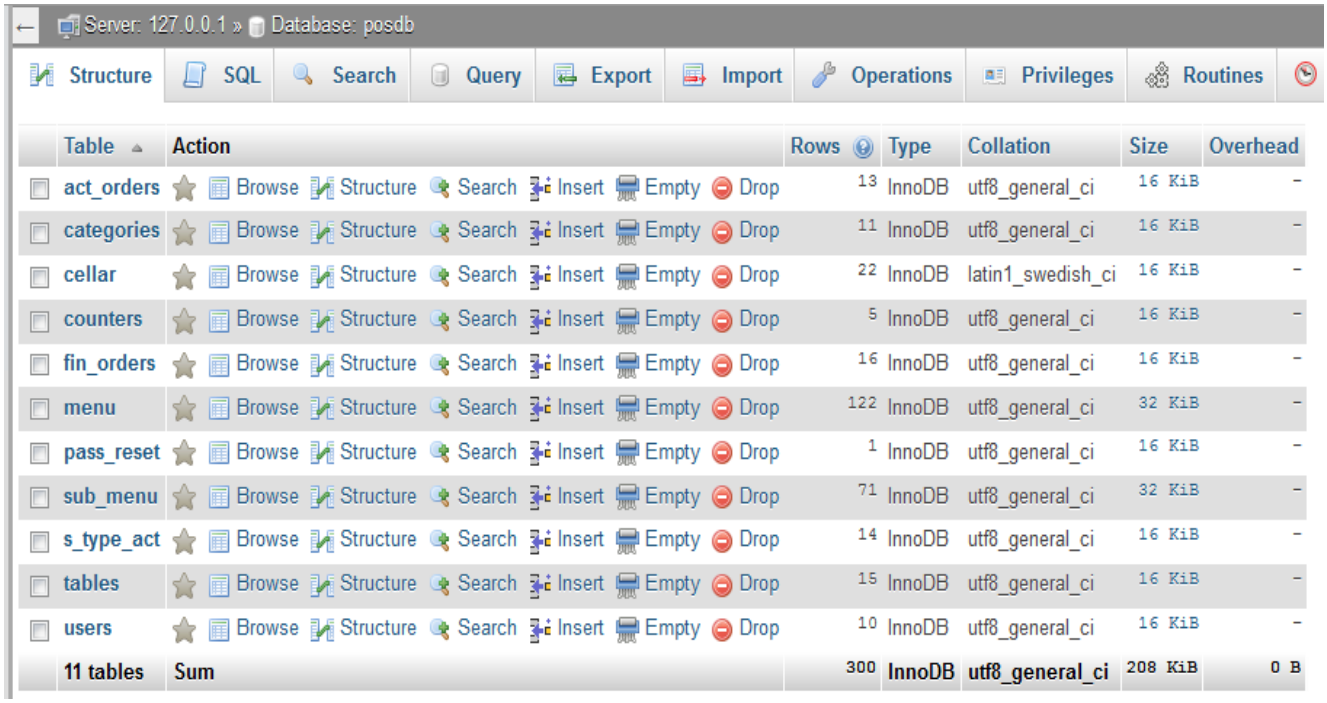
Η ΔΙΚΗ ΜΑΣ ΒΑΣΗ

ΕΙΣΑΓΩΓΗ

Χρησιμοποιώντας το phpMyAdmin το οποίο έρχεται πακέτο με το XAMPP, δημιουργήσαμε τη δική μας βάση δεδομένων προσαρμοσμένη στις ανάγκες μας. Την ονομάσαμε posdb.sql και παρακάτω θα αναλύσουμε έναν έναν όλους τους πίνακες και τα πεδία που την αποτελούν καθώς και τη χρησιμότητα της. Να τονίσουμε στο σημείο αυτό πως κάθε εφαρμογή είναι απαραίτητο να υποστηρίζεται από μια καλά οργανωμένη βάση δεδομένων και για το λόγο αυτό η δική μας βάση, δεν δημιουργήθηκε με μιας, καθώς οι ανάγκες υλοποίησης του προγράμματος, μας έδειξαν σταδιακά και τις ανάγκες ύπαρξης ή μη των περισσότερων πινάκων και σχέσεων στη βάση.

2.1. phpMyAdmin

Παρακάτω βλέπουμε την εικόνα της βάσης μας μέσα από το πρόγραμμα επεξεργασίας το οποίο βρίσκεται κάτω από το σύνδεσμο localhost/phpmyadmin:

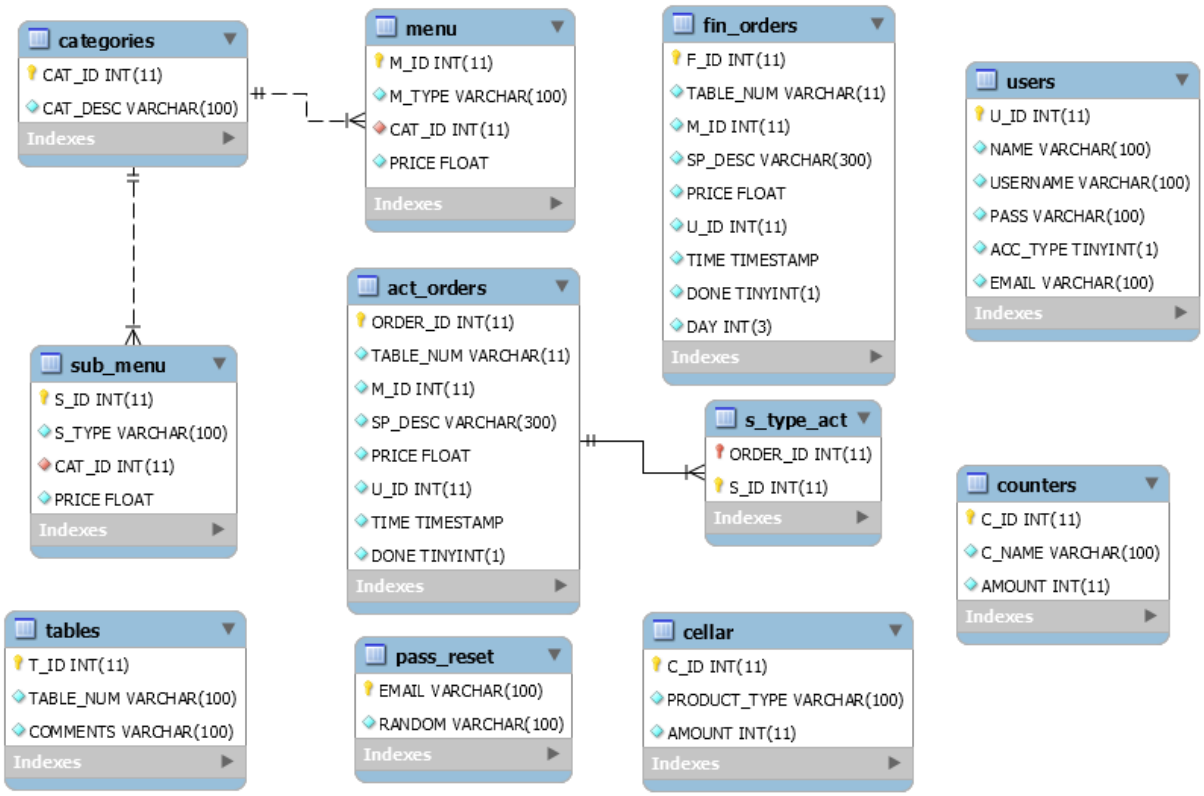


Εικόνα 2: phpMyAdmin Interface

Όλες οι βασικές λειτουργίες οι οποίες χρειαζόμαστε για να διαχειριστούμε τη βάση μας βρίσκονται εδώ. Μπορούμε είτε με το ποντίκι είτε γράφοντας κώδικα SQL να προσθέσουμε ή να αφαιρέσουμε πίνακες και πεδία σε αυτούς, να κάνουμε ερωτήματα και να δημιουργήσουμε σχέσεις μεταξύ των πινάκων μας, πχ ξένα κλειδιά ή διπλό primary key για κάποιους πίνακες, και να γεμίσουμε τους πίνακες μας με δεδομένα.

2.2. ER DIAGRAM

Το διάγραμμα της βάσης που ακολουθεί δημιουργήθηκε με το MySQL Workbench με τη μέθοδο του Reverse Engineering [27]. Φαίνονται στο διάγραμμα οι σχέσεις μεταξύ των πινάκων, καθώς και τα πεδία που περιέχει ο καθένας με τους αντίστοιχους τύπους τους σε παρένθεση:



Εικόνα 3: ER Diagram

Για την ορθή λειτουργία του προγράμματος το οποίο υλοποιούμε χρειαζόμαστε τους 11 παραπάνω πίνακες. Ας δούμε τι περιέχει ο καθένας και ποια η χρησιμότητά του:

- **Tables**: Περιέχει το πεδίο t_id το οποίο είναι το κύριο κλειδί του πίνακα, το table_num στο οποίο βρίσκεται ο αριθμός (όνομα) του τραπεζιού, και το πεδίο

comments στο οποίο μπορεί προαιρετικά ο administrator να εισάγει κάποια σχόλια.

- **Users**: Ο πίνακας των χρηστών του POS αποθηκεύεται εδώ. Το u_id είναι το κύριο κλειδί του πίνακα, το πεδίο name είναι το όνομα του χρήστη, το username το όνομα χρήστη, το pass ο κωδικός του χρήστη για την είσοδο του στο σύστημα, στο πεδίο acc_type βρίσκεται η πληροφορία για τον τύπο χρήστη που ανήκει ο χρήστης, πχ αν ο συγκεκριμένο χρήστης είναι διαχειριστής ή όχι του συστήματος, και παίρνει τις τιμές 0 ή 1 ή 2, και τέλος το email για είσοδο στο σύστημα ή επαναφορά του κωδικού σε περίπτωση απώλειας. Στον πίνακα αυτό έχει πρόσβαση ο Administrator και μόνο, χωρίς όμως να μπορεί να δει τους κωδικούς κανενός χρήστη. Ο πίνακας αυτός είναι επεξεργάσιμος από τον Administrator στην καρτέλα Users (κεφ. 3.5.2) του προγράμματός του.
- **Categories**: Ο πίνακας αυτός περιέχει ένα πεδίο id και ένα πεδίο description. Το cat_id είναι το κύριο κλειδί του πίνακα και το cat_desc (description) είναι η ονομασία της κατηγορίας προϊόντων που μπορεί να υπάρχουν σε ένα μαγαζί, όπως πχ beer, refreshments, snacks, coffee, ice cream κ.α.
- **Menu**: Τον πίνακα αυτόν τον χρησιμοποιούμε για να αποθηκεύσουμε τα δεδομένα του μενού του καταστήματος, στα πεδία m_id που είναι το κυρίως κλειδί του πίνακα, m_type το οποίο αναφέρεται στην ονομασία (τύπος) των προϊόντων του μενού, όπως πχ freddo cappuccino, Heineken, Tequila, Coca Cola κ.α. , το πεδίο cat_id το οποίο μας υποδεικνύει την κατηγορία που ανήκει ο συγκεκριμένος τύπος μενού και συνδέετε άμεσα με πεδίο cat_id του πίνακα categories, και τέλος το πεδίο price στο οποίο αποθηκεύεται η τιμή για το συγκεκριμένο τύπο προϊόντος.
- **Submenu**: Ο πίνακας αυτός αποτελεί κλώνο του πίνακα Menu με τη διαφορά ότι τα δεδομένα που περιέχει δεν έχουν καμία σχέση με τον πίνακα Menu. Περιέχει δεδομένα τύπου γλυκός, μέτριος, με γάλα, tonic, soda, χωρίς πάγο κ.α., τα οποία θεωρούνται υπομενού των τύπων που είναι αποθηκευμένοι στον πίνακα Menu. Τα πεδία που έχει είναι τα αντίστοιχα, δηλαδή s_id, s_type, cat_id, price. Το πεδίο price έχει νόημα σε περίπτωση που κάποιο υπομενού προσθέτει αξία στο μενού που επιλέξαμε. Αν πχ παραγγείλουμε παγωτό μπανάνα, πρέπει να επιλέξουμε στο υπομενού αν αυτό θα αποτελείται από μια, δυο, τρεις ή περισσότερες μπάλες, πράγμα που καθορίζει την τελική τιμή

του προϊόντος. Αντίστοιχα το μενού παγωτό στον πίνακα παγωτό θα έχει τη τιμή μηδέν (0 €) επειδή η τιμή καθορίζεται εξ ολοκλήρου από το submenu.

- **Act_orders**: Είναι ο πίνακας αυτός που περιέχει όλες τις ενεργές παραγγελίες, δηλαδή οτιδήποτε μέχρι εκείνη τη χρονική στιγμή έχει σταλεί σαν παραγγελία από κάποιον client και δεν έχει ακυρωθεί ή πληρωθεί ακόμα. Στον πίνακα αποθηκεύεται σε κάθε γραμμή κάθε ένα προϊόν ξεχωριστά το οποίο επιλέγεται από τον client. Ο πίνακας αυτός χρησιμεύει ώστε ο client να είναι ενήμερος ανά πάσα στιγμή για την κατάσταση οποιουδήποτε τραπεζιού στο μαγαζί, και για τραβήξει η κουζίνα όλα της τα data. Περιέχει για τον σκοπό αυτό τα πεδία order_id το οποίο είναι το κύριο κλειδί του πίνακα, το table_num το οποίο περιέχει το όνομα (αριθμό) του τραπεζιού στο οποίο ανήκει το συγκεκριμένο προϊόν της εγγραφής, το m_id το οποίο είναι ο κωδικός του μενού που περιέχει η εγγραφή, το sp_desc (special description) που περιέχει κάποιο κείμενο που έχει εισαχθεί χειροκίνητα από τον client, το price το οποίο είναι η τιμή του προϊόντος είναι το άθροισμα της τιμής από το πίνακα menu και sub_menu, το u_id που είναι ο μοναδικός κωδικός του User που έστειλε την παραγγελία, το time το οποίο περιέχει το χρονικό στιγμιότυπο στο οποίο μπήκε στον πίνακα η συγκεκριμένη εγγραφή και τέλος το πεδίο done το οποίο χρησιμεύει στην εφαρμογή της κουζίνας, σε περίπτωση που το σύστημα καταρρεύσει απρόσμενα, να γνωρίζει η εφαρμογή ποια προϊόντα ετοιμάστηκαν ούτος ώστε να μην ετοιμαστούν και δεύτερη φορά.
- **S_type_act**: Ο ιδιαίτερος αυτός πίνακας δημιουργήθηκε για να εξυπηρετήσει τις ανάγκες της πολλαπλής επιλογής υπομενού για κάθε ένα μενού. Για παράδειγμα το μενού φραπέ μπορεί να συνδυαστεί με τα υπομενού γλυκός, με γάλα, ελαφρύς. Αυτό σημαίνει ότι θα έπρεπε στον πίνακα act_orders να έχουμε πολλά έξτρα πεδία που στις περισσότερες περιπτώσεις θα ήταν άδεια αλλά μπορεί σε κάποιες να μην ήταν και αρκετά! Ο πίνακας αυτός με τα πεδία order_id και s_id οργανώνει απόλυτα τα πράγματα καθώς μπορούμε να έχουμε πολλαπλές επιλογές από s_id για κάθε order_id δηλαδή για κάθε μενού. Κύριο κλειδί του πίνακα είναι ο συνδυασμός των δύο πεδίων.
- **Fin_orders**: Στον πίνακα αυτό μεταβαίνουν τα data από τον πίνακα act_orders τα οποία χρειαζόμαστε για οργανωτικούς λόγους, και συγκεκριμένα για να εξάγουμε reports για τον Administrator. Περιέχει τα πεδία f_id το οποίο είναι το κύριο κλειδί του πίνακα και τα table_num, m_id,sp_desc, price, u_id,

time, done και day που αντιγράφονται από τον πίνακα act_orders, τη στιγμή που κάποιο προϊόν πληρώνεται από τον πελάτη. Ο πίνακας αυτός παίζει και έναν σημαντικότερο ρόλο. Θα μπορούσαμε να έχουμε ένα γενικό πίνακα orders και να αποφύγουμε έτσι τη χρήση των fin_orders και act_orders, βάζοντας ένα ακόμα πεδίο fin τύπου Boolean στον πίνακα. Αυτό όμως θα σήμαινε πως ο πίνακας orders να μεγάλωνε ολοένα και περισσότερο με αποτέλεσμα ο client να προσπελαύνει ένα μεγάλο πλήθος πληροφορίας για να πάρει τα δεδομένα που θέλει κάθε φορά, τα οποία τις περισσότερες φορές κυμαίνονται γύρω στις 5-10 εγγραφές. Ίσως να είχαμε πρόβλημα τότε καθυστέρησης για την επιστροφή των δεδομένων αν ο πίνακας ήταν τεράστιος και έψαχναν πολλοί χρήστες ταυτόχρονα. Με τη χρήση δύο ξεχωριστών πινάκων, κερδίζουμε σε χρόνο πολύ αφού ο πίνακας act_orders παραμένει πάντα μικρός, και δεν μας πειράζει που ο fin_orders μεγαλώνει συνεχώς διότι χρησιμοποιείται μόνο από τον administrator, ίσως ούτε και από αυτόν κάποιες φορές, για να πάρει πληροφορίες σχετικά με την κίνηση της ημέρας, άρα η προσπέλαση του γίνεται ελάχιστες φορές την ημέρα από έναν μόνο χρήστη.

- **Cellar**: Ο πίνακας αυτός είναι διαθέσιμος στο διαχειριστή, στην τελευταία καρτέλα του προγράμματος του. Είναι ο τρόπος με τον οποίο ενημερώνει ο διαχειριστής και ενημερώνεται για την ποσότητα των προϊόντων που έχει ανά πάσα στιγμή στο μαγαζί του. Περιλαμβάνει τα πεδία c_id, το οποίο είναι και το κύριο κλειδί, product_type και amount. Όπως είναι κατανοητό ο πίνακας περιλαμβάνει προϊόντα των οποίων η ποσότητα είναι μετρήσιμη. Δεν μπορούμε να υπολογίσουμε σε ένα μαγαζί ποια είναι η ποσότητα ζάχαρης ή καφέ που χρησιμοποιήθηκε στο τέλος της ημέρας με ακρίβεια, ενώ μπορούμε να υπολογίσουμε πόσες μπύρες μάρκας Amstel πουλήθηκαν. Όλα αυτά αναλυτικότερα θα τα δούμε στο κεφάλαιο 3.5.5.
- **Pass_reset**: Χρησιμεύει ο πίνακας αυτός για να κάνει reset κάποιος χρήστης τον κωδικό του που ενδεχομένως τον έχει ξεχάσει. Το email του χρήστη και ένα random string 100 χαρακτήρων αποθηκεύονται στα αντίστοιχα πεδία του πίνακα και γίνονται οι κατάλληλοι έλεγχοι με βάση αυτόν τον πίνακα.
- **Counters**: Με τη βοήθεια αυτού του πίνακα λύνουμε βασικά προβλήματα που αφορούν μεταβλητές τις οποίες θέλουμε πάντα να έχουμε στο πρόγραμμα μας και τις οποίες δεν μπορούμε να διαχειριστούμε με php γιατί τα sessions

τερματίζονται και τότε χάνονται τυχόν μεταβλητές που έχουμε ορίσει. Στατιστικά στοιχεία για τους σεφ, η τιμή του end_of_day αλλά και το security code, είναι παραδείγματα πεδίων του πίνακα. Ο πίνακας περιέχει τις στήλες c_id που είναι και το κύριο κλειδί του πίνακα,c_name όπου βρίσκεται το όνομα των τιμών που θέλουμε να κρατήσουμε στη βάση μας και την amount όπου είναι η τιμή που κρατάμε.

Όλα τα παραπάνω πεδία χρησιμοποιούν διάφορους τύπους μεταβλητών για τη λειτουργία τους. Τα πεδία που περιέχουν κείμενο είναι τύπου varchar(100), τα πεδία που χρησιμοποιούν αριθμούς όπως τα περισσότερα κύρια κλειδιά είναι τύπου int(11), οι τιμές των προϊόντων (price) είναι τύπου float, τα πεδία time είναι τύπου timestamp και τα πεδία της μορφής Boolean όπως το done του πίνακα act_orders είναι τύπου tinyint(1).

Οι σχέσεις οι οποίες χρησιμοποιούνται είναι για να ενωθούν οι πίνακες categories με τον menu και τον submenu μεταξύ τους και είναι τύπου ένα προς πολλά, όπως είναι και οι σχέση του s_type_act με τον act_orders. Σε περίπτωση διαγραφής ή ανανέωσης κάποιου category από το διαχειριστή του συστήματος έχεις οριστεί να γίνει επέκταση (cascade) της ενέργειας και στους πίνακες menu και submenu αν περιέχουν το συγκεκριμένο category. Το ίδιο ισχύει και για τον πίνακα s_type_act, ο οποίος αδειάζει με αυτό τον τρόπο κάθε φορά που μια εγγραφή του πίνακα act_orders φεύγει από τον πίνακα.

2.3. ΣΥΝΔΕΣΗ ΣΤΗ ΒΑΣΗ

Με τον κώδικα που βλέπουμε παρακάτω μπορούμε να πραγματοποιήσουμε σύνδεση στη βάση δεδομένων μας μέσα από την php για να εκτελέσουμε τα ερωτήματα μας, τα update ή τα insert μας.

Αρχικά ορίζουμε ότι οι μεταβλητές που θα χρησιμοποιηθούν είναι τύπου global για να τις ψάξει ο Server έξω από το συγκεκριμένο αρχείο, όπου χρειαστεί. Είναι όλες δηλωμένες στο αρχείο common.php. Με τη χρήση της mysqli_connect συνδεόμαστε στη βάση περνώντας τις global μεταβλητές σαν παραμέτρους στη συνάρτηση. Ελέγχουμε το μήνυμα που θα επιστρέψει η συνάρτηση σε περίπτωση αποτυχίας σύνδεσης αλλιώς επιστρέφουμε τη μεταβλητή $con όπου αρχικά είχαμε αποθηκεύσει το αποτέλεσμα της σύνδεσης. Τη μεταβλητή αυτή τη χρησιμοποιούμε όπου χρειαζόμαστε να γίνει

σύνδεση σε άλλα αρχεία php στους υπόλοιπους φακέλους του προγράμματος και για να τερματίσουμε τη σύνδεση με τη χρήση της mysqli_close($con);

```php
//connects to database using credentials found in commons.php
function dbConnect(){
    global $DB_SERVER,$DB_USER,$DB_PASS,$DB_NAME;

    //connect to database
    $con=mysqli_connect($DB_SERVER,$DB_USER,$DB_PASS,$DB_NAME);

    if (mysqli_connect_errno()) {
        if ($DEBUG){
            die('Failed to connect to MySQL: ' . mysqli_connect_error());
        }else{
            die('Fatal error occured!');
        }
    }

    return $con;
}
```

Εικόνα 4: dbConnect() στο data.php

OLAP

OLAP είναι τα αρχικά των λέξεων On Line Analytical Processing. Στα πλαίσια της συγκεκριμένης πτυχιακής εργασίας η OLAP τεχνολογία δεν έχει χρησιμοποιηθεί. Με τη χρήση OLAP θα κερδίσουμε :

- Ευέλικτη υψηλής απόδοσης πρόσβαση και ανάλυση μεγάλου όγκου σύνθετων δεδομένων από διαφορετικές εφαρμογές.
- Ειδικού τύπου ερωτήσεις.
- Οπτικοποίηση / στατιστική ανάλυση / πολυδιάστατη ανάλυση.

Λειτουργικά χαρακτηριστικά απαιτήσεων OLAP:

- Πρόσβαση σε μεγάλο όγκο δεδομένων
- Συμμετοχή αθροιστικών και ιστορικών δεδομένων σε πολύπλοκες ερωτήσεις.
- Μεταβολή της οπτικής γωνίας ή βαθμού αφαίρεσης παρουσίασης των δεδομένων (π.χ., από πωλήσεις ανά περιοχή-> πωλήσεις ανά τμήμα κλπ.).
- Συμμετοχή πολύπλοκων υπολογισμών(π.χ. στατιστικές συναρτήσεις)
- Γρήγορη απάντηση σε οποιαδήποτε χρονική στιγμή τεθεί ένα ερώτημα(“On-Line”)

Μελλοντικά θα πρέπει να υλοποιηθεί η τεχνολογία OLAP στο συγκεκριμένο πρόγραμμα για να θεωρηθεί αξιόπιστο εργαλείο λήψης αποφάσεων και στατιστικών στοιχείων.[5]

Γνωστά εργαλεία υλοποίησης OLAP είναι:

- phpMyOlap (http://sourceforge.net/projects/phpmyolap/)
- RapidMiner (http://sourceforge.net/projects/rapidminer/?source=directory)
- Pentaho (http://sourceforge.net/projects/pentaho/?source=directory)
- Alfresco Audit Analysis and Reporting (http://sourceforge.net/projects/aaar/?source=directory)

ΕΠΙΛΟΓΟΣ

Συμπερασματικά θα λέγαμε πως αυτό που μας μένει από την δημιουργία και επεξεργασία της βάσης μας είναι πως συνεχώς αναπτύσσονται ανάγκες ελέγχου και εύρεσης της πληροφορίας με αποτέλεσμα οι υπάρχοντες πίνακες να γεμίζουν με νέα πεδία και η βάση μας με νέους πίνακες. Με τη χρήση του phpMyAdmin απλουστεύσαμε την δημιουργία και επεξεργασία της βάσης μιας εφ όσον το είχαμε σαν περιεχόμενο του πακέτου XAMPP.

Μελλοντικά και πέρα από τα πλαίσια της συγκεκριμένη πτυχιακής εργασίας είναι απαραίτητο να μελετηθεί η προσθήκη της τεχνολογία OLAP στο συγκεκριμένο πρόγραμμα. Με τη χρήση OLAP θα έχουμε πρόσβαση σε περισσότερα στατιστικά στοιχεία και επίσης θα μπορούμε να αναλύσουμε υψηλού όγκου δεδομένα για τη λήψη αποφάσεων.

3. ΚΕΦΑΛΑΙΟ ΤΡΙΤΟ

ADMINISTRATOR UI

ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο θα αναλύσουμε τους λόγους για τους οποίους επιλέξαμε στη συγκεκριμένη πτυχιακή εργασία να χρησιμοποιήσουμε το XAMPP ως το βασικό εργαλείο για να στήσουμε το Server τον οποίο χρειαζόμαστε για την εφαρμογή μας, καθώς και τα πλεονεκτήματα τα οποία θα έχουμε χρησιμοποιώντας τις δυνατότητες αυτής της πλατφόρμας. Θα αναλύσουμε λεπτομερώς όλα τα συστατικά στοιχεία της εφαρμογής (ιστοσελίδας) που γράφτηκαν για τις ανάγκες του Administrator του POS εξηγώντας τα περισσότερα σημεία του κώδικα της εφαρμογής. Τέλος θα παρουσιαστεί και το Interface της εφαρμογής για κάθε λειτουργία που μπορεί να διεκπεραιώσει ο Administrator.

3.1. ΓΙΑΤΙ XAMPP;

XAMPP είναι τα αρχικά των λέξεων Cross Platform (X), Apache (A), My SQL (M), PHP (P), Perl programming languages (P). Ο ορισμός και μονό του XAMPP αρκεί για να κατανοήσουμε και τον λόγο για τον οποίο θα το χρησιμοποιήσουμε στη συγκεκριμένη πτυχιακή. Ο βασικότερος λόγος είναι η ύπαρξη του Apache Web Server τον οποίο χρησιμοποιούμε για τη διαχείριση του POS που τρέχει σε PC. Επίσης η συνύπαρξη όλων των υπόλοιπων δυνατοτήτων που θα χρειαστούμε για τη δημιουργία ενός server, η My SQL για τη δημιουργία της βάσης δεδομένων του POS όπως επίσης και η PHP για την επικοινωνία του χρήστη με τη βάση μέσω των Services, καθιστούν το XAMPP εύκολη και αναγκαία λύση. Επίσης το κόστος χρήσης του XAMPP είναι μηδενικό αφού διανέμεται δωρεάν.[1]

3.1.1. APACHE

Ο Apache είναι ο πλέον γνωστός και αξιόπιστος Web Server με πολλές δυνατότητες για τον τελικό χρήστη. Είναι ανοικτό λογισμικό, πράγμα που σημαίνει ότι μπορούμε να έχουμε πρόσβαση σε όλο τον πηγαίο κώδικα του. Αυτό με τη σειρά του, μας δίνει την δυνατότητα να δημιουργήσουμε μια

εξειδικευμένη έκδοση του server ανάλογα με τις απαιτήσεις μας. Τέλος, και ίσως ο πιο σημαντικός λόγος που έγινε χρήση αυτού του λογισμικού στην παρούσα πτυχιακή εργασία είναι φυσικά το κόστος, μιας και διατίθεται δωρεάν στην διεύθυνση http://httpd.apache.org/download.cgi. Αν επισκεφτούμε την ιστοσελίδα του Apache HTTD στην διεύθυνση http://httpd.apache.org/ θα δούμε τις τρέχουσες εκδόσεις του Apache. Ξεκινάνε από την 1.3, 2.0, 2.2 και τέλος 2.4. Στην παρούσα πτυχιακή εργασία χρησιμοποιήθηκε ο Apache 2.4.10 μιας και ήταν η τελευταία έκδοση όταν ξεκίνησε η κατασκευή του ηλεκτρονικού καταστήματος. Τα χαρακτηριστικά του Apache 2.2.x περιλαμβάνουν περισσότερη υποστήριξη για φιλτράρισμα, caching, εξισορρόπηση φορτίου και άλλες λειτουργίες του συστήματος.

Ο βασικότερος λόγος που ο Apache Web Server είναι το πιο δημοφιλές προϊόν που χρησιμοποιείται στο διαδίκτυο, οφείλεται στα μεγάλα πλεονεκτήματα που. Ας δούμε μερικά από τα πλεονεκτήματα του Apache που οι χρήστες τον προτιμούν ως κύριο εργαλείο διακομιστή:

- Τα προηγμένα χαρακτηριστικά του- Ο Apache Web Server πάντα καινοτομεί και είναι σε θέση να χρησιμοποιεί τα πιο πρόσφατα πρωτόκολλα που χρησιμοποιούνται στο διαδίκτυο.
- Ευελιξία- Ο Apache μπορεί να προσαρμοστεί πολύ εύκολα λόγω της αρθρωτής του δομής.
- Ευκολία στη διαχείριση- Η διαχείριση είναι ίσως το κυριότερο στοιχείο σε κάθε Server. Ο Apache διαθέτει λίστα με αρχεία ρυθμίσεων, πολύ καλά τεκμηριωμένα, με όλες τις απαραίτητες πληροφορίες έτσι ώστε να είναι δυνατή η προσπέλαση και η ενημέρωση για όλα τα χαρακτηριστικά και τις ρυθμίσεις του διακομιστή.
- Ο Apache είναι ανοικτός-Ένα από τα πλεονεκτήματα του Apache είναι το γεγονός πως είναι επεκτάσιμο εργαλείο. Ανήκει στην κατηγορία ανοικτού κώδικα, πράγμα που σημαίνει πως ο χρήστης μπορεί να προσθέσει επεκτάσεις(extensions) γραμμένες από τον ίδιο. Μπορεί επίσης να γράψει εκ νέου κώδικα προσαρμοσμένο στις δυνατότητες του Apache και στις ανάγκες του ίδιου του χρήστη.
- Αποτελεσματικότητα- Η αποτελεσματικότητα του Apache είναι ένα πολύ σημαντικό προσόν για έναν web server. Έχει καταφέρει πράγματα που

όλοι οι άλλοι διακομιστές δεν μπορούν να κάνουν. Οι προσπάθειες για ένα πιο βελτιστοποιημένο server ήταν επιτυχής. Σήμερα μπορούμε να έχουμε έναν πολύ σταθερό και ώριμο web server με μεγάλη αποδοτικότητα όσο κανένας άλλος server στον κόσμο.

- Υποστήριξη-Το μεγαλύτερο όφελος της κοινότητας του ανοικτού κώδικα είναι η υποστήριξη στις κορυφαίες εκδόσεις λογισμικών, η οποία είναι πολύ μεγαλύτερη από των λογισμικών της αγοράς.
- Φορητότητα-Τέλος, ο Apache προσφέρει εξαιρετική φορητότητα, αφού μπορεί να εγκατασταθεί και να λειτουργήσει κάτω από πολλαπλές πλατφόρμες με υψηλό επίπεδο φορητότητας.

3.1.2. My SQL

Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων. Εκεί μπορείτε να προσθέσετε, να ανακτήσετε και να διαχειριστείτε πληροφορίες που είναι αποθηκευμένες σε μια βάση δεδομένων. Η σχεσιακή MySQL σημαίνει ότι μια πληροφορία αποθηκεύεται σε χωριστούς πίνακες και όχι σε έναν μεγάλο πίνακα. Μπορούν να καθιερωθούν σχέσεις μεταξύ πινάκων και να ανακτούνται πληροφορίες χρησιμοποιώντας δομημένη γλώσσα διατύπωσης ερωτήσεων (SQL).

Πλεονεκτήματα της MySQL:

Η MySQL μας προσφέρει ορισμένα πλεονεκτήματα, όπως χαμηλό κόστος, εύκολη διαμόρφωση και μάθηση και ο κώδικας προέλευσης είναι διαθέσιμος. Παρακάτω αναλύουμε ορισμένα από αυτά.

- **Απόδοση**

Η MySQL είναι χωρίς αμφιβολία γρήγορη. Μπορείτε να δείτε την σελίδα δοκιμών http://web.mysql.com/benchmark.html. Πολλές από αυτές τις δοκιμές δείχνουν ότι η MySQL είναι αρκετά πιο γρήγορη από τον ανταγωνισμό.

- **Χαμηλό κόστος**

Η MySQL είναι διαθέσιμη δωρεάν, με άδεια ανοικτού κώδικα (Open Source) ή με χαμηλό κόστος , αν πάρετε εμπορική άδεια, αν απαιτείται από την εφαρμογή σας.

- **Ευκολία Χρήσης**

Οι περισσότερες μοντέρνες εφαρμογές χρησιμοποιούν SQL. Αν έχετε χρησιμοποιήσει ένα άλλο σύστημα διαχείρισης βάσεων δεδομένων δεν θα έχετε πρόβλημα να προσαρμοστείτε σε αυτό.

- **Μεταφερσιμότητα**

Η MySQL μπορεί να χρησιμοποιηθεί σε πολλά διαφορετικά συστήματα Unix όπως επίσης και στα Microsoft Windows .

- **Κώδικας Προέλευσης**

Όπως και με την PHP , μπορείτε να πάρετε και να τροποποιήσετε τον κώδικα προέλευσης της MySQL.

- **Νέα έκδοση**

Η νέα έκδοση MySQL 5 έχει έρθει με νέες λειτουργίες. Είναι πλέον ικανή να υποστηρίξει πολύ μεγάλα projects με υψηλή αξιοπιστία.

Κατά τη διάρκεια συγγραφής αυτής της πτυχιακής εργασίας χρησιμοποιήθηκε η MySQL 5.6.20

3.1.3. PHP

Η PHP είναι μια γλώσσα προγραμματισμού για τη δημιουργία σελίδων web με δυναμικό περιεχόμενο. Μια σελίδα PHP περνά από επεξεργασία από ένα συμβατό διακομιστή του Παγκόσμιου Ιστού (πχ Apache), ώστε να παραχθεί σε πραγματικό χρόνο το τελικό περιεχόμενο, που θα σταλεί στο πρόγραμμα περιήγησης των επισκεπτών σε μορφή κώδικα HTML.

Πλεονεκτήματα της PHP

Κάποιοι από τους βασικούς ανταγωνιστές της PHP είναι ο Perl, Microsoft Active Server Pages (ASP), Java Server Pages (JSP) και Allaire Cold Fusion . Σε σύγκριση με αυτά τα προϊόντα, η PHP έχει πολλά πλεονεκτήματα όπως:

- **Υψηλή απόδοση**

Η PHP είναι πολύ αποτελεσματική. Με ένα φθηνό διακομιστή μπορείτε να εξυπηρετήσετε εκατομμύρια επισκέψεων καθημερινά. Οι δοκιμές που δημοσιεύθηκαν από την Zend Technologies (http://www.zend.com), δείχνουν ότι η PHP ξεπερνά τους ανταγωνιστές της.

- **Διασυνδέσεις με πολλά διαφορετικά συστήματα βάσεων δεδομένων**

Η PHP έχει εγγενείς συνδέσεις για πολλά συστήματα βάσεων δεδομένων. Εκτός από την MySQL μπορείτε να συνδεθείτε με τις βάσεις δεδομένων PostgreSQL, mSQL Oracle, dbm, filePro, Informix, InterBase, Sybase, κ.α. Χρησιμοποιώντας το Open Database Connectivity Standard (ODBC) μπορείτε να συνδεθείτε σε οποιαδήποτε βάση δεδομένων παρέχει ένα πρόγραμμα οδήγησης ODBC . Αυτό περιλαμβάνει και τα προϊόντα της Microsoft products, μεταξύ άλλων.

- **Ενσωματωμένες βιβλιοθήκες για πολλές συνηθισμένες Web διαδικασίες**

Επειδή η PHP σχεδιάστηκε για να χρησιμοποιείται στο Web , έχει πολλές ενσωματωμένες βιβλιοθήκες, που εκτελούν πολλές χρήσιμες λειτουργίες σχετικές με το Web. Μπορείτε να δημιουργήσετε εικόνες GIF δυναμικά , να συνδεθείτε με άλλες υπηρεσίες δικτύων, να στείλετε ηλεκτρονικό ταχυδρομείο, να δουλέψετε με cookies και να δημιουργήσετε PDF έγγραφα : όλα αυτά με λίγες γραμμές κώδικα

- **Χαμηλό κόστος**

Η PHP είναι δωρεάν . Μπορείτε να κατεβάσετε την τελευταία έκδοση από το http://www.php.net, χωρίς χρέωση.

- **Ευκολία μάθησης και χρήσης**

Η σύνταξη της PHP βασίζεται σε άλλες γλώσσες προγραμματισμού ,βασικά στην C και στην Perl.

• **Μεταφερσιμότητα**

Η PHP είναι διαθέσιμη για πολλά λειτουργικά συστήματα. Μπορείτε να γράψετε κώδικα PHP για δωρεάν συστήματα τύπου Unix , όπως LINUX και FreeBSD, για εμπορικές εκδώσεις του UNIX, όπως το Solaris και το IRIX ή για διαφορετικές εκδώσεις των Microsoft Windows. Ο κώδικα σας συνήθως θα δουλεύει χωρίς αλλαγές στα συστήματα που τρέχουν την PHP.

• **Διαθεσιμότητα του κώδικα προέλευσης**

Έχετε πρόσβαση στον κώδικα προέλευσης της PHP. Αντίθετα με εμπορικά, κλειστά προγράμματα, αν υπάρχει κάτι που θέλετε να αλλάξετε ή να προσθέσετε στη γλώσσα, μπορείτε να το κάνετε. Δεν χρειάζεται να περιμένετε τον κατασκευαστή να εμφανίσει διορθώσεις. Δεν θα ανησυχείτε αν ο κατασκευαστής θα σταματήσει να υπάρχει ή αν θα σταματήσει να υποστηρίζει το προϊόν.

3.2. ΛΕΙΤΟΥΡΓΙΕΣ ADMINISTRATOR UI

Οι λειτουργίες που μπορούν να εκτελεστούν από τη μεριά του Administrator είναι αρκετές και σημαντικές. Κατά την εκκίνηση του XAMPP και πατώντας localhost στο πεδίο διευθύνσεων του browser, μεταφερόμαστε στο Login Screen της εφαρμογής μας όπου καλούμαστε να εισάγουμε τα στοιχεία μας για να πραγματοποιήσουμε είσοδο στην εφαρμογή. Αναλόγως με τον τύπο του λογαριασμού μας θα μεταφερθούμε είτε στο Administrator UI είτε στο Chef UI το οποίο θα αναλύσουμε στο επόμενο κεφάλαιο.

Υπάρχει η δυνατότητα αλλαγής του κωδικού πρόσβασης μέσω authentication με e-mail σε περίπτωση που έχει ξεχάσει ο Administrator τον κωδικό του από μέσω του link “Forgot you Password?” και εν συνεχεία χρησιμοποιώντας το Reset Screen.

Ο Administrator έχει πρόσβαση σε πολλούς πίνακες της βάσης δεδομένων του οποίους μπορεί να επεξεργαστεί για να διαμορφώσει το σύνολο των λειτουργιών του μαγαζιού του. Οι λειτουργίες αυτές είναι η προσθαφαίρεση τραπεζιών στο POS, η οποία είναι διαθέσιμη στην πρώτη καρτέλα (Tables) του Administrator UI, η προσθαφαίρεση χρηστών που βρίσκεται στη δεύτερη καρτέλα (Users), η προσθαφαίρεση και παραμετροποίηση του συνολικού

μενού του μαγαζιού στην τρίτη καρτέλα (Menu) , η λήψη στατιστικών πληροφοριών και αναφορών για την κίνηση προϊόντων αλλά και τη δραστηριότητα των χρηστών του POS που βρίσκεται στη τέταρτη καρτέλα (Reports) και επίσης η αναθεώρηση και προσθαφαίρεση ποσοτήτων των προϊόντων στην αποθήκη (Cellar).

Τέλος υπάρχει η δυνατότητα παρακολούθησης και αλληλεπίδρασης με την εφαρμογή της κουζίνας (Chef UI) πατώντας το κουμπί Move to chef app, η δυνατότητα τερματισμού της ημέρας πατώντας το κουμπί End of Day και η λήψη του κωδικού ασφαλείας του POS πατώντας το κουμπί Security Code, του οποίου τη χρησιμότητα θα εξηγήσουμε στο κεφάλαιο που αφορά την εφαρμογή Android για τον Client.

3.3. ΟΡΓΑΝΩΣΗ ΦΑΚΕΛΩΝ - ΑΡΧΕΙΩΝ

Τα αρχεία του προγράμματος μας όπως και όλοι οι χρήστες της πλατφόρμας XAMPP, είναι οργανωμένα κάτω από τον κατάλογο htdocs που βρίσκεται στην τοποθεσία C:\xampp\htdocs. Εκεί έχουμε τους φακέλους presentation, όπου είναι και τα περισσότερα αρχεία μας που χρησιμοποιούμε για το login screen, το reset password, το manage της βάσης από τον administrator και άλλα αρχεία php, τα οποία συνδέονται μεταξύ τους για τη σωστή λειτουργία του προγράμματος μας, τον css όπου όπως δηλώνει και η λέξη περιλαμβάνει τα αρχεία μορφοποίησης των σελίδων μας, τον logic όπου βρίσκονται υλοποιημένες οι περισσότερες αν όχι όλες οι μέθοδοι που χρειαζόμαστε να χρησιμοποιήσουμε στα αρχεία μας στο presentation, τον data όπου βρίσκονται υλοποιημένες μέθοδοι που χρησιμοποιούμε σε κάθε σχεδόν αρχείο μας όπως είναι η σύνδεση στη βάση δεδομένων και τα select ερωτήματα που κάνουμε προς τη βάση για να πάρουμε τις πληροφορίες που χρειαζόμαστε, τον img όπου έχουμε αποθηκευμένες τις εικόνες που χρησιμοποιούμε στο πρόγραμμά μας και τέλος ο φάκελος js όπου βρίσκονται όλα τα αρχεία με κώδικα JQUERY που χρησιμοποιούμε, καθώς και το αρχείο JQuery.js, το οποίο είναι μια βιβλιοθήκη JavaScript σχεδιασμένη να απλοποιήσει τη υλοποίηση σεναρίων (scripting) στη πλευρά του πελάτη (client-side) της HTML και υποστηρίζει πολλαπλούς φυλλομετρητές Ιστού, το όποιο το βρίσκουμε έτοιμο στο διαδίκτυο, και συγκεκριμένα πατώντας το link http://jquery.com/download/ και το χρησιμοποιούμε ανάλογα με τις ανάγκες

μας. Με το συγκεκριμένο τρόπο οργάνωσης διευκολύνουμε τον τρόπο με τον οποίο ο προγραμματιστής θα κάνει μεταγενέστερες αλλαγές στον κώδικα του προγράμματος χωρίς να χρειάζεται να ανοίξει και να αλλάξει όλα τα αρχεία, πχ αν αλλάξει ο κωδικός του χρήστη root στη βάση δεδομένων μας αρκεί να ανοίξουμε το αρχείο common.php στον κατάλογο logic το οποίο περιλαμβάνει global μεταβλητές τις οποίες χρησιμοποιούμε συχνά σε άλλα αρχεία μας, και να κάνουμε την αλλαγή που επιθυμούμε.

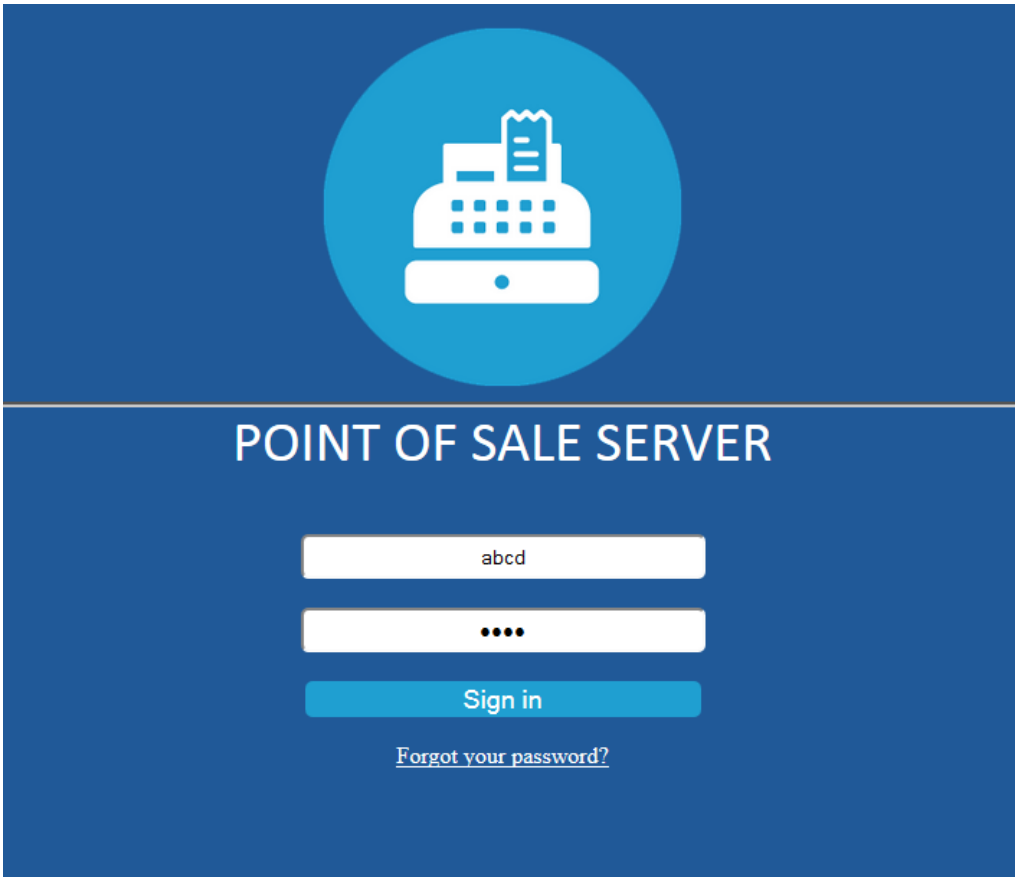
```php
<?php
    //Contains settings as global variables

    //database settings
    $DB_SERVER = 'localhost';
    $DB_USER = 'root';
    $DB_PASS = '';
    $DB_NAME = 'posdb';

    //server hostname
    $LOCAL_SERVER = 'localhost';

    //account recovery (email settings)
    $SMTP_SERVER = "smtp.live.com";
    $SMTP_AUTH = true;
    $SMTP_SECURE = "tls";
    $SMTP_PORT = 587;

    $SMTP_USERNAME = 'service_pos_service@hotmail.com';
    $SMTP_PASSWORD = 'tasospavlos1';

    $SMTP_FROM_EMAIL = 'service_pos_service@hotmail.com';
    $SMTP_FROM_NAME = 'POS Service';

    //Debug options
    $DEBUG = true;

    $PAGING_LIMIT = 10;
?>
```

Εικόνα 5: common.php

3.4. LOGIN SCREEN (Αρχική Σελίδα)

Για να ξεκινήσει η λειτουργία του προγράμματος για τον διαχειριστή ή του προγράμματος για την κουζίνα (chef.php) πρέπει να εκκινήσουμε το XAMPP το οποίο με τη σειρά του εκκινεί τις υπηρεσίες του Apache Server και της MySQL. Αν δεν το κάνουμε αυτό αρχικά δεν θα έχουμε πρόσβαση στο localhost με αποτέλεσμα ο browser να μας πετάξει μήνυμα ότι η ιστοσελίδα δεν είναι διαθέσιμη ή ότι δεν είναι δυνατή η σύνδεση στη τοποθεσία.

Μετά την εκκίνηση του XAMPP και πατώντας το σύνδεσμο localhost μεταφερόμαστε αυτόματα στη σελίδα localhost/presentation/login.php. Εκεί θα δούμε την παρακάτω οθόνη που μας καλεί να εισάγουμε το Username ή το e-mail μας για να μπούμε στην κυρίως εφαρμογή. Κάτω από τη φόρμα εισαγωγής των στοιχείων μας βλέπουμε το hyperlink “Forgot your password?” το οποίο χρησιμεύει στην επαναφορά ενός ξεχασμένου κωδικού πρόσβασης που όμως θα αναλύσουμε στην επόμενη παράγραφο αυτού του κεφαλαίου. Σε αυτό το σημείο αξίζει να σημειωθεί πως όλη η εφαρμογή γύρω από τον administrator και τους σεφ χρησιμοποιεί τη session_start() όπου κρίνεται αναγκαίο για την αποφυγή πρόσβασης σε δεδομένα χωρίς να έχει γίνει login από τον administrator ή χρήστη τύπου chef.[11]



Εικόνα 6: Login Screen

3.4.1. ΕΠΕΞΗΓΗΣΗ ΚΩΔΙΚΑ login.php

Η φόρμα την οποία βλέπουμε στην παραπάνω εικόνα, προκύπτει από τον κώδικα από κάτω, που βρίσκεται στο αρχείο login.php.

```
echo

'
<center><div id="login_div" style="width : 500px;"><form id="login_form" action="login.php" method="post">
<input class="login_inputs" type="text" name="username" placeholder="Username or Email" ></br></br>
<input class="login_inputs" type="password" name="password" placeholder="Password"></br></br>
<input id="login_button" type="submit" value="Sign in">
</form><p><a id="forgot_link" href="/presentation/forgot.php">Forgot your password?</a></p></div><center>';

if (isset($_SESSION['error_login']))
{
    echo '<p id="wrong_input">Wrong username or password!<p></br>';
    unset($_SESSION['error_login']);
}
```

Εικόνα 7: Login Form

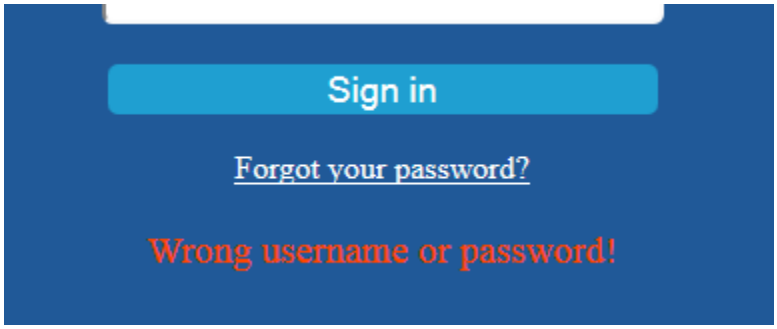
Ένα μήνυμα λάθους θα προκύψει αν το Username ή το E-mail που εισήγαμε σε συνδυασμό με το Password δεν βρίσκονται στη βάση δεδομένων μας (Εικόνα 8). Ο έλεγχος για την ύπαρξη των στοιχείων στη βάση γίνεται με ένα ερώτημα select το οποίο στη βάση αναφέρεται στον πίνακα Users και φαίνεται παρακάτω:

```
//logs in user and redirect to manage screen if login was successfull, or returns false if not successful
function loginUser($username, $password){

    //connect to database
    $con=dbConnect();

    //escape user input before query
    $username = mysqli_real_escape_string($con,$_POST['username']);
    $password = mysqli_real_escape_string($con,$_POST['password']);

    $result = executeQuery($con,"SELECT NAME FROM users where USERNAME='$username' and PASS='$password' and ADMIN='1'");
    $result2 = executeQuery($con,"SELECT NAME FROM users where EMAIL='$username' and PASS='$password' and ADMIN='1'");

    $numrows = mysqli_num_rows($result);
    $numrows2 = mysqli_num_rows($result2);

    if ($numrows == 1){
        $_SESSION['loggedin'] = true;
        $row1 = mysqli_fetch_row($result);
        $_SESSION['user'] = htmlspecialchars($row1[0]);
        header( 'Location: /presentation/manage.php?tab=1' );
    }
    else if ($numrows2 == 1){
        $_SESSION['loggedin'] = true;
        $row2 = mysqli_fetch_row($result2);
        $_SESSION['user'] = htmlspecialchars($row2[0]);
        header( 'Location: /presentation/manage.php?tab=1' );
    }else{
        return false;
    }
}
```

Εικόνα 8: Login queries στο app_logic.php

Αφού πάρουμε σαν είσοδο αυτά που εισήγαγε ο χρήστης με τη χρήση των $_POST[‘username’] και $_POST[‘password’] κάνουμε το ερώτημα στη βάση μας και αναλόγως με τον αριθμό των γραμμών που θα επιστραφούν

αποφασίζουμε αν θα προχωρήσουμε στο login ή αν θα θέσουμε τη μεταβλητή $_SESSION[‘error_login’]= true, για να εμφανιστεί το παρακάτω μήνυμα λάθους στην οθόνη μας:



Εικόνα 9: Wrong input

Όπως βλέπουμε στην εικόνα 7 ο έλεγχος των στοιχείων που δόθηκαν από το χρήστη γίνεται με τη χρήση της function loginUser($username,$password) η οποία βρίσκεται στο αρχείο app_logic.php. Ουσιαστικά καλούμε τη συγκεκριμένη μέθοδο στο login.php για να πραγματοποιήσουμε την είσοδο στην εφαρμογή.

Σε περίπτωση που η εισαγωγή των στοιχείων είναι σωστή, εάν ο λογαριασμός αφορά administrator τότε αυτός θα μεταφερθεί στην επόμενη οθόνη που είναι ουσιαστικά το κυρίως πρόγραμμα όπου μπορεί να έχει πρόσβαση στη βάση δεδομένων αλλά και σε στατιστικά στοιχεία (reports.php). Εάν ο λογαριασμός αφορά κάποιον σεφ τότε αυτός θα μεταφερθεί στο κυρίως πρόγραμμα της κουζίνας όπου θα μπορεί να δει τις επόμενες παραγγελίες προς εκτέλεση.

3.5. FORGOT YOUR PASSWORD?

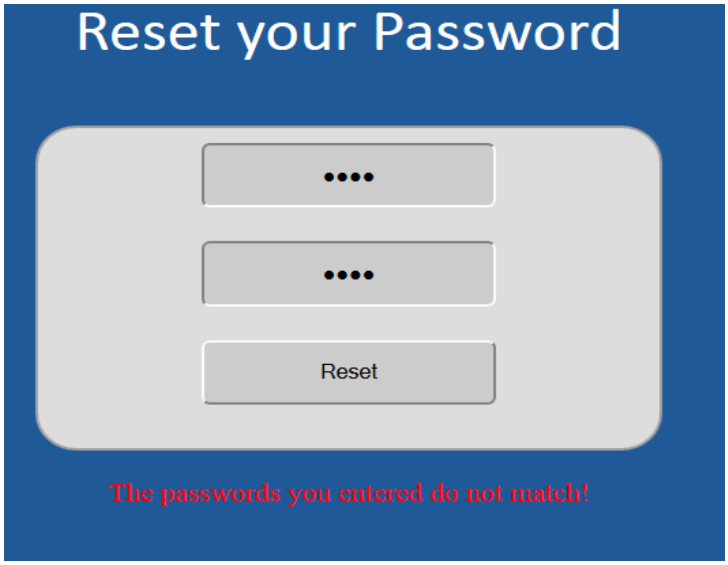
Κάτω από τη φόρμα εισόδου παρατηρούμε το link Forgot your Password?. Όπως γίνεται αντιληπτό είναι ο σύνδεσμος που χρησιμοποιούμε όταν έχουμε ξεχάσει τον κωδικό εισόδου μας. Κατά το γέμισμα της βάσης δεδομένων από τον δημιουργό της εφαρμογής ή και από τον διαχειριστή σε επόμενη φάση κάθε χρήστης που είναι περασμένος στη βάση δεδομένων και συγκεκριμένα στον πίνακα Users έχει ως στοιχείο και το email του το οποίο μπορεί να χρησιμοποιήσει για την είσοδο του στην εφαρμογή ή και για την ανάκτηση του κωδικού του σε περίπτωση απώλειας. Παρακάτω φαίνεται η φόρμα εισαγωγής του mail σε περίπτωση απώλειας του κωδικού.

Εικόνα 10: Forgot form

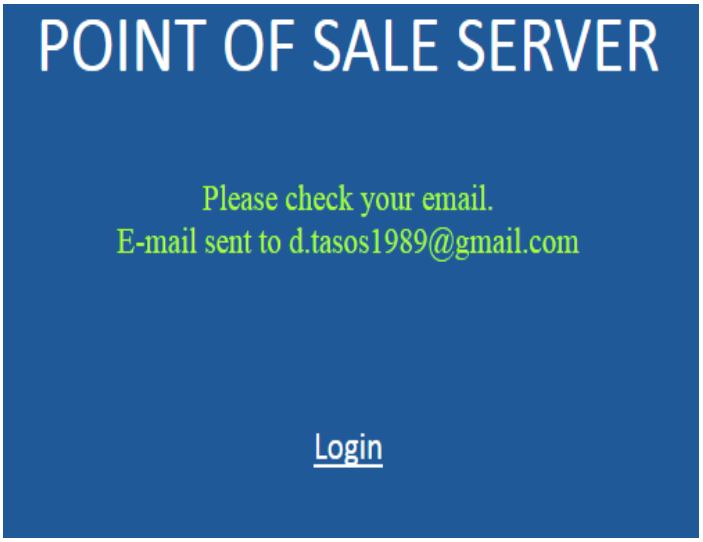
Το email που καλούμαστε να εισάγουμε σε αυτή εδώ τη φόρμα πρέπει να είναι ένα email που υπάρχει στη βάση δεδομένων μας

Αν η εισαγωγή του email δεν πληροί την παραπάνω προϋπόθεση τότε με το αντίστοιχο μήνυμα θα πρέπει ο χρήστης να εισάγει εκ νέου ένα valid email.

Κατά την επιτυχία της συγκεκριμένης φόρμας θα λάβουμε στην οθόνη μας ένα κείμενο επιβεβαίωσης αποστολής email για να επαναφέρουμε το κωδικό μας.



Εικόνα 11: Μήνυμα εσφαλμένης εισαγωγής email



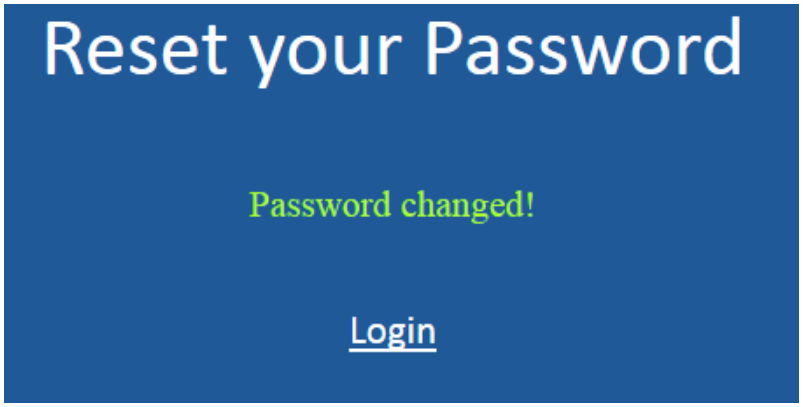
Εικόνα 12: Μήνυμα ορθής εισαγωγής email

Αν ελέγξουμε το email μας πλέον θα παρατηρήσουμε ένα εισερχόμενο μήνυμα με ένα link μεγάλου μήκους και τυχαίων χαρακτήρων το όποιο αν το πατήσουμε μεταφερόμαστε αυτόματα στη σελίδα presentation/reset.php όπου εκεί υπάρχει η φόρμα επαναφοράς του password το οποίο έχουμε ξεχάσει. Βασική προϋπόθεση για να επαναφέρουμε το κωδικό μας είναι να κάνουμε κλικ στο link του τελευταίου email που μας έχει έρθει από την εφαρμογή και όχι από link ανάκτησης του κωδικού παλαιότερου email. Επίσης κατά την εισαγωγή του νέου μας κωδικού πρέπει να ταιριάζουν οι δύο κωδικοί που καλούμαστε να δώσουμε. Αν δεν ταιριάζουν θα λάβουμε το απαραίτητο μήνυμα στην οθόνη μας, όπως και αν ταιριάζουν ένα διαφορετικό μήνυμα.

Εικόνα 13: Passwords don’t match

To reset your password open the following link in your browser
http://localhost/reset.php?email=d.tasos1989%40gmail.com&random=
NowtKebUd3LkUdKO4y7XnpjR5Dk7UyDO32axeqqpGxnZ7KtoKSHdihvxZNTY
xMiAa3HZAZzh1pR2cE4mjAaR6xABMmjc3nPJas5O

Εικόνα 14: Κείμενο τυχαίου string



Εικόνα 15: Passwords match

3.5.1. ΕΠΕΞΗΓΗΣΗ ΚΩΔΙΚΑ forgot.php

Κατά την εισαγωγή ενός έγκυρου email στη φόρμα της σελίδας forgot.php, γίνεται η σύνδεση στη βάση δεδομένων με τον τρόπο που εξηγήσαμε σε προηγούμενο κεφάλαιο και στη συνέχεια εκτελείτε ο κώδικας που φαίνεται παρακάτω από το αρχείο app_logic.php:

```
if ($numrows == 1){

    //create random string token, save it with email in database and send reset link to email
    $characters = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
    $randomString = '';
    for ($i = 0; $i < 100; $i++) {
        $randomString .= $characters[rand(0, strlen($characters) - 1)];
    }

    //delete old info if exist and add new reset info
    executeQuery($con,"DELETE FROM PASS_RESET WHERE EMAIL='$email'");
    executeQuery($con,"INSERT INTO PASS_RESET VALUES('$email','$randomString')");

    sendResetMail($_POST['email'],$randomString);

    return true;
}else{
    return false;
}
```

Εικόνα 16: forgotPassword() στο app_logic.php

- Θέτουμε μια μεταβλητή με όνομα characters η οποία περιλαμβάνει όλους τους αριθμούς και όλους τους χαρακτήρες, πεζούς και κεφαλαίους.
- Θέτουμε μια μεταβλητή με όνομα randomString η οποία είναι κενή.
- Δημιουργούμε με μια for ένα τυχαίο String 100 χαρακτήρων και το

αποθηκεύουμε στη μεταβλητή randomString.

- Στην συνέχεια εισάγουμε στον πινάκα PASS_RESET το email που πήραμε από τη φόρμα, $_POST['email'], και το randomString που μόλις δημιουργήθηκε, αφού πρώτα σβήσουμε κάθε εγγραφή του πίνακα που περιέχει ήδη το συγκεκριμένο mail που δόθηκε. Αυτό γίνεται για να μπορούμε να κάνουμε reset μόνο με το link από το τελευταίο mail που έχουμε πάρει από τη εφαρμογή.
- Αποστέλλουμε το email με τη χρήση της sendResetMail().
- Τέλος εμφανίζεται το μήνυμα επιβεβαίωσης (Εικόνα 12).

```
echo '<p id="success_msg">Please check your email.<br />E-mail sent to  '. $_POST['email'].'</p><br />';
echo '</br></br>';
echo '<center><a id="login" href="/login.php">Login</a></center>';
```

Εικόνα 17: Μήνυμα επιβεβαίωσης αποστολής e-mail

Αποστολή email

Παρακάτω φαίνεται ο κώδικας που χρησιμοποιούμε για να στείλουμε το email στο διαχειριστή για να επαναφέρει τον κωδικό του. Αυτός δεν είναι ο μόνος κώδικας που χρησιμοποιείται καθώς θα δούμε ότι περιέχουμε στον κώδικα μας (include) κώδικα και από άλλες κλάσεις οι οποίες δεν είναι δυνατόν να αναλυθούν λόγω του τεράστιου μεγέθους τους. Ο παρακάτω κώδικας υπάρχει έτοιμος στο διαδίκτυο σε περίπτωση που κάποιος επιθυμεί να στείλει email μέσω php από τον υπολογιστή του (localhost). Αρκεί κάποιος να αναζητήσει το PHPmailer στο Google ή να ακολουθήσει το link http://phpmailer.worxware.com/index.php?pg=sf&p=sfp. Για τις ανάγκες της δικής μας πτυχιακής αλλάξαμε λίγο τον κώδικα αυτόν για να δουλέψει όπως εμείς χρειαζόμασταν.

Αρχικά, καλούμε την class.phpmailer.php αν δεν την έχουμε καλέσει πάλι με την require once και κάνουμε include την class.smtp.php. Και οι δύο αυτές κλάσεις διανέμονται δωρεάν στο διαδίκτυο. Η ανάλυση των δυο αυτών κλάσεων δεν θα γίνει στα πλαίσια της συγκεκριμένης πτυχιακής εργασίας. Στη συνέχεια αντικαθιστούμε στον κώδικα τα δικά μας στοιχεία για να κάνουμε το Mail να «φύγει» από το localhost. Φυσικά τη δουλειά για μας την κάνει ο smtp server του hotmail στη συγκεκριμένη περίπτωση όπου και στέλνουμε τα

στοιχεία που χρειάζονται. Οι μεταβλητές που πειράξαμε στον παρακάτω κώδικα είναι οι εξής:

```php
function sendResetMail($email,$randomString){

    //send email
    require_once('class.phpmailer.php');
    require_once('class.smtp.php');

    global $LOCAL_SERVER,$SMTP_SERVER,$SMTP_AUTH,$SMTP_SECURE,$SMTP_PORT,$SMTP_USERNAME,$SMTP_PASSWORD;
    global $SMTP_FROM_EMAIL,$SMTP_FROM_NAME;

    $mail = new PHPMailer();

    $server = $LOCAL_SERVER;

    $body = "To reset your password open the following link in your browser \n
    http://$server/presentation/reset.php?email=" . urlencode($email) . "&random=$randomString";

    $mail->IsSMTP(); // telling the class to use SMTP
    $mail->Host       = $SMTP_SERVER; // SMTP server
    $mail->SMTPDebug  = false;                     // enables SMTP debug information (for testing)
                                                   // 1 = errors and messages
                                                   // 2 = messages only
    $mail->SMTPAuth   = $SMTP_AUTH;                  // enable SMTP authentication
    $mail->SMTPSecure = $SMTP_SECURE;

    $mail->Port       = $SMTP_PORT;                    // set the SMTP port for the GMAIL server
    $mail->Username   = $SMTP_USERNAME; // SMTP account username
    $mail->Password   = $SMTP_PASSWORD; // SMTP account password


    $mail->SetFrom($SMTP_FROM_EMAIL, $SMTP_FROM_NAME);

    $mail->Subject    = "POS reset admin password";

    $mail->Body = $body;

    $address = $email;
    $mail->AddAddress($address, "POS User");

    $mail->Send();
}
```

Εικόνα 18: sendResetMail() στο app_logic.php

- **$body** : εδώ είναι το κυρίως κείμενο το οποίο στέλνουμε μέσα στο μήνυμα και περιλαμβάνει τη σελίδα που θα ανοίξει μόλις πατηθεί το link, το email του χρήστη που έχει κάνει την αίτηση για επαναφορά κωδικού και το randomString των εκατό χαρακτήρων. Αυτά θα τα διαχειριστούμε σαν μεταβλητές με την $_GET[] παρακάτω στην reset.php.

- **$mail->Host = "smtp.live.com";** Θέτουμε τη διεύθυνση του smtp server που θα κάνει τη δουλειά για εμάς και θα στείλει το mail στο χρήστη που θέλει να κάνει επαναφορά κωδικού.

- **$mail->Port = 587;** Θέτουμε το port που χρησιμοποιεί ο εκάστοτε smtp server. Στην περίπτωση μας 587.

- **$mail->Username = "service_pos_service@hotmail.com";** Θέτουμε ένα mail το οποίο πρέπει να είναι δικό μας και πραγματικό από το οποίο θα στείλει ο smtp server το email.

- **$mail->Password = "tasospavlos1";** Εδώ χρειάζεται και το password του email που θα χρησιμοποιήσουμε ως αποστολέας.

- **$mail->SetFrom('service_pos_service@hotmail.com', 'POS Service');** Εδώ ορίζουμε το όνομα του αποστολέα που θα φαίνεται στον παραλήπτη, και επιλέγουμε το POS Service για όνομα αποστολέα και το πραγματικό μας mail από το όποιο στείλαμε το mail.

- **$mail->Subject = "POS reset password";** Ορίζουμε το θέμα του email εδώ.

- **$address = $_POST['email'];** Σε αυτή τη μεταβλητή θέτουμε το email στο οποίο ο smtp server θα στείλει το email και είναι αυτό που έχει εισάγει ο χρήστης.

- **$mail->AddAddress($address, "POS User");** Περνάμε σαν παράμετρο τη διεύθυνση του χρήστη και το όνομά του στην μέθοδο mail.

- **$mail->Send();** Αποστολή του mail.

3.5.2. ΕΠΕΞΗΓΗΣΗ ΚΩΔΙΚΑ reset.php

Όπως αναφέραμε παραπάνω, θα μεταφερθούμε αυτόματα στη σελίδα localhost/presentation/reset.php μόλις πατήσουμε το link ανάκτησης του κωδικού μας από το εισερχόμενο email που οι ίδιοι στείλαμε. Παρακάτω θα εξηγήσουμε πως λειτουργεί στο συγκεκριμένο σημείο η εφαρμογή μας.

```php
function checkResetInfo(){
    //connect to database
    $con = dbConnect();

    //check if email and random string are in database
    $email = mysqli_real_escape_string($con,$_GET['email']);
    $random = mysqli_real_escape_string($con,$_GET['random']);

    $result = executeQuery($con,"SELECT * FROM PASS_RESET WHERE EMAIL='$email' AND RANDOM='$random'");

    $numrows = mysqli_num_rows($result);

    //if reset email and random string are valid
    if ($numrows == 1){
        $_SESSION['reset_email'] = $email;
        $_SESSION['reset_random'] = $random;
    }else{
        //redirect to homepage
        header('Location: /presentation/login.php');
    }
}
```

Εικόνα 19: checkResetInfo() στο applogic.php

Αρχικά ελέγχεται αν υπάρχουν οι μεταβλητές email και random στο link το οποίο πατήσαμε από το email. Αν όντως υπάρχουν, τότε τις αποθηκεύουμε στις μεταβλητές $email και $random με τις $_GET[‘email’] και $_GET[‘random’] αντίστοιχα. Στη συνέχεια αποθηκεύουμε στη μεταβλητή $result το αποτέλεσμα του ερωτήματος το οποίο επιστρέφει τη γραμμή του πίνακα PASS_RESET όπου ταιριάζουν το email πήραμε από το link που πατήθηκε και το String των 100 χαρακτήρων. Με άλλα λόγια η τιμή της μεταβλητής που θα επιστραφεί μπορεί να είναι και null αν ο χρήστης κατά λάθος έχει πατήσει το link από παλαιότερο email ανάκτησης του κωδικού του. Αυτό συμβαίνει γιατί όπως είπαμε παραπάνω ο πινάκας PASS_RESET γεμίζει κατά την αποστολή του email στο χρήστη οπότε είναι πάντα ενημερωμένος με το τελευταίο String των εκατό χαρακτήρων.

Αν το αποτέλεσμα από το ερώτημα είναι μια γραμμή τότε σημαίνει πως έχει επιστραφεί η σωστή τιμή και έτσι θέτουμε αρχικές τιμές στις μεταβλητές $_SESSION[‘reset_email’] και $_SESSION[‘reset_random’]. Σε περίπτωση που το αποτέλεσμα του ερωτήματος επιστρέψει κάτι άλλο εκτός από τη γραμμή που εμείς χρειαζόμαστε αμέσως ο χρήστης μεταφέρεται στη σελίδα εισόδου, μέσω της αρχικής σελίδας, και δεν είναι σε θέση να επαναφέρει τον κωδικό του.

```php
//resets password using input data from user
function resetPassword(){

    global $showform,$match;

    $con = dbConnect();

    if ($_POST['newpass']==$_POST['confirm'] && strlen($_POST['newpass'])>=6){
    //if passwords match and are at least 6 chars long
        $newpass = mysqli_real_escape_string($con,$_POST['newpass']);

        executeQuery($con,"UPDATE USERS SET PASS='$newpass' where EMAIL='".$_SESSION['reset_email']."'");

        $showform = false;
    }else{
        $match = false;
    }
}
```

Εικόνα 20: resetPassword() στο app_logic.php

Στη συνέχεια πραγματοποιείται ο έλεγχος για το αν οι δύο κωδικοί που εισήγαγε ο χρήστης στη φόρμα επαναφοράς είναι όμοιοι. Οι κωδικοί προκύπτουν από την φόρμα της εικόνας 14, και ελέγχονται από τις μεταβλητές $_POST['newpass'] και $_POST['confirm'] όπου newpass και confirm είναι τα ονόματα των textbox που χρησιμοποιούνται στη φόρμα. Αν οι κωδικοί εν τέλει είναι όμοιοι και μεγαλύτεροι από 6 χαρακτήρες, τότε γίνεται update του πινάκα USERS με το καινούριο password για το συγκεκριμένο χρήστη. Θέτουμε την μεταβλητή $showform = false για να τη χρησιμοποιήσουμε παρακάτω. Τέλος αν οι δύο κωδικοί που εισήγαγε ο χρήστης δεν ταιριάζουν απόλυτα τότε θέτουμε τη μεταβλητή $match = false για να εμφανίσουμε μήνυμα λάθους κάτω από την φόρμα εισαγωγής.

```php
<?php
    //if passwords do not match
    if (isset($showform) && $showform){
        echo
        '
        <div id="large_div"><div id="df"><form id="input_form" action="reset.php" method="post">
        <input type="password" name="newpass" placeholder="New Password"></br></br>
        <input type="password" name="confirm" placeholder="Confirm Password"></br></br>
        <input id="sub" type="submit" value="Reset">
        </form></div></div>';

        if (isset($match) && $match == false){
            echo '<p id="wrong_input">The passwords you entered do not match!</p></br>';
        }
    }else if (isset($showform) && $showform == false){
        echo '<center><p id="success_msg">Password changed!</p></center><br/>';
        echo '<center><a id="login" href="/login.php">Login</a></center>';
    }
?>
```

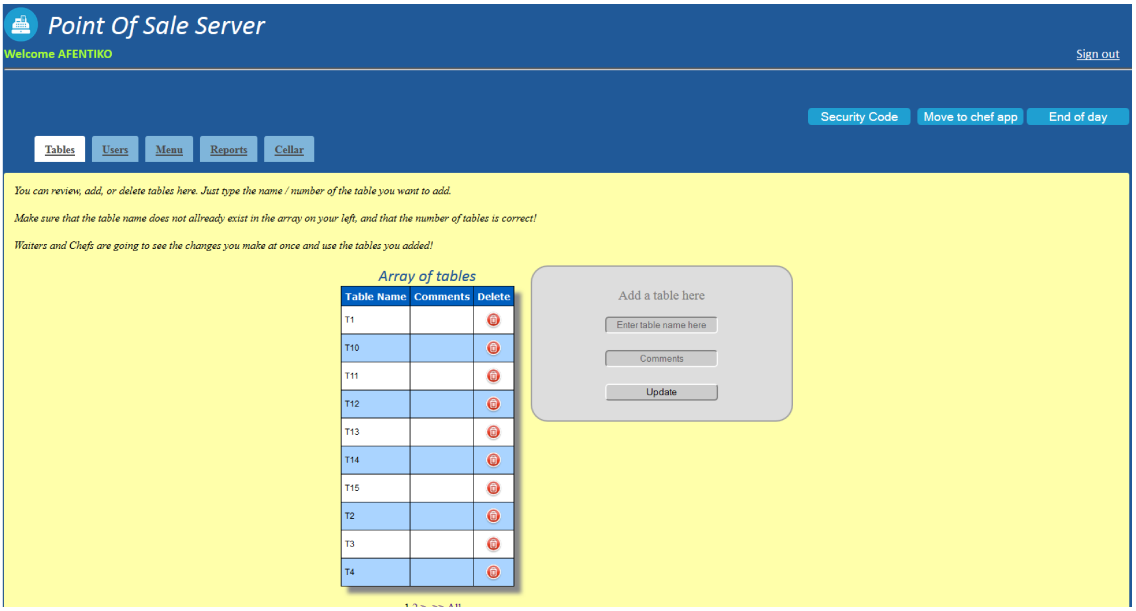
Εικόνα 21: Reset Form και μήνυμα Password Changed!

Εάν οι κωδικοί δεν ταιριάζουν μεταξύ τους εμφανίζουμε τη φόρμα με ένα μήνυμα λάθους όπως φαίνεται στην εικόνα 14. Αν οι κωδικοί ταιριάζουν,

εμφανίζουμε ένα πράσινο κείμενο (Εικόνα 15) και χωρίς τη φόρμα, με το μήνυμα ‘’Password Changed!’’ και το κουμπί Login το οποίο θα μας μεταφέρει στην αρχική σελίδα για εισαγωγή Username και Password.

3.6. MANAGE AND REVIEW (manage.php)

Μετά από το επιτυχημένο login ενός χρήστη με δικαιώματα administrator, εμφανίζεται η παρακάτω οθόνη, από το αρχείο manage.php :



Εικόνα 22: Αρχική οθόνη του manage.php

Σε αυτή την οθόνη αυτό που πρέπει να παρατηρήσουμε είναι έξι σημαντικά στοιχεία.

1. Μήνυμα καλωσορίσματος
2. Tabbed Menu
3. Sign out link
4. Δυνατότητα επεξεργασίας της βάσης δεδομένων
5. End of day button
6. Security Code button

Αρχικά το μήνυμα καλωσορίσματος προκύπτει από τα στοιχεία εισαγωγής της φόρμας στο login screen. Επιστρέφεται το όνομα του χρήστη από το email ή το username που εισήγαγε. Αυτό γίνεται με τη βοήθεια του παρακάτω κώδικα:

```php
<?php
    echo '<b id="welcome" style="color:#ADFF2F">Welcome '. $_SESSION['user'].'</b>';
?>
```

Εικόνα 23: Welcome User

Η μεταβλητή $_SESSION[‘user’] παίρνει τη τιμή της από τη μέθοδο loginUser() στο app_logic.php που φαίνεται στην εικόνα 8.

Με τη βοήθεια των tabs δημιουργούμε ένα μενού για τον Administrator απλό και κατανοητό το οποίο οργανώνει με αυτή την εμφάνιση θεματικά όλες τις βασικές λειτουργίες που χρειάζεται να έχει ένας Administrator του Point of Sale. Για το σκοπό αυτό χρησιμοποιούμε μια λίστα στον παρακάτω κώδικα σε html και php σε συνδυασμό με το αρχείο tabs.css το οποίο είναι υπεύθυνο για την εμφάνιση της λίστας με μορφή tabs.

```
<ul class="tab-links">
    <li <?php if (isset($_GET['tab']) && $_GET['tab'] == '1' ) echo 'class="active" ';?>><a href="manage.php?tab=1">Tables</a></li>
    <li <?php if (isset($_GET['tab']) && $_GET['tab'] == '2' ) echo 'class="active" ';?>><a href="manage.php?tab=2">Users</a></li>
    <li <?php if (isset($_GET['tab']) && $_GET['tab'] == '3' ) echo 'class="active" ';?>><a href="manage.php?tab=3">Menu</a></li>
    <li <?php if (isset($_GET['tab']) && $_GET['tab'] == '4' ) echo 'class="active" ';?>><a href="manage.php?tab=4">Reports</a></li>
    <li <?php if (isset($_GET['tab']) && $_GET['tab'] == '5' ) echo 'class="active" ';?>><a href="manage.php?tab=5">Cellar</a></li>

</ul>
 <div class="tab-content">
    <div id="tab1" class="tab<?php if (isset($_GET['tab']) && $_GET['tab'] == '1' ) echo ' active';?>">
        <?php
            if (isset($_GET['tab']) && $_GET['tab'] == '1' ){
                //load tables tab content
                require_once "tables.php";
            }

        ?>
    </div>
```

Εικόνα 24: Tabbed menu στο manage.php

Στον παραπάνω κώδικα βλέπουμε πως δημιουργούνται τα tabs με τη βοήθεια των tags ul και li που δημιουργούν στοιχεία λίστας με ονόματα Tables, Users, Menu, Reports και Cellar που θα εξηγήσουμε αναλυτικά παρακάτω τι περιλαμβάνει το καθένα και τι γίνεται με τον κώδικα που βρίσκεται από πίσω. Σε αυτό το σημείο παρατηρούμε και τρόπο με τον οποίο ενεργοποιούμε και απενεργοποιούμε ένα tab θέτοντας class=”active” με τη βοήθεια της μεταβλητής $_GET[‘tab’].

Στο πρώτο div που συναντάμε βλέπουμε ότι γίνεται ο έλεγχος για το αν το tab 1 είναι active και αν ισχύει αυτό τότε καλούμε το αρχείο tables.php το οποίο είναι και η πρώτη οθόνη του Administrator, που θα αναλύσουμε εκτενέστερα στο επόμενο κεφάλαιο.

Αφού έχει πραγματοποιηθεί το login και βρισκόμαστε στην πρώτη οθόνη, με τη μεταβλητή $_SESSION δεν αφήνουμε το χρήστη να μετακινηθεί σε

καμία άλλη λειτουργία του προγράμματος όπως είναι το forgot.php ή το reset.php γιατί ουσιαστικά δεν χρειάζεται να το κάνει από τη στιγμή που έχει γίνει επιτυχημένα το login. Κάθε προσπάθεια να μεταφερθούμε σε κάποιο άλλο link της εφαρμογής θα μας φέρει εκ νέου στη σελίδα localhost/presentation/manage.php?tab=1. Πραγματοποιείται kill του session με το κουμπί Sign out που βρίσκεται στα δεξιά ή με το κλείσιμο του browser, ως προεπιλογή από την php.

```php
<?php session_start();

session_destroy();
header( 'Location: /presentation/login.php' ) ;

?>
```

Εικόνα 25: Logout και redirect στο logout.php

Τη αναλυτικότερη περιγραφή του κουμπιού End of day θα την εξηγήσουμε στο υποκεφάλαιο 3.5.4 όπου θα γίνει πλήρης περιγραφή της καρτέλας Reports, για την οποία έχει δημιουργηθεί ουσιαστικά το κουμπί αυτό. Βρίσκεται εκτός της καρτέλας Reports για το λόγο ότι είναι μια λειτουργία καθοριστικής σημασίας για το POS και πρέπει να μπορεί ο Administrator να έχει άμεση πρόσβαση στο κουμπί αυτό αφού πατώντας το όχι μονό μπορεί να κλείσει την ημέρα την οποία πέρασε αλλά ταυτόχρονα να ξεκινήσει και την επόμενη μέρα, σε επίπεδο βάσης δεδομένων πάντα.

3.6.1. ΚΑΡΤΕΛΑ TABLES (tables.php)

Τα βασικά σημεία στα οποία πρέπει να σταθούμε στο πρώτο tab διαχείρισης του POS είναι ότι αποτελείται από έναν πίνακα με τα τραπέζια του μαγαζιού τα οποία είναι καταχωρημένα στη βάση δεδομένων. Ο πίνακας αυτός γεμίζει ουσιαστικά με ένα ερώτημα προς τη βάση πάνω στον πίνακα tables και εμείς εδώ εμφανίζουμε την πληροφορία που μας ενδιαφέρει, στη συγκεκριμένη περίπτωση τα πεδία table_num και comments. Ο πίνακας αυτός όπως φαίνεται και στο διάγραμμα ER της εικόνας 3 περιλαμβάνει και το πεδίο T_ID που όμως είναι πληροφορία την οποία χρησιμοποιούμε σαν προγραμματιστές για τη διευκόλυνση μας ενώ η πληροφορία αυτή είναι εντελώς άχρηστη στον Administrator του συστήματος και γι’ αυτόν δεν είναι διαθέσιμη σε αυτόν. Ο κώδικας τον οποίο χρησιμοποιούμε για την δημιουργία και το γέμισμα του πίνακα αυτού φαίνεται παρακάτω:

```php
function getTablesData($page) {
    global $PAGING_LIMIT;

    // connect to database
    $con = dbConnect ();

    // set number of pages
    if ($page == -1) { // show all
        $_SESSION ['table_max'] = 1; // show one page only

        $result = executeQuery ( $con, "SELECT * FROM TABLES ORDER BY TABLE_NUM" );
    } else {

        $skip = $page * $PAGING_LIMIT; // how many records to skip
        $skip = mysqli_real_escape_string ( $con, $skip );

        // count rows
        $result = executeQuery ( $con, "SELECT count(*) as cnt FROM TABLES" );
        $row = mysqli_fetch_array ( $result );

        $_SESSION ['table_max'] = ceil ( $row ['cnt'] / ( double ) ($PAGING_LIMIT) );

        $result = executeQuery ( $con, "SELECT * FROM TABLES ORDER BY TABLE_NUM LIMIT $skip,$PAGING_LIMIT" );
    }

    return $result;
}
```

Εικόνα 26 : getTablesData() στο app_logic.php

Με μια σύνδεση στη βάση μας και ένα select το οποίο μας επιστρέφει όλο ή μέρος του πίνακα Tables, παίρνουμε σαν αποτέλεσμα το $result και το διαχειριζόμαστε στο tables.php. Τα χαρακτηριστικά του pagination Θα τα αναλύσουμε στο κεφάλαιο 3.5.3 όπου θα δοθεί και παράδειγμα για τη λειτουργία του.

```php
//tables.php was just included from manage.php
$result = getTablesData($currentPage);

echo '<div style="display: inline-block; vertical-align: top;">
    <i class="tables_desc">Array of tables</i>
 <table class="tables">
  <tr>

    <td>Table Name</td>
    <td>Comments</td>
    <td>Delete</td>
  </tr>
';

while($row = mysqli_fetch_array($result)) {
  echo '<tr><td>' . $row['TABLE_NUM'] . '</td><td>' . $row['COMMENTS'] . '</td><td>'. '<a href="tables.php?del='
  . $row['T_ID'] . '" ><center><img src="../img/delete.png" alt="Delete" style="width:30px;height:30px"></center></a> </td></tr>';
}

echo '</table>';

showPagingControls($currentPage,$_SESSION['table_max'],'./manage.php?tab=1&table_pg');
```

Εικόνα 27: Δημιουργία πίνακα και γέμισμα με tables

Μέσα στο php block του tables.php δημιουργούμε ένα πίνακα με τρία πεδία, τα Table name, Comments και Delete. Όλα τα πεδία του πίνακα αυτού

γεμίζουν με ένα while, το οποίο βάζει στο πεδίο Table name του πίνακα όλα τα πεδία του πίνακα Tables της βάσης που αντιστοιχούν στο όνομα του τραπεζιού, και στο πεδίο Comments τα περιεχόμενα του πεδίου που αντιστοιχούν με σχόλια στη βάση.

Τέλος το πεδίο Delete του πίνακα μας γεμίζει με μια εικόνα της επιλογής μας που απεικονίζει έναν μικρό κόκκινο κάδο ανακύκλωσης και ουσιαστικά είναι link για τη μέθοδο deleteTable() η οποία βρίσκεται στο app_logic και σβήνει την εγγραφή του πίνακα χρησιμοποιώντας το t_id της εγγραφής.

```php
//delete one table from the database
function deleteTable(){
    //connect to database
    $con = dbConnect();

    $tableid = mysqli_real_escape_string($con,$_GET['del']);
    executeQuery($con,"DELETE FROM TABLES WHERE T_ID = '$tableid'");

    //redirect
    header("Location: /presentation/manage.php?tab=1");
}
```

Εικόνα 28: deleteTables() στο app_logic.php

Το όρισμά της η εντολή DELETE που απευθύνεται στη posdb, το παίρνει από το url με την $_GET[‘del’] η οποία διαμορφώνεται πατώντας στην εικόνα με τον κόκκινο κάδο. Αμέσως μετά τη διαγραφή μια εγγραφής του πίνακα γίνεται αυτόματα ένα redirect στο ίδιο tab του αρχείου manage.php για να γίνει και η ανανέωση του πίνακα που εμφανίζεται μπροστά μας.

Στη φόρμα που βρίσκεται στα δεξιά του πίνακα με τα τραπέζια μπορούμε να εισάγουμε καινούρια τραπέζια στη βάση μας και να δούμε τον πίνακα που υπάρχει στα αριστερά μας να συμπληρώνεται αυτόματα. Προϋπόθεση σημαντική, όπως αναφέρει και το κείμενο πάνω από τη φόρμα είναι να μην εισάγουμε τραπέζι με κενό όνομα γιατί τότε θα πάρουμε το αντίστοιχο μήνυμα σφάλματος, και επίσης να μην εισάγουμε τραπέζι με όνομα το οποίο ήδη υπάρχει. Έχει επιλεγεί κατά τη διάρκεια του σχεδιασμού της βάσης δεδομένων το πεδίο table_num της βάσης να είναι τύπου και varchar και όχι int αν και αναφέρεται σε νούμερο τραπεζιού για το λόγο ότι σε πολλά μαγαζιά τα νούμερα των τραπεζιών συμπληρώνονται από γράμματα τα οποία βοηθούν στην οργάνωση του μαγαζιού και στην λειτουργικότητά του, όπως πχ αν κάποιος έχει 51 τραπέζια σε ένα τετράγωνο χώρο πλάτους 3 τραπεζιών,

μπορεί να τα χωρίσει σε 3 σειρές Α,Β,Γ και να αριθμήσει τη κάθε σειρά με 17 νούμερα, Α1…Α17, Β1…Β17, Γ1…Γ17. Έτσι μπορεί ο χρήστης να εισάγει τραπέζι με οποιοδήποτε όνομα από το πληκτρολόγιο του με δική του ευθύνη για το αν το όνομα του τραπεζιού θα έχει σχέση με τα τραπέζια του υπόλοιπου πίνακα ή όχι. Παρακάτω θα δούμε τον κώδικα που χρησιμοποιείται για τη δημιουργία τραπεζιού και τους ελέγχους που πραγματοποιούνται.

```php
//adds a new table to the database
function addNewTable(){
    //connect to database
    $con = dbConnect();
    $tablenum = mysqli_real_escape_string($con,$_POST['table_num']);
    $comments = mysqli_real_escape_string($con,$_POST['comments']);
    $result = executeQuery($con,"SELECT TABLE_NUM FROM TABLES WHERE TABLE_NUM='$tablenum'");
    $numrows = mysqli_num_rows($result);

    if ($numrows == 1){
        $_SESSION['exists'] = true;
    }
    elseif ($_POST['table_num'] != '' && !isset($_SESSION['exists'])){
        executeQuery($con,"INSERT INTO TABLES VALUES (null, '$tablenum','$comments')");
    }else{
        $_SESSION['tables_form_error'] = true;
    }
    //redirect
    header("Location: /presentation/manage.php?tab=1");
}
```

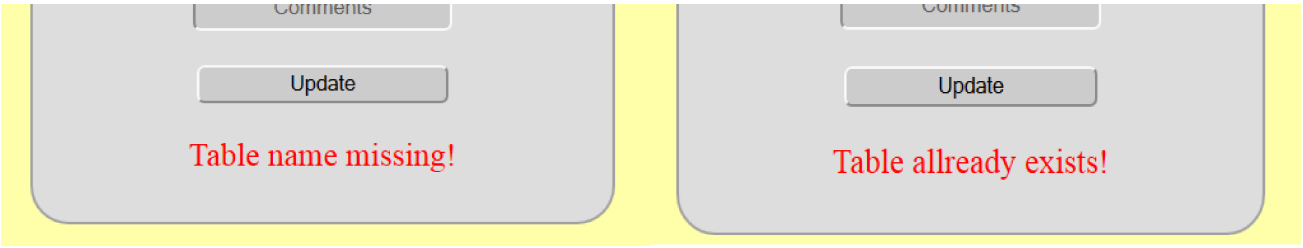
Εικόνα 29: addNewTable() στο app_logic.php

Πραγματοποιείται σύνδεση στη βάση δεδομένων αρχικά, όπως και στις περισσότερες μεθόδους, αποθηκεύονται σε τοπικές μεταβλητές τα δεδομένα που έχει εισάγει ο χρήστης από τις $_POST[‘table_num’] και $_POST[‘comments’] και εκτελείται ερώτημα προς τη βάση για το αν το όνομα που εισήγαγε ο χρήστης υπάρχει ή όχι. Αν η select επιστρέψει μια γραμμή τότε υπάρχει το όνομα του τραπεζιού στη βάση και έτσι θέτουμε τη $_SESSION[‘exists’]=true για να τη χρησιμοποιήσουμε παρακάτω στο tables.php για την εμφάνιση του αντίστοιχου μηνύματος. Αν η select δεν επιστρέψει αποτέλεσμα τότε σημαίνει πως δεν υπάρχει στη βάση καμία εγγραφή με το ίδιο table_num άρα μπορούμε να ελέγξουμε ότι το $_POST[‘table_num’] δεν είναι ίσο με το κενό και να προχωρήσουμε στο update του πίνακα Tables. Αν ο χρήστης δεν έχει εισάγει ούτε το όνομα του τραπεζιού τότε θα πάρει άλλο μήνυμα το οποίο προκύπτει από την αρχικοποίηση του $_SESSION[‘tables_form_error’]=true.

```
echo '
<div id="df" style="display: inline-block"><form id="input_form" action="tables.php" method="post">
<p>Add table here</p>
<input type="text" name="table_num" placeholder="Enter table name here"><br/><br/>
<input type="text" name="comments" placeholder="Comments"><br/><br/>
<input id="sub" type="submit" value="Update">';

if (isset($_SESSION['tables_form_error'])){
    echo '<br/><p id="wrong_input">Table name missing!</p></form></div>';
    unset($_SESSION['tables_form_error']);
}
elseif(isset($_SESSION['exists'])){
    echo '<br/><p id="wrong_input">Table allready exists!</p></form></div>';
    unset($_SESSION['exists']);
}
else{
    echo '</form></div>';
}
```

Εικόνα 30: Φόρμα εισαγωγής και μηνύματα λάθους στο tables.php

Table is missing! είναι το μήνυμα λάθους που εμφανίζεται αν το τραπέζι που εισήγαγε ο χρήστης είναι κενό και Table already exists! αν το όνομα του τραπεζιού υπάρχει ήδη στην posdb, και φαίνονται και τα δύο στην εικόνα 31 παρακάτω.



Εικόνα 31: Μηνύματα λαθεμένης εισαγωγής στο tables.php

Να τονίσουμε ότι παίζει ρόλο τα ονόματα των τραπεζιών στη βάση μας να εισάγονται με χαρακτήρες της ίδιας γλώσσας. Αν δούμε διπλότυπο τραπέζι στη βάση μας σημαίνει πως έχουμε εισάγει χαρακτήρες του ενός ή του άλλου τραπεζιού σε διαφορετική γλώσσα πχ T15 με αγγλικό ‘T’ και T15 με ελληνικό ‘T’. Στην περίπτωση αυτή όχι μόνο δεν θα λάβουμε μήνυμα σφάλματος εισαγωγής αλλά θα εισάγουμε κανονικά και νόμιμα το τραπέζι στη βάση και θα το δούμε μάλιστα να υπάρχει δύο φορές. Αυτό μπορεί να προκαλέσει τη σύγχυση του σερβιτόρου στο POS client που θα έχει μπροστά του όμως δεν θα υπάρξει καμία επιπλοκή απολύτως στη λειτουργία του προγράμματος.

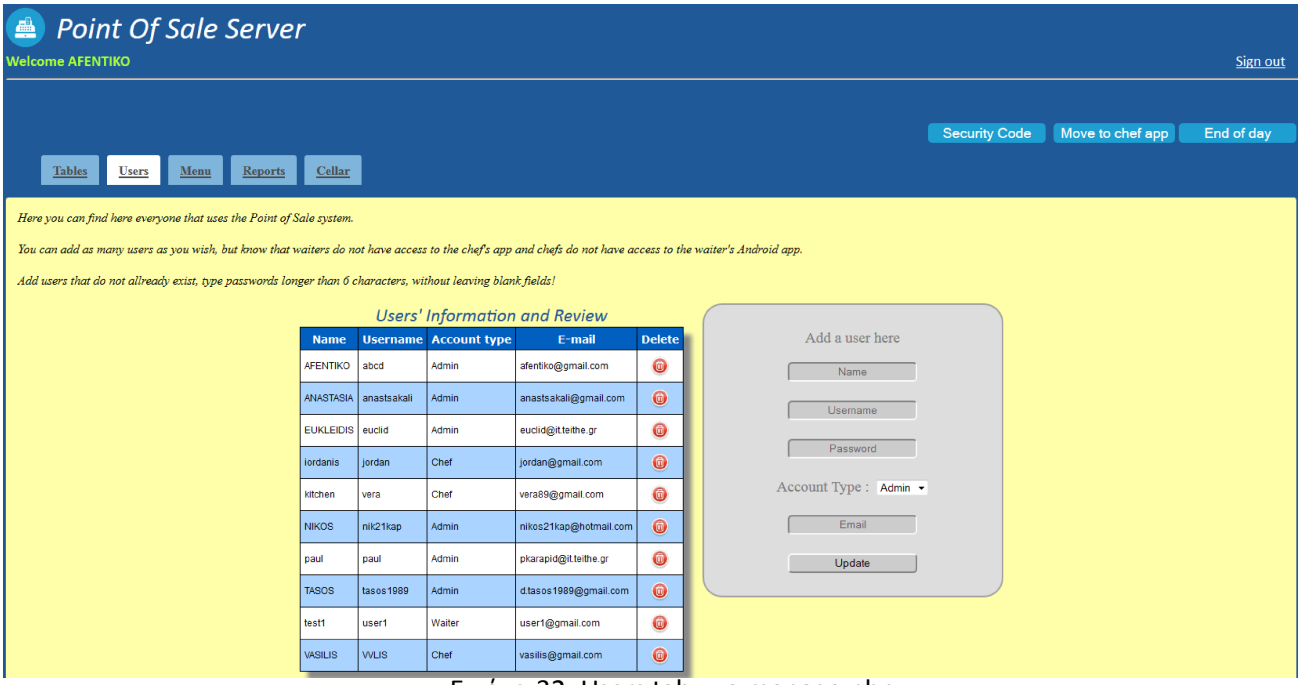
3.6.2. ΚΑΡΤΕΛΑ USERS (users.php)

Με την ίδια ακριβώς λογική αλλά με λίγο διαφορετική υλοποίηση, την οποία θα δείξουμε παρακάτω, η καρτέλα Users δίνει τη δυνατότητα στον

Administrator να διαχειριστεί τους διάφορους χρήστες του POS. Αυτό που παρατηρούμε σε αυτήν εδώ τη καρτέλα είναι 2 πράγματα.

Τα passwords των χρηστών δεν είναι διαθέσιμα στον administrator.

Ο administrator μπορεί να εισάγει μόνο 3 είδη χρηστών.



Εικόνα 32: Users tab στο manage.php

Ο τρόπος με τον οποίο εμφανίζεται το users.php από το manage.php είναι ο αντίστοιχος της εικόνας 24, δηλαδή με include του αρχείου users.php στο tab 2. Βλέπουμε στα αριστερά μας τον πίνακα users από τη βάση μας αλλά επιστρέφουμε μόνο τα πεδία που είναι σημαντικά για τον διαχειριστή. Για παράδειγμα το u_id δεν είναι διαθέσιμη πληροφορία στον διαχειριστή γιατί είναι κάτι το οποίο χρησιμοποιούμε εσωτερικά στην phpMyAdmin για δική μας διευκόλυνση σε ερωτήματα select, update κ.α.

```
function getUsersData(){
    //connect to database
    $con = dbConnect();

    $result = executeQuery($con,"SELECT * FROM USERS ORDER BY NAME");

    return $result;
}
```

Εικόνα 33: getUsersData() στο app_logic.php

Αρχικά πραγματοποιείται σύνδεση με τη βάση μας και ένα select μας επιστρέφει όλο τον πίνακα Users, τον οποίο παίρνουμε σαν αποτέλεσμα από

τη μεταβλητή $result και τη διαχειριζόμαστε στο users.php

```
echo '

 <table id="user_array">
  <tr>

    <th>Name</th>
    <th>Username</th>
    <th>Account type</th>
    <th>E-mail</th>
    <th>Delete</th>
  </tr>
';

while($row = mysqli_fetch_array($result)) {
  echo '<tr><td>' . $row['NAME'] . '</td><td>' . $row['USERNAME'].'</td><td>'. $row['ACC_TYPE'].
  '</td><td>'.$row['EMAIL'].'</td><td>'.'<a href="users.php?del=' . $row['U_ID'] .
  '" ><img id="del_icon" src="../img/delete.png" alt="Delete" style="width:30px;height:30px"></a> </td></tr>';
}

echo '</table>';
```

Εικόνα 34: Δημιουργία πίνακα και γέμισμα με users

Ο πινάκας αποτελείται όπως φαίνεται στον παραπάνω κώδικα από τα πεδία Name, Username, Account type, Email και Delete. Το γέμισμα του πίνακα αυτού γίνεται επίσης με ένα while, όπως γινόταν και στο προηγούμενο tab για τα τραπέζια αντιστοιχίζονται τις τιμές της βάσης στα σωστά πεδία του πίνακα μας. Το πεδίο Name είναι το όνομα του χρήστη, το username και το email είναι προσωπικά στοιχεία για την είσοδο του χρήστη στο πρόγραμμα αλλά και για την επαναφορά του κωδικού του χρήστη και το Account type είναι ένα από τα σημαντικότερα πεδία αυτής της βάσης δεδομένων. Είναι το στοιχείο αυτό που καθορίζει ουσιαστικά το είδος στο οποίο ανήκει ο κάθε χρήστης, και κατ' επέκταση τα δικαιώματα του. Αν για παράδειγμα ένας χρήστης είναι τύπου Admin μπορεί να έχει πρόσβαση σε όλα τα επίπεδα της εφαρμογής, δηλαδή στο chef.php όπου παρακάτω θα αναλύσουμε και είναι η εφαρμογή για την κουζίνα, μέσω του κουμπιού που "Move to chef app" που αναφέραμε στο προηγούμενο κεφάλαιο, αλλά και στο πρόγραμμα client του Android. Τέλος το κουμπάκι με τον κόκκινο κάδο ανακύκλωσης είναι μια εικόνα με ένα link διαγραφής του χρήστη από τη βάση δεδομένων, όπως και στο προηγούμενο tab, με τον κώδικα που το υλοποιεί να φαίνεται στην εικόνα 35.

```php
//delete one user from database
function deleteUser(){
    //connect to database
    $con = dbConnect();

    $userid = mysqli_real_escape_string($con,$_GET['del']);
    $result = executeQuery($con,"DELETE FROM USERS WHERE U_ID = '$userid'");

    //redirect
    header("Location: /presentation/manage.php?tab=2");
}
```

Εικόνα 35: deleteUser() στο app_logic.php

Αρχικά αποθηκεύουμε στη μεταβλητή $userid τη μεταβλητή $_GET['del'] της οποία η τιμή διαμορφώνεται πατώντας το κόκκινο κουμπάκι του κάδου στον πίνακα και περνάει στην εντολή delete για να διαγράψει τον χρήστη με το συγκεκριμένο u_id. Τέλος γίνεται redirect στο ίδιο tab το οποίο βρισκόμαστε για να δούμε ότι η αλλαγή μας πραγματοποιήθηκε.

Στη φόρμα που υπάρχει στα δεξιά μας μπορεί ο Administrator του προγράμματος να προσθέσει χρήστες και να δει τους χρήστες να συμπληρώνονται αυτόματα στον πίνακα αριστερά. Έχουμε υλοποιήσει τρεις βασικές προϋποθέσεις οι οποίες φαίνονται και πάνω από τον πίνακα στο κείμενο για την ορθή λειτουργία της φόρμας κατά την εισαγωγή χρηστών στη βάση. Η πρώτη προϋπόθεση είναι τα textbox που υπάρχουν στη φόρμα να είναι όλα συμπληρωμένα. Ακόμα και ένα κενό textbox αρκεί για να πάρουμε μήνυμα λάθους. Η δεύτερη προϋπόθεση είναι ο χρήστης τον οποίο επιθυμούμε να εισάγουμε να μην υπάρχει ήδη στη βάση δηλαδή να μην ταυτίζονται το username ή το email με κάποιο υπάρχον. Τελευταία και επίσης σημαντική προϋπόθεση είναι ο κωδικός του χρήστη να είναι ίσος ή μεγαλύτερος των έξι χαρακτήρων. Το μέτρο αυτό είναι ένα μικρό μέτρο ασφάλειας το οποίο χρησιμοποιούμε. Εδώ να σημειώσουμε ότι οι τρεις επιλογές τύπου χρήστη, admin, waiter, chef έχουν μπει hardcoded στο πρόγραμμα και είναι το μοναδικό σημείο που γίνεται αυτό σε ολόκληρη την υλοποίηση της πτυχιακής μας εργασίας, πράγμα που δεν συμβαίνει στο drop down select της φόρμας που βρίσκεται στο menu.php που θα αναλύσουμε στο επόμενο κεφάλαιο. Ο λόγος για τον οποίο δεν υπάρχει ξεχωριστός πίνακας στη βάση μας για τα account types είναι διότι στα πλαίσια της συγκεκριμένη πτυχιακής οι τύποι των χρηστών οι οποίοι θα εξετάσουμε είναι τρεις και μόνο. Γι' αυτό προτιμήσαμε να το κάνουμε αυτό στο συγκεκριμένο σημείο. Αν ο διαχειριστής του προγράμματος επιλέξει κάποια από τις επιλογές

admin, waiter ή chef τότε στη βάση μας θα περαστεί η κατάλληλη τιμή τύπου tinyint όπως θα δούμε παρακάτω.

```php
//adds a new user to the database
function addNewUser(){

    //connect to database
    $con = dbConnect();

    $name = mysqli_real_escape_string($con,$_POST['name']);
    $username = mysqli_real_escape_string($con,$_POST['username']);
    $pass= mysqli_real_escape_string($con,$_POST['pass']);
    $acc_type='';
    $email= mysqli_real_escape_string($con,$_POST['email']);
    $result = executeQuery($con,"SELECT USERNAME, EMAIL FROM USERS WHERE USERNAME='$username'");
    $result2 = executeQuery($con,"SELECT USERNAME, EMAIL FROM USERS WHERE EMAIL='$email'");
    $numrows = mysqli_num_rows($result);
    $numrows2 = mysqli_num_rows($result2);

    if ($_POST['usertype'] == 'waitor'){
        $acc_type = '0';
    }elseif($_POST['usertype'] == 'admin'){
        $acc_type = '1';
    }else {
        $acc_type = '2';
    }

    if ($numrows == 1 || $numrows2==1){
        $_SESSION['userexists'] = true;
    }
    elseif($_POST['name']=='' || $_POST['username']=='' || $_POST['pass']=='' || $_POST['email']==''){
        $_SESSION['missing'] = true;
    }
    elseif(strlen($_POST['pass'])<6){
        $_SESSION['shortpass']=true;
    }
    else{
        $result = executeQuery($con,"INSERT INTO USERS (U_ID,NAME,USERNAME,PASS,ACC_TYPE,EMAIL) VALUES
        (null, '$name','$username','$pass','$acc_type','$email')");
        $_SESSION['userok']=true;}

    //redirect
    header("Location: /presentation/manage.php?tab=2");
}
```

Εικόνα 36: addNewUser() στο app_logic

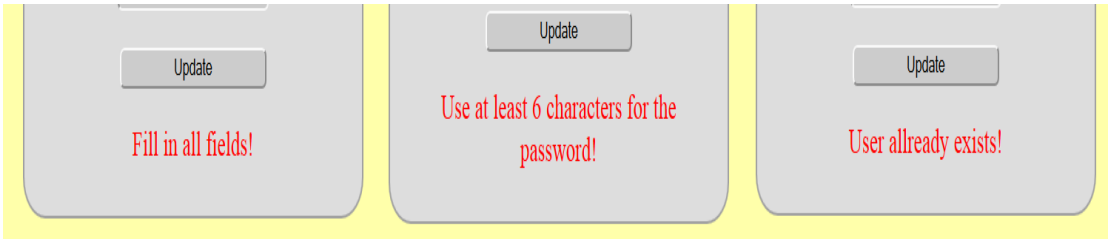
Πραγματοποιείται σύνδεση στη βάση δεδομένων αρχικά, όπως και στις περισσότερες μεθόδους, αποθηκεύονται σε τοπικές μεταβλητές τα δεδομένα που έχει εισάγει ο χρήστης από τις $_POST[‘name’], $_POST[‘username’], $_POST[‘pass’], $_POST[‘email’] και αρχικοποιείται η μεταβλητή $acc_type για να πάρει τιμή αν ο χρήστης έχει επιλέξει waiter τη τιμή 0, αν έχει επιλέξει admin τη τιμή 1 και αν έχει επιλέξει chef τη τιμή 2 με την $_POST[‘usertype’]. Στις μεταβλητές $result και $result2 αποθηκεύουμε το αποτέλεσμα των select προς τη βάση και στη συνέχεια γίνονται οι απαραίτητοι έλεγχοι. Αν ο χρήστης υπάρχει στη βάση τότε η τιμή της $_SESSION[‘userexists’] = true. Αν τα πεδία της φόρμας είναι κενά τότε θέτουμε τη μεταβλητή $_SESSION[‘missing’] =

true. Αν οι χαρακτήρες που εισήγαγε ο χρήστης για κωδικό πρόσβασης είναι λιγότεροι από έξι τότε θέτουμε την $_SESSION[‘shorpass’] = true. Αυτές τις μεταβλητές τις χρησιμοποιούμε στο users.php παρακάτω για την εμφάνιση των κατάλληλων μηνυμάτων λάθους. Τέλος αν καμία από τις παραπάνω συνθήκες δεν αληθεύει τότε εκτελούμε κανονικά το insert στη βάση μας. Το redirect το οποίο φαίνεται στην εικόνα πραγματοποιείται σε οποιαδήποτε περίπτωση αποτελέσματος των ελέγχων.

```
echo '
<div id="df" class="add_user"><form id="input_form" action="users.php" method="post">
<p>Add a user here</p>
<input type="text" name="name" placeholder="Name"></br></br>
<input type="text" name="username" placeholder="Username"></br></br>
<input type="text" name="pass" placeholder="Password"></br></br>
Account Type :  <select name="usertype">

    //populate form with usertypes
    <option value="admin">Admin</option>
    <option value="waitor">Waitor</option>
    <option value="chef">Chef</option>

</select></br></br>
<input type="text" name="email" placeholder="Email"></br></br>
<input id="sub" type="submit" value="Update">';

if (isset($_SESSION['missing'])){
    echo '<br/><p id="wrong_input">Fill in all fields!</p></form></div>';
    unset($_SESSION['missing']);
}
elseif(isset($_SESSION['userexists'])){
    echo '<br/><p id="wrong_input">User allready exists!</p></form></div>';
    unset($_SESSION['userexists']);
}
elseif(isset($_SESSION['shortpass'])){
    echo '<br/><p id="wrong_input">Use at least 6 characters for the password!</p></form></div>';
    unset($_SESSION['shortpass']);
}
else{
    echo '</form></div>';
}
```

Εικόνα 37: Φόρμα προσθήκης νέων χρηστών στο users.php

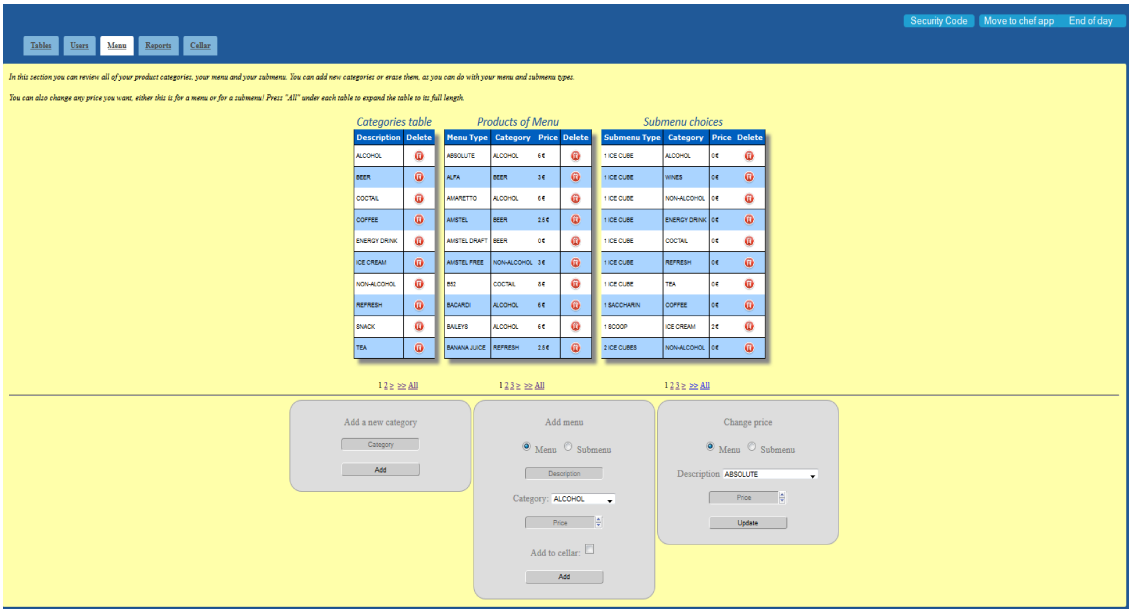
Fill in all fields! είναι το μήνυμα το οποίο εμφανίζεται στη φόρμα αν κάποιο πεδίο δεν έχει συμπληρωθεί από τον administrator, User already exists! είναι το μήνυμα σε περίπτωση που ο χρήστης υπάρχει ήδη και Use at least 6 characters for the password! είναι το μήνυμα αν ο κωδικός είναι μικρότερος από έξι χαρακτήρες.



Εικόνα 38: Μηνύματα λάθους

Να σημειωθεί ότι δεν παίζουν ρόλο οι κεφαλαίοι και οι μικροί χαρακτήρες για την εισαγωγή νέων χρηστών στη βάση. Αν κάποιος χρήστης υπάρχει στη βάση τότε ο έλεγχος που κάνουμε δεν θα μας αφήσει να εισάγουμε το νέο χρήστη ακόμα και αν τον έχουμε γράψει με κεφαλαία γράμματα.

3.6.3. ΚΑΡΤΕΛΑ MENU (menu.php)



Εικόνα 39: Menu Tab στο manage.php

Στο τρίτο tab για το πρόγραμμα του διαχειριστή παρατηρούμε ότι περιλαμβάνονται οι τρείς πίνακες μέσα στους οποίους περιέχονται οι κατηγορίες προϊόντων, οι τύποι μενού για την κάθε κατηγορία και το υπομενού.

Οι πίνακες αυτοί γεμίζουν με τρία ερωτήματα προς τη βάση δεδομένων για τους πίνακες categories, menu, sub_menu. Από τα ερωτήματα αυτά επιλέγουμε τα στοιχεία που εμφανίζονται στον administrator και αυτά είναι τα ονόματα των κατηγοριών και των μενού, την κατηγορία που ανήκει το κάθε μενού ή υπομενού και τέλος την αντίστοιχη τιμή κάθε προϊόντος. Τα id του κάθε πεδίου δεν είναι διαθέσιμα στο διαχειριστή γιατί η πληροφορία αυτή αφορά τους προγραμματιστές του POS για την καλύτερη οργάνωση και πρόσβαση στη βάση δεδομένων. Παρατηρούμε επίσης ότι κάτω από όλους τους πίνακες χρησιμοποιείται λειτουργία pagination. Αυτό συμβαίνει γιατί οι συγκεκριμένοι πίνακες στους οποίους έχει πρόσβαση ο administrator είναι μεγάλου μεγέθους και η εμφάνιση τους από το browser σ' ένα παράθυρο μόνο είναι άσχημη εμφανισιακά. Ο κώδικας τον οποίο χρησιμοποιούμε για την

δημιουργία και το γέμισμα αυτών των πινάκων φαίνεται στις εικόνες παρακάτω:

```php
//takes as input the page of the data to be returned
function getCategoriesData($page){
    global $PAGING_LIMIT;

    //connect to database
    $con = dbConnect();

    if ($page == -1){//show all
        //set number of pages
        $_SESSION['menu_pg1_max'] = 1;

        $result = executeQuery($con,"SELECT * FROM CATEGORIES ORDER BY CAT_DESC");
    }else{
        $skip = $page * $PAGING_LIMIT;//how many records to skip
        $skip = mysqli_real_escape_string($con,$skip);

        //count rows
        $result = executeQuery($con,"SELECT count(*) as cnt FROM CATEGORIES");
        $row = mysqli_fetch_array($result);

        //set number of pages
        $_SESSION['menu_pg1_max'] = ceil($row['cnt'] / (double)($PAGING_LIMIT));

        $result = executeQuery($con,"SELECT * FROM CATEGORIES
                                     ORDER BY CAT_DESC LIMIT $skip,$PAGING_LIMIT");
    }

    return $result;
}
```

Εικόνα 40: getCategoriesData() στο app_logic.php

Η λογική για την εμφάνιση ενός πίνακα με pagination είναι ότι εμφανίζουμε κάθε φορά τα δεδομένα που χρειαζόμαστε κάνοντας το κατάλληλο request στη βάση δεδομένων, χρησιμοποιώντας την εντολή LIMIT. Αν για παράδειγμα θέλουμε ο πίνακας μας να έχει maximum 10 εγγραφές προς εμφάνιση, τότε κάθε φορά που πατάμε κάποιο κουμπί για να αλλάξουμε 10άδα παίρνουμε από τη βάση τις απαραίτητες εκείνες 10 εγγραφές που χρειαζόμαστε. Αν πάλι ο πίνακας μας έχει λιγότερες από 10 εγγραφές συνολικά τότε η λειτουργία Paging χάνεται μιας και δεν υπάρχουν επόμενες ή προηγούμενες σελίδες. Το παράδειγμα αυτό φαίνεται στην εικόνα 39 παραπάνω όπου ο πίνακας categories δεν έχει controls για αλλαγή σελίδας γιατί περιλαμβάνει μόνο 10 εγγραφές τη συγκεκριμένη χρονική στιγμή. Επίσης μπορεί ο χρήστης αν το επιλέξει να δει όλες τις εγγραφές του πίνακα πατώντας το All και να επιστρέψει πίσω πάλι στη λειτουργία pagination πατώντας το Pages που εμφανίζεται πλέον κάτω από τον ολόκληρο πίνακα.

```php
//show categories
$cat_result = getCategoriesData($_SESSION['menu_pg1']);

echo '<div style="display: inline-block; vertical-align: top;">

 <table class="tables" style="vertical-align: top">
  <tr>

    <td>Description</td>
    <td>Delete</td>
  </tr>
';

while($row = mysqli_fetch_array($cat_result)) {
  echo '<tr><td>' . $row['CAT_DESC'] . '</td><td><a href="menu.php?delcat=' . $row['CAT_ID'] .
  '" ><img id="del_icon" src="../img/delete.png" alt="Delete" style="widtd:30px;height:30px"></a> </td></tr>';
}

echo '</table></br>';

showPagingControls($_SESSION['menu_pg1'],$_SESSION['menu_pg1_max'],'./manage.php?tab=3&menu_pg1');

echo '</div>';
```

Εικόνα 41: Γέμισμα πίνακα με categories και εμφάνιση paging controls

Το πόσες σελίδες θα έχει ο κάθε πίνακας προκύπτει με ένα select count στον πίνακα που θέλουμε να επιστρέψουμε το οποίο διαιρείται με τη μεταβλητή $PAGING_LIMIT. Αυτή η μεταβλητή φαίνεται στην εικόνα 3 στο κάτω μέρος του αρχείου common.php και είναι μια global μεταβλητή. Ορίσαμε ως όριο εγγραφών το 10 γιατί εμφανίζονται ομαλά 10 εγγραφές στην οθόνη χωρίς να κουράζουν και δεν είναι ένας αριθμός πολύ μικρός.

Καλούμε την getCategoriesData() με όρισμα την σελίδα την οποία θέλουμε να μας επιστρέψει. Η σελίδα αυτή προκύπτει από τη μέθοδο showPagingControls() που θα αναλύσουμε παρακάτω. Γεμίζουμε τον πίνακα με τα αποτελέσματα της getCategoriesData() και τέλος εμφανίζουμε τα controls για το pagination.

Λόγω του μεγάλου μεγέθους της συγκεκριμένης μεθόδου θα αναλύσουμε τη λειτουργία των κουμπιών όταν βρισκόμαστε στην πρώτη σελίδα του πίνακα και στην τελευταία.

Οι παράμετροι της μεθόδου αυτής είναι η σελίδα που βρισκόμαστε αυτή τη στιγμή για να χρησιμοποιηθεί σαν δείκτης για την επόμενη ή την προηγούμενη σελίδα, το μέγιστο μέγεθος σελίδων για να γνωρίζει η μέθοδος ποια θα είναι η μέγιστη σελίδα την οποία θα εμφανίσει και τέλος η μεταβλητή $anchorString χρησιμοποιείται σαν link για τα κουμπάκια του pagination.[16]

```php
function showPagingControls($currentPage,$maxPages,$anchorString){

    $lastPage = $maxPages - 1;

    if ($maxPages > 1){//show controls

        if ($currentPage == 0){//(first page) 1 2 3 > >>
            $currentPage++;//normalize current page to be shown to user

            echo "1 <a href=\"$anchorString=1\">2</a> ";

            if ($maxPages >= 3){
                echo "<a href=\"$anchorString=2\">3</a> ";
            }
            echo "<a href=\"$anchorString=1\">&gt;</a> 
                <a href=\"$anchorString=$lastPage\">&gt;&gt;</a> ";

        }else if ($currentPage == $lastPage){//(last page) << < preprev prev cur

            $currentPage++;//normalize current page to be shown to user
            $previousPage = $lastPage - 1;
            $prePreviousPage = $previousPage - 1;
            $previousPageNorm = $previousPage + 1;
            $prePreviousPageNorm = $prePreviousPage + 1;

            echo "<a href=\"$anchorString=0\">&lt;&lt;</a> ";
            echo "<a href=\"$anchorString=$previousPage\">&lt;</a> ";

            if ($maxPages >=3){
                echo "<a href=\"$anchorString=$prePreviousPage\">$prePreviousPageNorm</a> ";
            }
            echo "<a href=\"$anchorString=$previousPage\">$previousPageNorm</a> ";
            echo $currentPage . ' ';
```

Εικόνα 42: showPagingControls() στο app_logic

Αρχικά αν βρισκόμαστε στην πρώτη σελίδα του πίνακα μας, θέλουμε να φαίνονται τα controls 1 2 3 > >> All. Αυτή η λειτουργία υλοποιείται στην πρώτη if που φαίνεται στην παραπάνω εικόνα. Μάλιστα το σύνολο των σελίδων που εμφανίζεται το έχουμε ορίσει ίσο με τρία. Αντίστοιχα στο επόμενο else γίνεται το αντίθετο καθώς εμφανίζουμε μόνο τα controls για τις προηγούμενες σελίδες και το All, δηλαδή τα κουμπιά << < 4 5 6 All.

Παρακάτω θα δείξουμε τη μέθοδο με το ερώτημα στη βάση για το γέμισμα του δεύτερου πίνακα. Ο ίδιος ακριβώς κώδικας χρησιμοποιήθηκε για το γέμισμα του τρίτου πίνακα μόνο που τα ερωτήματα αφορούσαν το πίνακα sub_menu και όχι τον menu.

Ορίζουμε εκ νέου τη global μεταβλητή $PAGING_LIMIT και πραγματοποιούμε μια σύνδεση στη βάση. Ελέγχουμε αν η μεταβλητή $page == -1 και αν είναι τότε με τη $_SESSION[‘menu_pg2_max’] = 1 εμφανίζουμε όλο το πίνακα χωρίς pagination παίρνοντας όλα τα αποτελέσματα από τη βάση μας. Διαφορετικά παίρνουμε από τη βάση και εμφανίζουμε μόνο τα αποτελέσματα που θέλουμε για τη συγκεκριμένη σελίδα του πίνακα.

```php
function getMenuData($page){
    global $PAGING_LIMIT;

    //connect to database
    $con = dbConnect();

    if ($page == -1){//show all

        $_SESSION['menu_pg2_max'] = 1;//show one page only

        $result = executeQuery($con,"SELECT M_ID,M_TYPE,CAT_DESC,PRICE FROM menu
        inner join CATEGORIES on MENU.CAT_ID=CATEGORIES.CAT_ID ORDER BY M_TYPE");

    }else{
        $skip = $page * $PAGING_LIMIT;//how many records to skip
        $skip = mysqli_real_escape_string($con,$skip);

        //count rows
        $result = executeQuery($con,"SELECT count(M_ID) as cnt FROM menu
                inner join CATEGORIES on MENU.CAT_ID=CATEGORIES.CAT_ID");
        $row = mysqli_fetch_array($result);

        //set number of pages
        $_SESSION['menu_pg2_max'] = ceil($row['cnt'] / (double)($PAGING_LIMIT));

        $result = executeQuery($con,"SELECT M_ID,M_TYPE,CAT_DESC,PRICE FROM menu inner join CATEGORIES
                    on MENU.CAT_ID=CATEGORIES.CAT_ID ORDER BY M_TYPE LIMIT $skip,$PAGING_LIMIT");
    }
    return $result;
}
```

Εικόνα 43: getMenuData() στο app_logic.php

Στην ουσία παραλείπουμε με τη βοήθεια της μεταβλητής $skip τις απαραίτητες εγγραφές χρησιμοποιώντας την στη LIMIT του select που στέλνουμε στη βάση. Για να βρούμε το σύνολο των σελίδων που πρέπει να έχει ο πίνακας μας, μετράμε τις εγγραφές του πίνακα που θέλουμε να εμφανίσουμε και διαιρούμε το αποτέλεσμα με τη $PAGING_LIMIT. Παίρνουμε το ceiling του αποτελέσματος για να γίνει στρογγυλοποίηση προς τα πάνω αφού χρειαζόμαστε μια extra σελίδα αν το αποτέλεσμα της διαίρεσης έχει υπόλοιπο διάφορο του μηδενός. Με την ίδια λογική έχει υλοποιηθεί και η getSubmenuData().

Εικόνα 44: Μηνύματα λάθους

Φυσικά όπως και σε προηγούμενα tabs έτσι και εδώ είναι απαραίτητος ο έλεγχος κατά την εισαγωγή δεδομένων στους πίνακες. Δεν μπορούμε να βάλουμε νέο category εάν αυτό υπάρχει ήδη, όπως επίσης δεν μπορούμε να εισάγουμε σε καμία φόρμα κενά strings, δηλαδή να αφήσουμε κενά πεδία. Θα λάβουμε το μήνυμα Fill in all fields σε αυτή την περίπτωση. Ακόμη δεν μπορούμε να εισάγουμε τίποτε άλλο εκτός από θετικό αριθμό στα πεδία price όπως είναι και το λογικό και σε αυτή την περίπτωση θα πάρουμε ένα text balloon με το κατάλληλο μήνυμα.

Η πολυλειτουργικότητα των φορμών οφείλεται στην τεχνολογία JQuery που χρησιμοποιείται στη συγκεκριμένη περίπτωση φόρμας. Στη δεύτερη φόρμα συγκεκριμένα θα παρατηρήσουμε ότι όταν επιλεγεί το radio button Submenu τότε το checkbox για την πρόσθεση του προϊόντος στο cellar χάνεται. Αυτό χρειάζεται για να μην μπορεί κατά λάθος ο Administrator του POS να εισάγει τιμές στο cellar τις οποίες θα πρέπει να σβήσει μετά λόγω λάθους. Το ίδιο συμβαίνει και στη τρίτη φόρμα όπου αναλόγως με την επιλογή του Menu ή Submenu, η λίστα με τα προϊόντα που διατίθενται για αλλαγή τιμής είναι διαφορετική. Να σημειωθεί το εξής σημαντικό στοιχείο στο σημείο αυτό, πως αν και έχουμε πολλά πεδία με το ίδιο όνομα submenu, το καθένα ανήκει σε διαφορετικό category. Αυτό συμβαίνει για πρακτικούς λόγους κατά τη χρήση του POS από την εφαρμογή client στο Android. Κατά την προσπάθεια αλλαγής της τιμής ενός submenu θα αλλάξουν ΟΛΑ τα submenu με το όνομα που επιλέξαμε ανεξάρτητα από το category. Αυτό δεν είναι κάποιο bug της εφαρμογής απλά είναι λογικό όταν αλλάξει η χρέωση ενός υπομενού να αλλάζει για όλες τις κατηγορίες προϊόντων στις οποίες ανήκει. Για παράδειγμα αν αποφασίσουμε να χρεώνουμε το γάλα που βάζουμε σε καφέδες, τσάι ή οπουδήποτε αλλού χρησιμοποιείται το γάλα, δεν έχει νόημα να το χρεώσουμε σε όσους πίνουν καφέ, αλλά όχι σε όσους πίνουν τσάι. Οπότε η αλλαγή στην τιμή λειτουργεί κατ’ αυτό τον τρόπο. Παρακάτω θα δούμε τον κώδικα με τον οποίο γίνεται η εναλλαγή λιστών στην αλλαγή τιμής και η απόκρυψη του checkbox στη φόρμα Add menu.

Ο τρόπος με τον οποίο γίνονται οι ενέργειες απόκρυψης και εναλλαγής, αφορά αποκλειστικά τα id των radio buttons. Αυτά είναι που όταν γίνουν checked τότε έχουμε τις εναλλαγές στις φόρμες. Το συγκεκριμένο αρχείο

γίνεται include στο manage.php, όπως και πολλά άλλα αρχεία του ίδιου τύπου για την καλή λειτουργία όλου του προγράμματος.

```
//menu submenu form
$("#menuradio").change(function() {
    if(this.checked) {
        $("#cellardiv").slideDown();
        $("html, body").animate({ scrollTop: $(document).height() }, 1000);
    }
});

$("#submenuradio").change(function() {
    if(this.checked) {
        $("#cellardiv").slideUp();
        $("html, body").animate({ scrollTop: $(document).height() }, 1000);
    }
});

//price form
$("#submenuselectdiv").hide();

$("#menuradioprice").change(function() {
    if(this.checked) {
        $("#menuselectdiv").slideDown();
        $("#submenuselectdiv").slideUp();
    }
});

$("#submenuradioprice").change(function() {
    if(this.checked) {
        $("#menuselectdiv").slideUp();
        $("#submenuselectdiv").slideDown();
    }
});
```

Εικόνα 45: menu.js στο manage.php

Τέλος θα δείξουμε τον τρόπο που εισάγουμε στοιχεία στη βάση μας μέσα από τη μέθοδο addNewMenuSubmenu() της οποίας η υλοποίηση μοιάζει με τις υπόλοιπες μεθόδους εισαγωγής αντικειμένων στη βάση στις άλλες φόρμες του menu tab:

Όπως και στην απόκρυψη ορισμένων επιλογών κατά περίπτωση με ajax έτσι και εδώ βασικό ρόλο στο αποτέλεσμα της μεθόδου παίζουν τα values από τα radiobuttons menu και submenu. Αναλόγως με την επιλογή, εκτελείται το κατάλληλο query στη βάση δεδομένων και για τον έλεγχο ύπαρξης ή όχι της τιμής που θέλουμε να εισάγουμε στον πίνακα. Όπως αναφέραμε και παραπάνω στην περίπτωση που η εισαγωγή αφορά ένα submenu τότε εξετάζεται και ο τύπος της εισαχθείσας κατηγορίας σε συνδυασμό με το όνομα του υπομενού που δώσαμε.

Σε περίπτωση που το αποτέλεσμα των ερωτημάτων ελέγχου αποτελείται από μια γραμμή, άρα υπάρχει το αντικείμενο ήδη στη βάση, τότε αρχικοποιούμε τις μεταβλητές στον πίνακα SESSION, την menu_exists και

την submenu_exists για να εμφανιστούν τα κατάλληλα μηνύματα στις φόρμες που φαίνονται στην εικόνα 44. Σε περίπτωση που δεν υπάρχει το αντικείμενο στον πίνακα τότε εκτελείται κανονικά το insert στη βάση και με το redirection το ίδιο tab το οποίο εκτελείται πάντα μπορούμε να δούμε το προϊόν που προσθέσαμε να εμφανίζεται στον αντίστοιχο πίνακα.

```
function addNewMenuSubmenu(){

        // connect to database
        $con = dbConnect ();
        $new_menu_type=mysqli_real_escape_string($con,trim($_POST['type']));
        $new_menu_price = mysqli_real_escape_string($con,trim($_POST['price']));
        $new_menu_catid = mysqli_real_escape_string($con,$_POST['cat_id']);

        if($_POST ['type'] == '' || $_POST ['price'] == ''){
            $_SESSION['menu_form_error']=true;
            //add a menu
        }elseif($_POST['input_menu']=='menu'){

            $result = executeQuery ( $con, "SELECT M_TYPE FROM MENU
                                    WHERE M_TYPE = '$new_menu_type' and CAT_ID='$new_menu_catid'" );
            $numrows = mysqli_num_rows ( $result );

            if ($numrows!=0){
                $_SESSION['menu_exists']=true;
            }else{

                executeQuery ( $con, "INSERT INTO MENU (M_ID,M_TYPE,CAT_ID,PRICE)
                        VALUES (null, '$new_menu_type','$new_menu_catid','$new_menu_price') " );

                if (isset($_POST['celladd'])){
                    executeQuery ( $con,"INSERT INTO CELLAR VALUES(NULL,'$new_menu_type','0')");
                }
            }
            //add a submenu
        }elseif($_POST['input_menu']=='submenu'){

            $result = executeQuery ( $con, "SELECT S_TYPE FROM SUB_MENU
                                    WHERE S_TYPE = '$new_menu_type' and CAT_ID='$new_menu_catid'" );

            $numrows = mysqli_num_rows ( $result );

            echo $numrows;

            if ($numrows!=0){
                $_SESSION['submenu_exists']=true;
            }else{
                executeQuery ( $con, "INSERT INTO SUB_MENU (S_ID,S_TYPE,CAT_ID,PRICE)
                        VALUES (null, '$new_menu_type','$new_menu_catid','$new_menu_price') " );
            }
        }

        header ( "Location: /presentation/manage.php?tab=3" );
}
```

Εικόνα 46: addMenuSubmenu() στο applogic.php

3.6.4. ΚΑΡΤΕΛΑ (reports.php)

Όπως σε όλα τα POS έτσι και στο δικό μας, υπάρχει η δυνατότητα ελέγχου της κίνησης προϊόντων ανά πάσα στιγμή, η λήψη πληροφοριών για την ενεργητικότητα του κάθε σερβιτόρου ή σεφ του μαγαζιού και άλλα ακόμα σημαντικά reports.

Εικόνα 47: Reports tab στο manage.php

Για την υλοποίηση όλων αυτών χρειάστηκε να δημιουργήσουμε μεθόδους με πολλές γραμμές κώδικα στο κυρίως αρχείο του προγράμματος μας, στο app_logic.php, τις οποίες θα αναλύσουμε παρακάτω. Το αναφέρουμε αυτό για τον λόγο ότι το κυρίως αρχείο που γίνεται include στο manage.php, είναι σχεδόν άδειο αφού δεν κάνει κάτι εκτός από το να περιέχει ένα κείμενο επεξήγησης, ένα κουμπί, μια εικόνα τύπου gif και ένα άδειο division (div). Αυτό το div είναι που θα γεμίσουμε με πίνακες και data μέσω JQuery για να έχουμε όλα τα απαραίτητα reports.

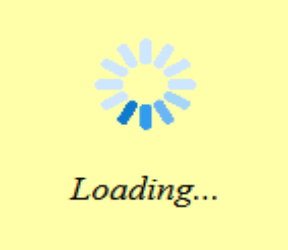
```
require_once('../logic/app_logic.php');

    if(loggedin(true)){

        echo '<div class="text">Find any kind of report you want in this section.<br/> <br/>
                Press the button on your right to see the day\'s traffic so far. <br/><br/>

                <input id ="show" type="button" value="Show current traffic" style="min-width:150px ; height: 25px"></input>
                <center><img id="loading" src="../img/loading.gif" height="50" width="50" /></center><br/>
                <center id="loading_msg" style="font-size : 18px">Loading...</center>
                <br/><br/></div>
                <div id="full_report"></div>';
    }
```

Εικόνα 48 : reset.php

Με τον παραπάνω κώδικα και μόνο γεμίζουμε την τέταρτη καρτέλα μας, αλλά το μυστικό όπως είπαμε κρύβεται στο div με id full report.

Αρχικά οι επιλογές μας είναι δύο μπορούμε να πατήσουμε το κουμπί Show Current traffic ή το κουμπί End of day. Αν και το κουμπί End of day βρίσκεται εκτός καρτέλας σχεδιάστηκε για να γεμίζει την συγκεκριμένη καρτέλα με πληροφορίες. Θα αναλύσουμε σε πρώτο στάδιο τη λειτουργία του πρώτου κουμπιού. Πατώντας αυτό το κουμπί εμφανίζεται η εικόνα Loading μέχρι να εκτελεστούν τα κατάλληλα queries στη βάση. Έπειτα μπορεί ο Administrator να πάρει στοιχεία για κάθε κατηγορία προϊόντος που πουλήθηκε, σε τι

ποσότητα πουλήθηκε και τη συνολική τιμή τους, μπορεί να δει τον αριθμό των παραγγελιών που έχει στείλει μέχρι εκείνη τη στιγμή ο κάθε ενεργός του χρήστης, μπορεί να δει στατιστικά στοιχειά για το ποιος σερβιτόρος ή σεφ έχει ολοκληρώσει τις περισσότερες παραγγελίες και τέλος να δει ένα 2μηνο report για τα προϊόντα που έχουν πουληθεί.



Εικόνα 49: Loading.gif

Products sold today

Categories	Amount	Total Price
COFFEE	3	11
REFRESH	0	0
ALCOHOL	0	0
SNACK	0	0
COCTAIL	0	0
BEER	2	9
ICE_C	0	0
TEA	1	5
NON-ALCOHOL	0	0
ENERGY_DRINK	0	0
TOTAL	---	25

User traffic today

Users	Total Orders	Total Price
TASOS	2	6
NIKOS	1	5
test1	1	5
paul	1	3
TOTAL	6	25

General statistics today

Element	MAX	MIN
Orders / User	TASOS	NIKOS
Orders / Table	T13	T2
Orders / Chef	tipota akoma	tipota akoma
Orders / Time	tipota akoma	tipota akoma
Product Type	ESPRESSO	KAISER

60-day product traffic

Categories	Amount	Total Price
COFFEE	3	11
REFRESH	0	0
ALCOHOL	0	0
SNACK	0	0
COCTAIL	0	0
BEER	2	9
ICE_C	0	0
TEA	1	5
NON-ALCOHOL	0	0
ENERGY_DRINK	0	0
TOTAL	---	25

Εικόνα 50: Πίνακες της καρτέλας reports

Όλοι οι αριθμοί που θα γεμίσουν τον καθένα πίνακα από αυτούς έρχονται από τη βάση δεδομένων με πολλά διαφορετικά queries. Οι τρεις πρώτοι πίνακες αφορούν τα δεδομένα της ημέρας ενώ ο τελευταίος πίνακας είναι ένα query στον πίνακα fin_orders ο οποίος περιέχει τα δεδομένα των τελευταίων 60 ημερών. Παρακάτω θα δούμε τις μεθόδους με τις οποίες γεμίζει ο κάθε πίνακα ξεχωριστά, πατώντας το Show Current Traffic.

Κατά το φόρτωμα του εγγράφου ορίζουμε να γίνει hide() στην εικόνα και στο κείμενο Loading. Στη συνέχεια ελέγχουμε την ενέργεια του κουμπιού και τη μέθοδο click. Έτσι εμφανίζουμε το loading με slide down και με ajax πλέον καλούμε το αρχείο reports_helper.php να γεμίζει το άδειο μέχρι στιγμής div με id full_report. Στη συνέχεια κάνουμε απόκρυψη των μηνυμάτων Loading.

```
$(document).ready(function(){
    $("#loading").hide();
    $("#loading_msg").hide();
    $("#show").click(function() {
        $("#loading").slideDown();
        $("#loading_msg").slideDown();
        $.ajax({url:"reports_helper.php",success:function(result){
            $("#full_report").html(result);
            $("#loading").slideUp();
            $("#loading_msg").slideUp();
        }});
    });
});
```

Εικόνα 51: JQuery.js διαχείρισης Show current traffic

Η reports_helper που καλείται απλώς καλεί τη μέθοδο reports_helper από το app_logic.php, με τον τρόπο που φαίνεται στο screenshot παρακάτω.

```
<?php session_start();
    require_once('../logic/app_logic.php');

    if(loggedin(true)){
        reports_helper();

    }
?>
```

Εικόνα 52: reports_helper.php

Η reports_helper () με τη σειρά της κάνει όλη τη δουλειά που χρειάζεται για να γεμίσει το άδειο div.

```
function reports_helper() {
    $result = getTotalCategoryReport ();
    $result2 = getCategorySumReport ();
    $row2 = mysqli_fetch_array ( $result2 );

    // Report TOTAL table
    echo '<div style="display: inline-block; vertical-align: top;">
                <i class="tables_desc">Products sold today</i>
                 <table class="tables">
                  <tr>

                    <td>Categories</td>
                    <td>Amount</td>
                    <td>Total Price</td>
                  </tr>
                ';

    while ( $row = mysqli_fetch_array ( $result ) ) {
        echo '<tr><td>' . $row ['cat_desc'] . '</td><td>' .
        $row ['amount'] . '</td><td>' . $row ['total_price'] . '</td></tr>';
    }
    echo '<tr><td><b>TOTAL</b></td><td>---</td><td> ' . $row2 ['sum'] . '</td></tr>';

    echo '</table></div>';
```

Εικόνα 53: Γέμισμα πρώτου πίνακα στο reports.php

Δημιουργεί όλους του πίνακες της καρτέλας και τους γεμίζει με μια σειρά από queries των οποίων τα αποτελέσματα παίρνει από άλλες μεθόδους του app_logic. Η μέθοδος αυτή αποτελείται από πάρα πολλές γραμμές κώδικα και εμείς θα εξηγήσουμε αναλυτικά τον κώδικα για τον πρώτο και τον τρίτο πίνακα καθώς τα ερωτήματα προς τη βάση μοιάζουν σε μεγάλο ποσοστό με τους υπόλοιπους πίνακες.

Δημιουργούμε έναν πίνακα που θα περιέχει όλα τα προϊόντα που πουλήθηκαν σήμερα ανά κατηγορία προϊόντος και με τις ποσότητες του στην επόμενη στήλη και στην τρίτη τη συνολική αξία ανά κατηγορία προϊόντος και πάλι. Έτσι λοιπόν καλούμε τις getTotalCategoryReport() και getCategorySumReport(). Η πρώτη μέθοδος επιστρέφει την ποσότητα των m_types του πίνακα fin_orders και το άθροισμα των τιμών τους, οργανωμένα ανά cat_id από τον πίνακα categories. Η δεύτερη επιστρέφει το sum(total_price) του ερωτήματος της προηγούμενης μεθόδου. Ο κώδικας για τις δύο αυτές μεθόδους φαίνεται στην εικόνα 54.

```
function getTotalCategoryReport() {
    $con = dbConnect ();

    $result = executeQuery ( $con, "select g.cat_desc, case when a.num is null then '0' else a.num end as amount,
                                    case when a.price is null then '0' else a.price end as total_price
                                    from categories g left join
                                    (select c.cat_desc, m.cat_id,m.m_id, count(f.m_id) num,sum(f.price) as price from menu m
                                    inner join fin_orders f on f.m_id=m.m_id inner join categories c on c.cat_id=m.cat_id
                                    WHERE f.DAY = (SELECT d.AMOUNT FROM COUNTERS d WHERE d.C_NAME = 'END_OF_DAY')
                                    group by c.cat_id) as a
                                    on a.cat_id=g.cat_id
                                    group by g.cat_id" );

    return $result;
}
function getCategorySumReport() {
    $con = dbConnect ();

    $result = executeQuery ( $con, "select sum(total_price) as sum from
                                    (select g.cat_desc, case when a.num is null then '0' else a.num end as amount,
                                    case when a.price is null then '0' else a.price end as total_price
                                    from categories g left join
                                    (select c.cat_desc, m.cat_id,m.m_id, count(f.m_id) num,sum(f.price) as price from menu m
                                    inner join fin_orders f on f.m_id=m.m_id inner join categories c on c.cat_id=m.cat_id
                                    WHERE f.DAY = (SELECT d.AMOUNT FROM COUNTERS d WHERE d.C_NAME = 'END_OF_DAY')
                                    group by c.cat_id) as a
                                    on a.cat_id=g.cat_id
                                    group by g.cat_id) as v" );
    mysqli_close ( $con );

    return $result;
}
```

Εικόνα 54: getTotalCategoryReport() και getCategorySumReport()

Το κλειδί του ημερήσιου report κρύβεται στο where των δύο select όπου ζητάμε από τη βάση τα αποτελέσματα του fin_orders όπου το πεδίο day είναι

ίσο με το πεδίο end_of_day του πίνακα counters. Έτσι λοιπόν, παρόλο που στον πίνακα fin_orders κρατάμε αρχείο για δύο μήνες πίσω, μπορούμε να έχουμε αποτελέσματα για τη σημερινή μέρα μόνο. Μελλοντικά αυτό μας δίνει τη δυνατότητα να υλοποιήσουμε reports για όποια μέρα του προηγούμενου διμήνου θέλουμε.

Για τον δεύτερο πίνακα ο οποίος αφορά μόνο τους χρήστες του POS και τη δραστηριότητα τους, η λογική είναι ίδια με την παραπάνω, με τη μόνη αλλαγή ότι τώρα δεν μας ενδιαφέρει πόσες φορές εμφανίζεται στον fin_orders κάποιο m_type άλλα τα u_id, που αντιστοιχούν στον πίνακα users και είναι μοναδικά. Έτσι μετράμε τις παραγγελίες που έχει στείλει ο κάθε σερβιτόρος και το συνολικό ταμείο που έχει πάνω του αυτή τη στιγμή.

```
// fill in third table
function getGeneralDataReport() {
    // MAX ORDERS/USER
    $con = dbConnect ();
    $result0 = executeQuery ( $con, "SELECT NAME FROM
                                (SELECT U.NAME AS NAME,COUNT(F.U_ID) AS ORDERS, SUM(PRICE)AS EURO FROM FIN_ORDERS F
                                INNER JOIN USERS AS U
                                WHERE U.U_ID=F.U_ID
                                AND F.DAY = (SELECT C.AMOUNT FROM COUNTERS C WHERE C_NAME = 'END_OF_DAY')
                                GROUP BY F.U_ID ORDER BY ORDERS DESC LIMIT 1)AS A" );
    $result [0] = mysqli_fetch_array ( $result0 );
    // MIN ORDERS/USER
    $result1 = executeQuery ( $con, "SELECT NAME FROM
                                (SELECT U.NAME AS NAME,COUNT(F.U_ID) AS ORDERS, SUM(PRICE)AS EURO FROM FIN_ORDERS F
                                INNER JOIN USERS AS U
                                WHERE U.U_ID=F.U_ID
                                AND F.DAY = (SELECT C.AMOUNT FROM COUNTERS C WHERE C_NAME = 'END_OF_DAY')
                                GROUP BY F.U_ID ORDER BY ORDERS ASC LIMIT 1)AS A" );
    $result [1] = mysqli_fetch_array ( $result1 );
    // MAX PRODUCT TYPE
    $result2 = executeQuery ( $con, "select MP from (SELECT m.M_TYPE as MP,count(f.M_ID)as a
                                FROM FIN_ORDERS f
                                inner join MENU m on m.M_ID=f.M_ID
                                WHERE F.DAY = (SELECT C.AMOUNT FROM COUNTERS C WHERE C_NAME = 'END_OF_DAY')
                                group by m.M_TYPE order by a DESC limit 1)as b" );
    $result [2] = mysqli_fetch_array ( $result2 );
    // MIN PRODUCT TYPE
    $result3 = executeQuery ( $con, "select MP from (SELECT m.M_TYPE as MP,count(f.M_ID)as a
                                FROM FIN_ORDERS f
                                inner join MENU m on m.M_ID=f.M_ID
                                WHERE F.DAY = (SELECT C.AMOUNT FROM COUNTERS C WHERE C_NAME = 'END_OF_DAY')
                                group by m.M_TYPE order by a ASC limit 1)as b" );
    $result [3] = mysqli_fetch_array ( $result3 );
    // MAX ORDERS/TABLE
    $result4 = executeQuery ( $con, "select MT from (SELECT T.TABLE_NUM as MT,count(f.TABLE_NUM)as a
                                FROM FIN_ORDERS f
                                inner join TABLES T on T.TABLE_NUM=f.TABLE_NUM
                                WHERE F.DAY = (SELECT C.AMOUNT FROM COUNTERS C WHERE C_NAME = 'END_OF_DAY')
                                group by T.TABLE_NUM order by a DESC limit 1)as b" );
    $result [4] = mysqli_fetch_array ( $result4 );
    // MIN ORDERS/TABLE
    $result5 = executeQuery ( $con, "select MT from (SELECT T.TABLE_NUM as MT,count(f.TABLE_NUM)as a
                                FROM FIN_ORDERS f
                                inner join TABLES T on T.TABLE_NUM=f.TABLE_NUM
                                WHERE F.DAY = (SELECT C.AMOUNT FROM COUNTERS C WHERE C_NAME = 'END_OF_DAY')
                                group by T.TABLE_NUM order by a ASC limit 1)as b" );
    $result [5] = mysqli_fetch_array ( $result5 );

    mysqli_close ( $con );

    return $result;
```

Εικόνα 55: generalDataReport() στο app_logic.php

Με τον ίδιο ακριβώς τρόπο γεμίζει και ο τέταρτος πίνακας της καρτέλας μας, ο οποίος μας ενημερώνει για την κίνηση των προϊόντων για το τελευταίο δίμηνο, με τη διαφορά ότι το where δεν υπάρχει καθόλου στα ερωτήματα sql αφού θέλουμε όλα τα δεδομένα του πίνακα fin_orders. Το τρόπο με τον οποίο κρατάει μόνο τα συγκεκριμένα data ο πίνακας θα τον δούμε στην αναλυτική περιγραφή του κουμπιού End of day.

Για το γέμισμα του τρίτου πίνακα χρησιμοποιούμε εκ νέου τον πίνακα fin_orders της βάσης μας, με τη διαφορά ότι πλέον κάθε ένα κελί του πίνακα είναι ένα ξεχωριστό ερώτημα στη βάση. Παραπάνω βλέπουμε τη μέθοδο που στέλνει ερωτήματα και επιστρέφει πίνακα αποτελεσμάτων (εικόνα 55). Ο πίνακας result[ ] επιστρέφεται με δεδομένα όπως max και min id εμφάνισης των users που χρησιμοποίησαν σήμερα το POS, ποιο τραπέζι είχε τις περισσότερες παραγγελίες, ποιο τις λιγότερες, ποιος σεφ ήταν πιο παραγωγικός, ποιος λιγότερο, ποιο προϊόν πουλήθηκε περισσότερο κ.α. Σημαντικά στατιστικά στοιχεία όλα αυτά, που αν λάβει υπόψη κάποιος μπορεί να αντλήσει δεδομένα για τη βελτίωση στον τρόπο λειτουργίας της επιχείρησης του. Εδώ κάποιο εργαλείο OLAP για την εξόρυξη δεδομένων θα ήταν ακριβώς ότι χρειάζεται μια σύγχρονη επιχείρηση που νοιάζεται για το μέλλον.

Αυτό που μένει πλέον να εξηγηθεί είναι η λειτουργία του κουμπιού End of day. Εξηγήσαμε πλήρως παραπάνω τις ενέργειες του Show current traffic, οπότε λόγω μεγάλης ομοιότητας των ενεργειών των δύο κουμπιών αυτών, θα αναλύσουμε τις διαφορές που προσφέρει το End of Day. Σε όλα τα μαγαζιά είναι αναγκαίο κάθε βράδυ να πραγματοποιείται ένα κλείσιμο της ημέρας όπου θα έχουμε το συνολικό ταμείο και τα απαραίτητα reports. Σε τι διαφέρει λοιπόν το κουμπί αυτό από το Show current traffic; Η απάντηση είναι απλή. Με το κλείσιμο της ημέρας αυξάνουμε κατά ένα μια τιμή amount του πίνακα counters, για end_of_day, με αποτέλεσμα να έχουμε κάνει ήδη εκκίνηση της επόμενης μέρας. Ταυτόχρονα διαγράφουμε όλες τις εγγραφές του πίνακα fin_orders που έχουν τιμή ημέρας μικρότερη από τη σημερινή μειωμένη κατά 60. Αυτός είναι και ο λόγος για τον οποίο ο τέταρτος πίνακας της καρτέλας reports δεν θέλει κάποια ιδιαίτερα κριτήρια στο select. Είναι ενημερωμένος πάντα ο πίνακας fin_orders με εγγραφές του τελευταίου διμήνου. Σταματάει να έχει υπόσταση το report του τελευταίου διμήνου, αν για κάποιο λόγο κάποιος

«αμελής» administrator δεν πατήσει ποτέ το End of day. Αν όμως κάποιος δεν το κάνει σημαίνει πως δεν ενδιαφέρεται για κανενός είδους στατιστικά στοιχεία. Οπότε συνεχίζει να εκτελείται το πρόγραμμα με την ίδια ημέρα στις εγγραφές του πάντα.

Με το πάτημα του κουμπιού αυτού, εκτελείται όπως και προηγουμένως ένα κώδικας από το αρχείο endofday.js το οποίο αρχικά φορτώνεται στο manage.php κατά την εκκίνηση.

```
$(document).ready(function(){
    $("#loading").hide();
    $("#loading_msg").hide();
        $("#end_of_day").click(function() {
            $("#loading").slideDown();
            $("#loading_msg").slideDown();
            $.ajax({url:"end_of_day.php",success:function(result){
                $("#full_report").html(result);
                $("#loading").slideUp();
                $("#loading_msg").slideUp();

            }});
    });
});
```

Εικόνα 56: JQuery διαχείρησης End of day

Όπως και πρίν έτσι και εδώ γίνεται απόκρυψη και εμφάνιση των μηνυμάτων Loading, και καλείται το end_of_day.php να γεμίσει μέσω ajax το div full_report.

```
<?php

    session_start();
    require_once('../logic/app_logic.php');

    if(loggedin(true)){

        reports_helper();
        endOfDay();
    }

?>
```

Εικόνα 57:end_of_day.php

Καλείται και πάλι ο κώδικας της reports_helper() που θα σχεδιάσει και θα γεμίσει όλους τους πίνακες και επιπλέον με το πέρας της μεθόδου αυτής εκτελείται η endOfDay() που κάνει τις ενέργειες που αναφέραμε παραπάνω,

δηλαδή την διαγραφή εγγραφών που είναι περισσότερο από δύο μήνες παλιές, και την εκκίνηση ουσιαστικά της επόμενης ημέρας. Στη συνέχεια αφού πατηθεί μια φορά το κουμπί αυτό για το ενεργό session του login, απενεργοποιούμε τη δυνατότητα εκτέλεσης εντολών της endOfDay() θέτοντας τη μεταβλητή $_SESSION['end_once']=true. Πλέον το κουμπί αυτό θα συμπεριφέρεται σαν να πατήσαμε το Show current traffic. Επίσης αλλάζουμε χρώμα στο κουμπί End of Day για να ξέρει ο Administrator ότι δεν μπορεί πλέον να κλείσει και πάλι την ημέρα εκτός και αν κάνει Logout και ξανά Login. Αυτό αν και φαίνεται να λειτουργεί σαν bug στη συγκεκριμένη εφαρμογή, δεν θα δημιουργήσει πρόβλημα στη σωστή λειτουργία του μαγαζιού για την επόμενη μέρα, καθώς όσο και να αυξηθεί η τιμή του end_of_day στον πίνακα counters, με την έλευση μιας καινούριας παραγγελίας από κάποιο σερβιτόρο θα μπει η κατάλληλη τιμή στον πίνακα fin_orders.

```
//END OF DAY FUNCTION
function endOfDay(){
    if(!isset($_SESSION['end_once'])){
        // connect to database
        $con = dbConnect ();

        executeQuery ( $con, "UPDATE COUNTERS SET AMOUNT = AMOUNT + 1
                              WHERE C_NAME='END_OF_DAY'" );
        //keep a two month total product report
        executeQuery ($con, "DELETE FROM fin_orders WHERE DAY
        < (SELECT AMOUNT FROM COUNTERS WHERE C_NAME='END_OF_DAY')-60 " );
        $_SESSION['end_once']=true;
    }
}
```

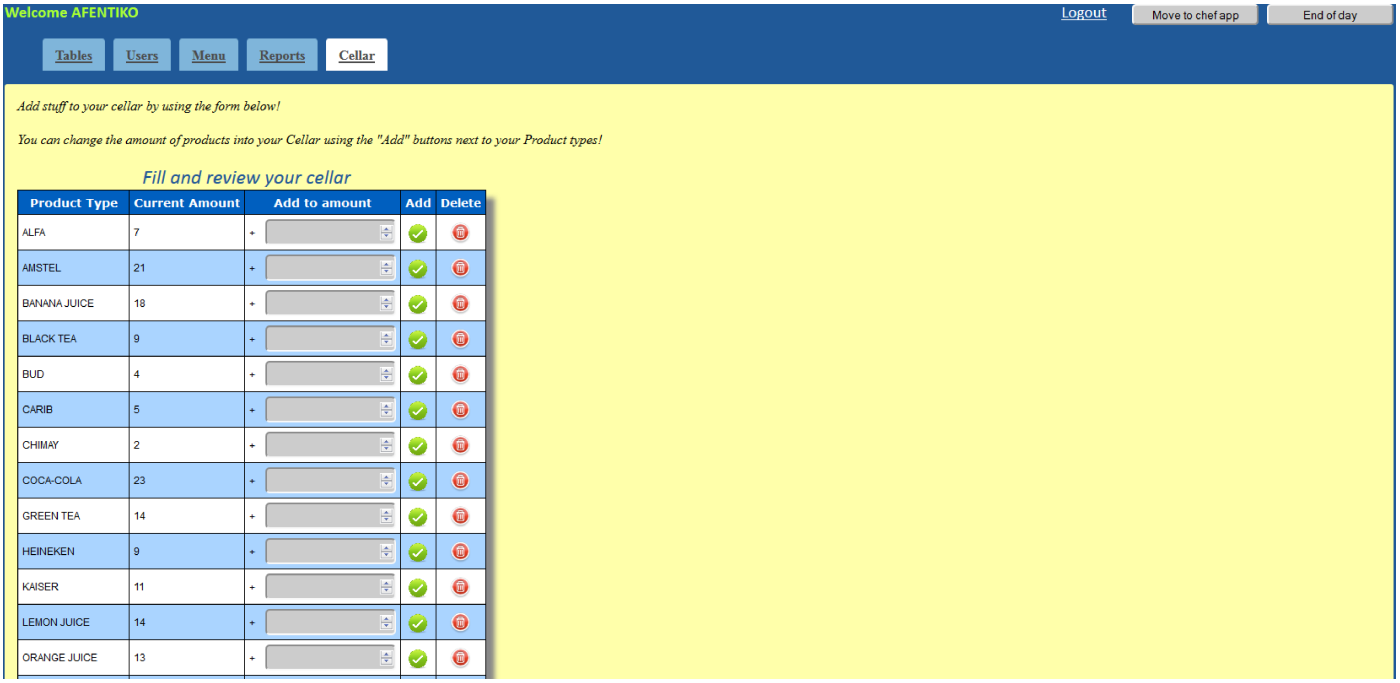
Εικόνα 58: endOfDay() στο app_logic.php

Εικόνα 59: Κατάσταση κουμπιού End of day πριν και μετά

3.6.5. ΚΑΡΤΕΛΑ (cellar.php)

Η δυνατότητα ελέγχου της αποθήκης για προϊόντα τα οποία διατίθενται σε τεμάχια και όχι χύμα στους πελάτες, είναι επίσης διαθέσιμη στο POS που δημιουργήσαμε. Στο πέμπτο και τελευταίο tab του administrator interface περιέχεται ένας πίνακας που αφορά αποκλειστικά τον πίνακα cellar της βάσης

μας. Όχι μόνο μπορεί να προσθέσει ή να αφαιρέσει ποσότητες προϊόντων στην αποθήκη του ο Administrator, αλλά μπορεί να δει τα ποσά αυτά να μειώνονται κάθε φορά που κάνει refresh τη σελίδα, καθώς με κάθε παραγγελία που έρχεται, αυτόματα μειώνεται ο τύπος μενού ο οποίος περιέχεται στη παραγγελία. Όπως θα δείξουμε παρακάτω, JQuery και ajax έκαναν και πάλι τη δουλειά για μας. Παρακάτω βλέπουμε την καρτέλα Cellar.



Εικόνα 60: Καρτέλα Cellar στο manage.php

Επιλέξαμε να μην χρησιμοποιήσουμε pagination στον συγκεκριμένο πίνακα γιατί είναι απαραίτητο να φαίνονται όλα τα προϊόντα με τις ποσότητες τους στον Administrator για να μπορεί να εισάγει με μιας ποσότητες προϊόντων σε όποια Product Type επιθυμεί. Αντ' αυτού έχουμε τοποθετήσει αλφαβητικά τα προϊόντα για να μπορεί εύκολα να ανατρέξει σε εκείνο που τον ενδιαφέρει. Παρακάτω θα δούμε τον κώδικα με τον οποίο ο συγκεκριμένος πίνακας γεμίζει.

Όπως και στις προηγούμενες καρτέλες έτσι και εδώ, παίρνουμε σε μια μεταβλητή τα αποτελέσματα της μεθόδου getCellarData(), που μας επιστρέφει όλα τα δεδομένα του πίνακα cellar ταξινομημένα. Με ένα while loop δημιουργούμε γραμμές και τις γεμίζουμε με τα data. Στην τρίτη στήλη, τοποθετούμε ένα input τύπου number για να μπορεί να εισάγει ο διαχειριστής το ποσό του amount που θέλει να προσθέσει ή να αφαιρέσει στο cellar του.

```
$result = getCellarData();

echo '
<div style="width : 40%">
<i class="tables_desc" style="margin-right: 60px">Fill and review your cellar
       </i>
<table class="tables" id="cellar_table">
  <tr>

    <td>Product Type</td>
    <td>Current Amount</td>
    <td>Add to amount</td>
    <td>Add</td>
    <td>Delete</td>
  </tr>
';

while($row = mysqli_fetch_array($result)) {
  echo '<tr><td>' . $row['PRODUCT_TYPE'] . '</td><td>' . $row['AMOUNT'].'</td><td>+   
  <input type="number" id="add_amount'.$row['C_ID'].'" width="20"></td><td>
  <img id="tick_icon'.$row['C_ID'].'" class="tick_icon" src="../img/tick.png" alt="Add to amount" style="width:30px;height:30px"></a></td><td>
  <a href="cellar.php?del=' . $row['C_ID'] .'" >
  <center><img id="del_icon" src="../img/delete.png" alt="Delete" style="width:30px;height:30px"></center></a> </td></tr>';
}

echo '<tr><td style="text-align: center" colspan="5"><input type="button" id="update" style="width: 450px" value="Add All Amounts"/></td></tr></table><br/>
    <img id="wait" src="../img/loading.gif" height="50" width="50" style="margin-right : 100px"/><br/>
</div>';
```

Εικόνα 61: Γέμισμα φόρμας στο cellar.php

Στη τέταρτη στήλη τοποθετούμε την εικόνα tick.png της οποίας το id είναι συνάρτηση του c_id, το οποίο είναι και το κύριο κλειδί του πίνακα cellar. Αυτό συμβαίνει για να μπορούμε να ελέγξουμε το id της κάθε εικόνας του πίνακα με τη JQuery όταν αυτό πατηθεί, για να γίνει η διαδικασία πρόσθεσης του amount μόνο για τη συγκεκριμένη γραμμή του πίνακα. Στην τελευταία στήλη τοποθετούμε κλασσικά το κουμπάκι της διαγραφής όπως και σε όλους τους πίνακες μέχρι τώρα.

Στο αρχείο cellar.js έχουμε τις δύο μεθόδους που κάνουν όλη τη δουλειά για μας. Και οι δύο ελέγχουν το κλικ σε κάποιο αντικείμενο, η πρώτη στο εικονίδιο πρόσθεσης ενός αντικειμένου στον πίνακα, και η δεύτερη στο κουμπί Add All Amounts στο κάτω μέρος του πίνακα που εισάγει όλες τις εισαχθείσες τιμές με μιας στον cellar. Στην πρώτη περίπτωση καλούμε μέσω ajax το link cellar.php?c_id=”current_id”&add_amount=”add_amount”+currentId. Έτσι εξασφαλίζουμε ότι οι $_GET μεταβλητές c_id και add_amount έχουν σεταριστεί, και με τη φόρτωση του αρχείου cellar.php, εκτελείται ο κώδικας της addCellarAmount() από το app_logic.php. Τότε γίνεται το κατάλληλο update στη βάση δεδομένων και τέλος έχουμε την επαναφόρτωση της σελίδα όπου θα δούμε τη αλλαγή που κάναμε. Το ίδιο συμβαίνει και στην περίπτωση που πατήσουμε το κουμπί Add All Amounts. Για κάθε πεδίο της στήλης Add to

amount, το οποίο έχει τιμή διάφορη του -1 καλούμε τη συγκεκριμένη μέθοδο από το app_logic.php και στέλνουμε ένα ένα τα request για πρόσθεση των ποσών στη βάση (async:false). [17] Έτσι περιμένοντας να τελειώσει το ένα request πριν στείλουμε το άλλο αποτρέπουμε το γεγονός να χαθεί κάποιο request στο δρόμο.

```
$(document).ready(function(){
    $("#wait").hide();
    $(".tick_icon").click(
        function() {
            var currentId = $(this).attr('id').substring(9);//cellar id

            $.ajax({url:"cellar.php?c_id="+currentId+'&add_amount='+
            $('#'+'add_amount'+currentId).val(),success:function(result){
                location.reload();
            }});
        }
    );
    $("#update").click(

        function() {
            var chil = $("#cellar_table td").children();

            for (var i = 0; i < chil.length - 1; i++){

                var id = $(chil[i]).attr('id');

                if (id && id.indexOf('add_amount')!=-1){
                    var amount = $('#'+id).val();
                    var cellarId = id.substring(10);//cellar id

                    if (amount != ''){
                        $("#wait").show();
                        $.ajax({url:"cellar.php?c_id="+cellarId+'&add_amount='+
                        $('#'+'add_amount'+cellarId).val(),async: false});
                    }
                }
            }
            $("#wait").hide();
            location.reload();
        }
    );
});
```

Εικόνα 62: cellar.js

Παίρνοντας από το ajax τις τιμές των μεταβλητών $_GET[‘add_amount’] και $_GET[‘c_id’] στέλνουμε το κατάλληλο request στη βάση για να κάνουμε update τον cellar προσθέτοντας την τιμή που θέλει ο χρήστης. Στην περίπτωση που πατήσουμε το Add All Amounts τότε η συγκεκριμένη μέθοδος θα τρέξει τόσες φορές όσες και ο αριθμός των αλλαγών που επιθυμεί ο χρήστης.

```php
function addCellarAmount() {
    // connect to database
    $con = dbConnect ();
    $adding_value = mysqli_real_escape_string ( $con, $_GET['add_amount'] );

    if ($adding_value != 0) {
        $c_id = mysqli_real_escape_string ( $con, $_GET ['c_id'] );

        executeQuery ( $con, "update cellar set AMOUNT = AMOUNT +
        '$adding_value' where C_ID='$c_id'" );
    }
    // redirect
    header ( "Location: /presentation/manage.php?tab=5" );
}
```

Εικόνα 63: addCellarAmount() στο app_logic.php

Κατά τη διαγραφή εκτελείται ο κώδικας της μεθόδου deleteCellarType() ο οποίο βασίζεται στο c_id της εκάστοτε εγγραφής όπως γίνεται και σε όλους τους πίνακες όλων καρτελών.

```php
function deleteCellarType() {
    // connect to database
    $con = dbConnect ();

    $c_id = mysqli_real_escape_string ( $con, $_GET ['del'] );
    $result = executeQuery ( $con, "DELETE FROM CELLAR WHERE C_ID = '$c_id'" );

    // redirect
    header ( "Location: /presentation/manage.php?tab=5" );
}
```

Εικόνα 64: deleteCellarType() στο app_logic.php

ΕΠΙΛΟΓΟΣ

Συμπερασματικά αυτό που καταλαβαίνει κανείς είναι ότι το σύνολο της εφαρμογής για τον administrator βασίζεται σε ερωτήματα προς τη βάση δεδομένων και τη διαχείριση των αποτελεσμάτων των select, update, delete και insert με την php. Μεταβλητές τύπου SESSION είναι αυτές που χρησιμοποιούνται για να γίνουν οι απαραίτητες ενέργειες ελέγχου κατά τη διάρκεια που ο χρήστης είναι logged in και μεταβλητές τύπου POST και GET είναι αυτές με τις οποίες ο διαχειριστής αλληλεπιδρά με τη βάση.

Όλο το πρόγραμμα εκτελείται κάτω από τον κατάλογο htdocs του XAMPP, με τον Apache Server και τη MySQL να τρέχουν παράλληλα. Το project είναι καλά οργανωμένο εκεί σε υποφακέλους, με τον συνολικό κώδικα των

μεθόδων τις οποίες χρειαστήκαμε σε πολλά σημεία της εφαρμογής να βρίσκεται στο αρχείο logic/app_logic.php.

Το επίπεδο των λειτουργιών ενός UI για τον administrator μπορεί να γίνει ακόμα καλύτερο στο μέλλον προσθέτοντας λειτουργίες που δεν αφορούν μόνο τον διαχειριστή σε καθημερινή βάση αλλά και το σύνολο της επιχείρησης, όπως για παράδειγμα η προσθήκη ή διαγραφή παραγγελίας σε live συνθήκες, η εκτύπωση δελτίου ημερήσιας κίνησης, το γνωστό «Z», για λόγους εφορίας, online σύνδεση με τους προμηθευτές κάθε φορά που αδειάζει ο πίνακας cellar, χάρτες εύρεσης των delivery ανά πάσα στιγμή και σύνδεση με ταμειακή μηχανή για αυτόματη έκδοση αποδείξεων.

4. ΚΕΦΑΛΑΙΟ ΤΕΤΑΡΤΟ

CHEF UI

ΕΙΣΑΓΩΓΗ

Η εκτέλεση μιας παραγγελίας σε ένα μαγαζί προϋποθέτει η παραγγελία να φτάσει στον προορισμό της. Ο προορισμός αυτός είναι συνήθως η κουζίνα ή το μπαρ ενός μαγαζιού και τελικά το άτομο το οποίο θα ετοιμάσει την παραγγελία. Το Chef UI που σχεδιάσαμε για τον σκοπό αυτό είναι λιτό μα πάνω από όλα λειτουργικό και εύχρηστο. Μπορεί να εκτελεστεί σε Tablet με οποιοδήποτε λειτουργικό σύστημα, αλλά προτείνεται η χρήση του σε οθόνη αφής των 20” και άνω για την καλύτερη εξυπηρέτηση και ευκολία. Ο σκοπός αγιάζει τα μέσα, οπότε μπορεί ο καθένας να επιλέξει τον τρόπο που τον βολεύει να βλέπει τις παραγγελίες προς εκτέλεση εν ώρα δουλειάς και κάτω από πίεση. Χρησιμοποιώντας μια μεγάλη οθόνη μπορεί κάποιος να βλέπει τις παραγγελίες του γρήγορα και άμεσα, ενώ χρησιμοποιώντας ένα tablet 10” εξοικονομούμε χώρο αλλά χάνουμε σε ευκολία ανάγνωσης των παραγγελιών μιας και η οθόνη είναι πολύ μικρή για μια τέτοια δουλειά.

Το σύνολο της συγκεκριμένης εφαρμογής αποτελείται εξ ολοκλήρου από JQuery [19] που παράγει όσο εκτελείται html και τη γεμίζει κατάλληλα. Χρησιμοποιείται αρχείο json για τη συλλογή των αποτελεσμάτων από τη βάση και ajax για την ολοκλήρωση των ενεργειών του chef.

4.1. ACTIVE ORDERS AND INTERACTION (chef.php)

Μόλις πραγματοποιηθεί είσοδος στην εφαρμογή μας με λογαριασμό waiter ή admin θα παρουσιαστεί η παρακάτω εικόνα. Να τονίσουμε ότι η συγκεκριμένη εφαρμογή μπορεί να χρησιμοποιηθεί από πολλά τερματικά ταυτόχρονα και να είναι όλα ενημερωμένα για την κατάσταση των ενεργών παραγγελιών. Οπότε ενδείκνυται η λειτουργία της από πολλούς σεφ. Τα tabs των τραπεζιών μειώνονται όσο εκτελούνται οι παραγγελίες και αυξάνονται όσο αυτές συνεχίζουν να έρχονται από τον client που της δημιουργεί στο android.

Εικόνα 65: chef.php populated

Βλέπουμε ότι είναι επιλεγμένη η παραγγελία T9. Αυτή περιέχει δύο προϊόντα τα οποία έχουν εκτελεστεί από τον σεφ, και άλλα δύο τα οποία όχι. Πατώντας ο σεφ το κουμπάκι Done στο τέλος της παραγγελίας εμφανίζεται το κουμπάκι OK το οποίο είναι εκεί για επιβεβαίωση, και το value του κουμπιού Done γίνεται αυτόματα Undo. Αν και το OK πατηθεί τότε η συγκεκριμένη γραμμή της παραγγελίας φεύγει από εκεί και τη θέση της παίρνει η από κάτω αν αυτή υπάρχει. Επίσης βλέπουμε ότι υπάρχουν κόκκινα tabs με ονόματα τραπεζιών. Αυτό συμβαίνει γιατί ακόμα δεν έχουμε δει καθόλου τι περιέχουν αυτά τα τραπέζια μέσα και πρέπει να κάνουμε κλικ πάνω τους για να δούμε άμεσα. Μάλιστα τα συγκεκριμένα tabs τα οποία είναι κόκκινα στη εικόνα αναβοσβήνουν έντονα για να τραβήξουν τη προσοχή του σεφ ή του μπάρμαν.

Όταν πατηθούν έστω και μία φορά τότε σταματούν να αναβοσβήνουν και παίρνουν τη μορφή που έχει το τραπέζι T13 στην εικόνα. Κάθε καινούρια παραγγελία που έρχεται για ένα τραπέζι είτε το τραπέζι υπάρχει ήδη στον πίνακα των tabs είτε όχι αναγκάζει το tab να ξεκινήσει να αναβοσβήνει εκ νέου, εκτός και αν είμαστε ήδη σε εκείνο το tab και εκτελούμε την υπάρχουσα παραγγελία. Το πράσινο φόντο ενεργοποιείται όταν για μια γραμμή του πίνακα των προϊόντων πατηθεί το DONE και απενεργοποιείται όταν πατηθεί το Undo. Τέλος όταν από μια παραγγελία πατηθούν όλα τα OK είτε μαζί είτε ένα ένα τότε η παραγγελία θεωρείται ότι ήρθε εις πέρας και το tab για το συγκεκριμένο

τραπέζι αποσύρεται και τη σειρά του παίρνει το επόμενο από αυτό τραπέζι με διαθέσιμη προς εκτέλεση παραγγελία αν υπάρχει. Παρακάτω θα δούμε τον κώδικα τον οποίο χρησιμοποιούμε στο chef.php, τον οποίο και γεμίζουμε με τη JQuery, ο οποίος μοιάζει στο layout με το manage.php.

```php
<?php
    session_start();
    require_once('../logic/app_logic.php');
    if(!loggedin(false)){
        header( 'Location: /presentation/login.php' ) ;
    }
    require_once('chef_header.php');
?>
<html xmlns="http://www.w3.org/1999/xhtml">

    <head>
        <script src="../js/jquery.js"></script>
        <script src="../js/chef.js"></script>
        <link rel="stylesheet" type="text/css" href="../css/tabs.css">
        <link rel="stylesheet" type="text/css" href="../css/chefstyle.css">

        <title>Point of sale</title>
    </head>

    <body>
        <div id="large_div">
            <div class="tabs">

                <ul class="tab-links"></ul>

                <div class="tab-content">

                </div>
            </div>
        </div>
    </body>
</html>
```

Εικόνα 66: chef.php

Περιλαμβάνουμε στον κώδικα αυτό το αρχείο JQuery.js, δυο αρχεία css για τη μορφοποίηση της σελίδας μας και στο body ένα μεγάλο div με δύο μικρότερα μέσα τα οποία συνθέτουν τα tabs για τα τραπέζια με τις παραγγελίες μέσα. Ο έλεγχος που κάνουμε στην αρχή του εγγράφου είναι για τις περιπτώσεις που τη σελίδα επισκέπτεται κάποιος σεφ ή κάποιος admin όπου θα επιστραφεί true και θα προχωρήσει η εκτέλεση του υπόλοιπου κώδικα, ενώ θα γίνει redirect στην αρχική σελίδα αν κάποιος άλλος προσπαθήσει να κάνει login.

4.2. ΛΕΙΤΟΥΡΓΙΚΟΤΗΤΑ (chef.js)

Η χρήση πινάκων αποτελεί επιτακτική ανάγκη στη δημιουργία ή αφαίρεση tabs τραπεζιών ή προϊόντων. Οι πίνακες που χρησιμοποιούμε είναι οι flashing_tables, tables, allTables, orders και allOrders.Οι πίνακες allTables και allOrders είναι προσωρινοί πίνακες και περιέχουν τα δεδομένα των τραπεζιών και των παραγγελιών αντίστοιχα τα οποία έχουν μόλις επιστραφεί από την κλήση του chef_json.php του οποίου το αποτέλεσμα είναι το εξής.

{"act_table_order":[{"table":"T12","order_id":"14","m_type":"FRAPPE","s_type":"SWEET MILK","sp_desc":""},
{"table":"T12","order_id":"15","m_type":"BLACK TEA","s_type":"HONEY","sp_desc":""},
{"table":"T5","order_id":"21","m_type":"FRAPPE","s_type":"MEDIUM","sp_desc":""},
{"table":"T7","order_id":"22","m_type":"CARIB","s_type":null,"sp_desc":"no glass"}]}

Αυτό είναι ένα δείγμα του αποτελέσματος διότι το πραγματικό αποτέλεσμα θα ήθελε σελίδες για να το περιγράψουμε. Με βάση το αποτέλεσμα αυτό γεμίζουμε τους προσωρινούς μας πίνακες και στη συνέχεια τους συγκρίνουμε με τους ήδη τοπικά υπάρχοντες. Εάν βρούμε αλλαγές τότε εισάγουμε τα νέα πεδία στους κυρίως πίνακες μας και αφαιρούμε από τους κυρίως πίνακες μας τραπέζια που ίσως λείπουν από το αποτέλεσμα του json. Έτσι έχουμε πάντα ενημερωμένη την οθόνης μας κάθε 10 δευτερόλεπτα. Στη συνέχεια “ζωγραφίζουμε” καινούρια tabs για τραπέζια που δεν είχαμε πριν στους οριστικούς μας πίνακες, και γραμμές μέσα σε αυτά με τα προϊόντα.

Επίσης διαγράφουμε tabs που ίσως έχουν αφαιρεθεί από άλλους χρήστες ενώ εμείς ετοιμάζαμε κάποιες άλλες παραγγελίες. Δεν δημιουργούμε εκ νέου όλα τα tabs από την αρχή ούτε όλες τις γραμμές που περιέχονται σε αυτά. Αυτό θα ήταν αργό και καταστροφικό σε περιπτώσεις μεγάλου αριθμού τραπέζιων με πολλά προϊόντα ανά τραπέζι. Δημιουργούμε και διαγράφουμε τα tabs και τα προϊόντα μέσα σε αυτά, τα οποία δεν τα είχαμε στο πριν το νέο refresh των 10 δευτερόλεπτων. Ο παρακάτω κώδικα εκφράζει όλα τα παραπάνω.

```
    if (orders.indexOf(orderID)==-1){
        orders.push(orderID);

        if (tables.indexOf(tableName)!=-1){

            newOrder(json.act_table_order[i].table,
                json.act_table_order[i].m_type,
                json.act_table_order[i].s_type,
                json.act_table_order[i].sp_desc,
                json.act_table_order[i].order_id);
        }else{
            newTable(tableName);

            newOrder(json.act_table_order[i].table,
                json.act_table_order[i].m_type,
                json.act_table_order[i].s_type,
                json.act_table_order[i].sp_desc,
                json.act_table_order[i].order_id);

            tables.push(tableName);
        }
    }
}

function refreshDeleted(){

    //delete orders that were not received
    for (var i=0;i<orders.length;i++){
        if (allOrders.indexOf(orders[i])==-1){
            deleteOrder(orders[i]);
            orders.splice(i,1);//delete element
            i--;//current elemenu deleted
        }
    }

    //delete tables that were ot received
    for (var i=0;i<tables.length;i++){
        if (allTables.indexOf(tables[i])==-1){
            deleteTable(tables[i]);
            tables.splice(i,1);//delete element
            i--;//current elemenu deleted
        }
    }
}
```

Εικόνα 67: Γέμισμα -άδειασμα πινάκων με ενημερωμένα data

Πριν αντιγράψουμε τα αποτελέσματα στους κυρίως πίνακες δημιουργούμε όπως είπαμε παραπάνω καινούρια tabs και γραμμές. Μια άλλη δουλειά που κάνουμε είναι, για τα τραπέζια τα οποία είναι καινούρια είτε παλιά αλλά περιέχουν καινούρια προϊόντα, να αναβοσβήνουν τα tabs. Έτσι ελέγχουμε αν τα αποτελέσματα υπάρχουν στον πίνακα flashingTables και αν όχι αφού τα

δημιουργήσουμε καλούμε την flashTab(tableName) με παράμετρο τον αριθμό του τραπεζιού. Παρακάτω βλέπουμε πως:

```
function newOrder(tableName,m_type,s_type,sp_desc,order_id){

    if (s_type==null){
        s_type = '';
    }

    $('#tab'+ tableName).append('<div id="order' + order_id + '"

    if (flashingTabs.indexOf(tableName)==-1){

        flashingTabs.push(tableName);
        flashTab(tableName);
    }
}
```

Εικόνα 68: Δημιουργία παραγγελίας και εκκίνηση flashTab()

Με τον ίδιο τρόπο δημιουργείται και ένα καινούριο τραπέζι. Αρκεί ο κώδικας παραπάνω για να ξεκινήσει ένα tab να αναβοσβήνει αφού εκτελείται για κάθε νέο table, εφ' όσον το τραπέζι έχει μέσα τουλάχιστον ένα νέο order και για κάθε παλιό tab που έχει μέσα ένα νέο order, εκτός αν το παλιό tab που θα έρθει το νέο order είναι αυτό που είμαστε ήδη μέσα και το κοιτάμε. Τότε θα δούμε μόνο την αλλαγή στις γραμμές των προϊόντων. Η δημιουργία νέων tab και γραμμών για τα orders γίνεται με τη χρήση της append η οποία προσθέτει στο τέλος της λίστας μας το κατάλληλο στοιχείο έτοιμο και μορφοποιημένο. Αυτό φαίνεται στην παραπάνω εικόνα επίσης.

Πατώντας το κουμπάκι Done θεωρούμε ότι έχουμε ολοκληρώσει την εκτέλεση μιας παραγγελίας αλλά το μόνο που κάνουμε είναι η εμφάνιση του κουμπιού OK για την επιβεβαίωση, και ο χρωματισμός του φόντου της γραμμής σε πράσινο. Αν και αυτό ακούγεται αργό σε περιπτώσεις πίεσης, είναι οι περιπτώσεις αυτές που μπορεί να γίνει εύκολα το λάθος. Έτσι χρησιμοποιούμε αυτό τον τρόπο σαν ένα μικρό μέτρο ασφάλειας. Και εδώ θα δούμε τον τρόπο με τον οποίο το πετυχαίνουμε αυτό με τη χρήση της JQuery με append.

```
$('.done').off();

$('.done').click(function(event){

    var orderDiv = $(this).parent();

    var okchild = $(orderDiv).children('input.ok');

    if (okchild.length == 0 ){
        $(orderDiv).css('background-color','#9AFF9A');
        $(this).val('Undo');
        $(orderDiv).append('<input class="ok" type="button" value="OK" style="font-size: 1

        okchild = $(orderDiv).children('input.ok');

        $(okchild).click(function(){

            var order_id = $(orderDiv).attr('id').substring(5);

            $.ajax({url:'chef_increase.php?order_id='+order_id,success:function(result){

                $(orderDiv).remove();
            }});
        });
    }else{
        $(this).val('Done');
        $(okchild).remove();
        $(orderDiv).css('background-color','');
    }
});
```

Εικόνα 69: Done and OK buttons click functions

4.3. ΕΝΗΜΕΡΩΣΗ ΠΙΝΑΚΩΝ ΒΑΣΗΣ (OK button)

Το service το οποίο φαίνεται ότι καλέσαμε με το πάτημα του κουμπιού OK έχει δύο βασικούς σκοπούς. Ο πρώτος είναι η αύξηση της τιμής AMOUNT στον πίνακα counters για τον chef που ετοίμασε την παραγγελία, και ο δεύτερος είναι η αλλαγή της τιμής DONE του πίνακα act_orders από μηδέν σε ένα. Στην πρώτη περίπτωση το update εκτελείται για τη λήψη στατιστικών στοιχείων για το ποιος χρήστης τύπου σεφ ετοίμασε τις περισσότερες παραγγελίες και ποιος τις λιγότερες. Αυτό το πετυχαίνουμε με τη μεταβλητή $_SESSION[‘email_address’] την οποία παίρνουμε από το Sign in του χρήστη.

Η δεύτερη περίπτωση είναι σαφώς σημαντικότερη από ένα report. Στη μεταβλητή DONE του πίνακα act_orders βασίζεται όλη η καλή λειτουργία της κουζίνας αφού, ότι βλέπει μπροστά του ο κάθε σεφ είναι οι τιμές του πίνακα όπου το DONE = 0. Με τον τρόπο αυτό εξασφαλίζουμε ότι καμία παραγγελία δεν θα γίνει δυο φορές αν πατηθεί εγκαίρως το OK από το χρήστη και επίσης

σε περίπτωση διακοπής λειτουργίας του συστήματος για τον οποιοδήποτε λόγο, δεν θα χαθούν δεδομένα για τα active orders. Το chef_increase.php κάνει αυτή τη δουλειά με τον εξής τρόπο.

```php
<?php
    session_start();
    require_once('../logic/app_logic.php');

    if (loggedin(false)){

        if (isset($_GET['order_id'])){
            $con = dbConnect ();

            $user = mysqli_real_escape_string ( $con, $_SESSION ['email_address'] );

            echo $_SESSION ['email_address'];

            executeQuery($con,"UPDATE COUNTERS SET AMOUNT = AMOUNT +1 WHERE C_NAME = '$user' ");

            $orderId = mysqli_real_escape_string ( $con, $_GET['order_id'] );

            executeQuery($con,"UPDATE act_orders set done='1' where order_id='$orderId'");
        }
    }
?>
```

Εικόνα 70: updates στο chef_increase.php

ΕΠΙΛΟΓΟΣ

Ο επίλογος για τις εφαρμογές που αφορούν τον προγραμματισμό σε php, JQuery, Javascript και SQL, ολοκληρώνεται με την εφαρμογή για τους σεφ. Η λειτουργία της απλή αλλά κατανοητή και ο κώδικας που την έφερε εις πέρας επίσης κατανοητός. Αυτό που πετύχαμε είναι η δημιουργία μιας εφαρμογής γρήγορης και μικρής, χωρίς την αναγκαία ανανέωση ολόκληρης της σελίδας για τα καινούρια events. Η γνώση που αποκτήθηκε σε τέτοιου είδους προγραμματισμό είναι σημαντική, μιας και παλαιότερη εμπειρία πάνω σε κάτι τέτοιο ήταν πολύ μικρή. Η οργάνωση που απαιτείται σε επιχειρήσεις με μεγάλο αριθμό πελατών απαιτεί μελλοντικά την υλοποίηση νέων functionalities για την διευκόλυνση περαίωσης του φόρτου εργασίας σε μικρό χρονικό διάστημα.

5. ΚΕΦΑΛΑΙΟ ΠΕΜΠΤΟ

JSON

ΕΙΣΑΓΩΓΗ

Το **JSON** (JavaScript Object Notation) είναι ένα ελαφρύ πρότυπο ανταλλαγής δεδομένων. Είναι εύκολο για τους ανθρώπους να το διαβάσουν και να το γράψουν. Είναι εύκολο για τις μηχανές να το αναλύσουν (parse) και να το παράγουν (generate). Είναι βασισμένο πάνω σε ένα υποσύνολο της γλώσσας προγραμματισμού JavaScript, Standard ECMA-262 Έκδοση 3η - Δεκέμβριος 1999. Το JSON είναι ένα πρότυπο κειμένου το οποίο είναι τελείως ανεξάρτητο από γλώσσες προγραμματισμού αλλά χρησιμοποιεί πρακτικές (conventions) οι οποίες είναι γνωστές στους προγραμματιστές της οικογένειας προγραμματισμού C, συμπεριλαμβανομένων των C, C++, C#, Java, JavaScript, Perl, Python, και πολλών άλλων. Αυτές οι ιδιότητες κάνουν το JSON μια ιδανική γλώσσα προγραμματισμού ανταλλαγής δεδομένων.[18]

Η επικοινωνία ανάμεσα στις Client εφαρμογές, δηλαδή την εφαρμογή σε Android για τους σερβιτόρους και την εφαρμογή σεφ για την κουζίνα, επιτυγχάνεται με τη χρήση JSON. Παρακάτω θα αναλύσουμε ορισμένα αρχεία php των οποίων τα αποτελέσματα, για όποια χρήση και αν προορίζονται, κωδικοποιούνται σε json και επιτυγχάνεται η επικοινωνία του Server με τους Clients. Για λόγους ομοιότητας των εργασιών που εκτελούνται μέσα σε ένα php αρχείο πριν γίνει η κωδικοποίηση σε json, θα αναλύσουμε 4 βασικά αρχεία, το userlogin_json.php, το act_orders_json.php, το categories_json.php και add_order_json.php.

5.1. ΛΕΙΤΟΥΡΓΙΑΣ ΤΑΥΤΟΠΟΙΗΣΗΣ (userlogin_json.php)

Βασική εργασία που οφείλει να εκτελεί ένα service που ευθύνεται για την επικοινωνία Clinet – Server είναι να επιστρέφει με μορφή json το αποτέλεσμα στον χρήστη που κάλεσε το service. Παρακάτω φαίνεται ο κώδικας από το αρχείο userlogin_json.php ο οποίο θα μας βοηθήσει να κατανοήσουμε τι ακριβώς συμβαίνει μέσα σε ένα service που επιστρέφει json.

```php
<?php
    require_once('../data/data.php');
    //$response = '';
    $con=dbConnect();
    $username = mysqli_real_escape_string($con,$_GET['username']);
    $password= mysqli_real_escape_string($con,$_GET['password']);

    // get the user you ask from users table
    $result = executeQuery($con,"SELECT * FROM USERS
    WHERE USERNAME = '$username' AND PASS='$password' AND ACC_TYPE IN (1,0)");

    $result2 = executeQuery($con,"SELECT * FROM USERS
    WHERE EMAIL = '$username' AND PASS='$password' AND ACC_TYPE IN (1,0)");

    $numrows = mysqli_num_rows($result);
    $numrows2 = mysqli_num_rows($result2);

    if ($numrows == 1 || $numrows2 == 1){
            $response['success'] = 1;
            //return user's name
            if ($numrows == 1){
                $result3 = executeQuery($con,"SELECT NAME FROM USERS
                            WHERE USERNAME = '$username' AND PASS='$password'");
                $row=mysqli_fetch_array($result3);
                $response['name']=$row['NAME'];
            }
            elseif ($numrows2 == 1){
                $result3 = executeQuery($con,"SELECT NAME FROM USERS
                            WHERE EMAIL = '$username' AND PASS='$password'");
                $row=mysqli_fetch_array($result3);
                $response['name']=$row['NAME'];
            }
        }else{
            $response['success'] = 0;
        }

         echo json_encode($response);
         //echo 'WELCOME '. $row['NAME'];
?>
```

Εικόνα 71: userlogin_json.php

Το link που καλείται να καλέσει η εφαρμογή από το Android είναι του τύπου localhost/json/userlogin_json.php?username=abcd&password=1234. Έτσι λοιπόν μέσα σε php μπορούμε να διαχειριστούμε τα αποτελέσματα ερωτημάτων select προς τη βάση. Χρησιμοποιώντας και σε αυτή την περίπτωση τις μεταβλητές GET οι οποίες βασίζονται σε τιμές που μας έχει στείλει ο χρήστης στο link, επιστρέφουμε ένα string σε μορφή json. Αρχικά αν τα στοιχεία του χρήστη που εκτελεί το login, υπάρχουν στη βάση και ο

χρήστης είναι είτε διαχειριστής είτε σερβιτόρος, επιστρέφουμε στην εφαρμογή ένα string “success” : 1 και το όνομά του από τη βάση δεδομένων. Τοποθετούμε όλα τα στοιχεία που θεωρούμε απαραίτητα μέσα σε έναν πίνακα, στη συγκεκριμένη περίπτωση τον πίνακα $response[ ], και εκτελούμε την εντολή echo json_encode($response) .Αν τώρα τρέξουμε το αρχείο αυτό στον browser μας θα πάρουμε το εξής αποτέλεσμα:

{"success":1,"name":" Antonis"}

Αυτό που θα παρατηρήσουμε εδώ είναι ότι το encode δούλεψε και επέστρεψε “success” και τον integer 1 όπως και το όνομα του χρήστη που έκανε login σε μορφή string. Από τη μεριά του προγραμματιστή στο Android αυτό θα πρέπει να γίνει parse και να χρησιμοποιηθούν οι τιμές αυτές για να ολοκληρωθεί το login.

5.2. ΕΠΙΣΤΡΟΦΗ ΠΑΡΑΓΓΕΛΙΑΣ (act_orders_json.php)

Τα πράγματα θα γίνουν λίγο πολύπλοκα από δω και πέρα διότι η επιστροφή data από τον πίνακα act_orders της βάσης μας δεν είναι τόσο απλή. Ο χρήστης του POS χρειάζεται αυτό το service για να ξέρει όποτε το θελήσει την ενεργή παραγγελία κάποιου πιασμένου τραπεζιού. Αυτό συνεπάγεται το γεγονός, ότι χρειάζεται αναλυτικά δεδομένα για κάθε προϊόν (m_type) του πίνακα συνδυασμένα με τα submenu για κάθε order_id (s_types), αλλά και τη συνολική τιμή όλων αυτών αφού για κάθε m_type είναι πολύ πιθανό να υπάρχουν s_type τα οποία χρεώνονται, όπως για παράδειγμα το παγωτό που ενώ σαν m_type δεν έχει τιμή, παίρνει τη τιμή του αναλόγως με τον s_type το οποίο αντιστοιχεί σε μια, δυο τρείς ή παραπάνω μπάλες παγωτού. Το ερώτημα προς τη βάση γίνεται περίπλοκο αλλά η δυσκολία δεν σταματά εκεί.

ORDER_ID ▲ 1	M_TYPE	S_TYPE	SP_DESC	S_PRICE	M_PRICE
16	GREEN TEA	SUGAR		0	2.5
16	GREEN TEA	MILK		2	2.5
17	FREDDO ESPRESSO	MEDIUM		0	3

Εικόνα 72: Αποτέλεσμα ερωτήματος

Η βάση δεν μπορεί να μας επιστρέψει τα δεδομένα όπως τα χρειαζόμαστε αλλά μπορεί να μας επιστρέψει πληροφορία χρήσιμη την οποία εμείς μπορούμε να διαχειριστούμε με την php. Στο συγκεκριμένο παράδειγμα ζητήσαμε από τη βάση να μας επιστρέψει τα περιεχόμενα του τραπεζιού T4. Αυτό που παίρνουμε είναι ο παραπάνω πίνακας. Έτσι καλούμαστε να συνθέσουμε ένα string το οποίο θα περιέχει το order_id, το m_type με όλα του τα s_types και το price. Η επεξήγηση του κώδικα που ακολουθεί θα λύσει το γρίφο αυτό.

```
while ($row=mysqli_fetch_array($result)) {

    $tempid = $row['ORDER_ID'];
    $tempmtype = $row['M_TYPE'];
    $tempstype = $row['S_TYPE'];
    $temp_price = $row['M_PRICE'];
    $tempspdesc = $row['SP_DESC'];

    //if multiple submenus for this order id
    if ($tempid == $curid){

        $temp['s_type'] = $temp['s_type'] . ',' . $tempstype;

    }else{

        if ($curid != -1){//if not first iteration

            //add to final array
            array_push($response['act_table_order'], $temp);
        }

        //temporarily save new data
        $temp['order_id'] = $tempid;
        $temp['m_type'] = $tempmtype;
        $temp['s_type'] = $tempstype;
        $temp['price'] = (double)$temp_price;
        $temp['sp_desc']= $tempspdesc;

        $curid = $tempid;
    }
}
//insert last row
array_push($response['act_table_order'], $temp);

echo json_encode($response);
```

Εικόνα 73: parsing αποτελέσματος select

Χρησιμοποιώντας ένα while παίρνουμε όλες τις τιμές που επιστρέφει το select και τις αποθηκεύουμε σε προσωρινές μεταβλητές. Αρχικά έχουμε ορίσει ένα $curid=-1 για να κάνουμε push τα δεδομένα μας σε περίπτωση που οι επαναλήψεις είναι περισσότερες από μια. Έπειτα αποθηκεύουμε σε προσωρινές μεταβλητές τα πεδία του select που μας ενδιαφέρουν και τα επιστρέφουμε κάνοντας όμως concatenate για όλα τα s_types ενός order. Δεν πραγματοποιείται υπολογισμός της τιμής συνολικά του m_type και των s_type για ένα order, διότι ο πίνακας act_orders, περιέχει ήδη το σωστό αποτέλεσμα όπως θα δούμε στο επόμενο κεφάλαιο που αφορά τις ενέργειες του waiter. Ουσιαστικά επιστέφουμε σε όποιον καλέσει αυτό το service, ένα string τύπου json για το τραπέζι που ζητήθηκε το οποίο είναι το εξής:

{"act_table_order":[{"order_id":"16","m_type":"GREENTEA","s_type":"SUGAR,MILK", "price":2.5,"sp_desc":""}, {"order_id":"17","m_type":"FREDDO ESPRESSO","s_type":"MEDIUM","price":3,"sp_desc":""}]}

Το casting που γίνεται στον κώδικα στην περίπτωση του price είναι για λόγους ευκολίας του client να κάνει προσθέσεις αφαιρέσεις και συγκρίσεις κατ' ευθείαν χωρίς να χρειαστεί περαιτέρω ενέργεια από μεριάς του. Φαίνεται στο παραπάνω κείμενο ότι η τιμή της ετικέτας price δεν είναι μέσα σε διπλά εισαγωγικά αλλά σκέτη.

5.3. ΕΠΙΣΤΡΟΦΗ ΟΛΟΚΛΗΡΟΥ ΤΟΥ MENU (categories_json.php)

Κάποιες φορές η ανάγκη για ταχύτητα στη εφαρμογή, από μεριάς client είναι σημαντική για την γρηγορότερη εξυπηρέτηση του πελάτη. Η επικοινωνία μεταξύ των συσκευών είναι συνήθως ασύρματη. Αυτό από μόνο του εγκυμονεί κινδύνους για το λόγο ότι μπορεί αν και σε μικρή απόσταση να έχουμε προβλήματα με τα πολλά ασύρματα δίκτυα που καταλαμβάνουν τον ίδιο χώρο πόσο μάλλον αυτά που καταλαμβάνουν το ίδιο κανάλι εκπομπής. Έτσι δημιουργήσαμε ένα service το οποίο κάνει πολλές δουλειές για εμάς φτάνει να το καλέσουμε μόνο μια φορά. Στην περίπτωση που θέλουμε να προσθέσουμε μια καινούρια παραγγελία σε ένα τραπέζι, τότε πρέπει να καλέσουμε το categories_json.php. Αυτό το service μπορεί να επιστρέψει με μια κλήση όλο

το μενού του μαγαζιού με όλες τις λεπτομερείς παραμέτρους του. Έτσι λοιπόν αν και ο client θα πάρει ένα μεγάλο string με ένα δυσδιάστατο πίνακα στο εσωτερικό του, που έχει άλλους δυσδιάστατους πίνακες στο εσωτερικό του, μπορεί εύκολα να το κάνει parse τοπικά, να εμφανίσει τις σωστές πληροφορίες και να συνθέσει μια παραγγελία κάνοντας μόνο ένα request στη βάση. Το service αυτό, επιστρέφει το εξής κείμενο για την πρώτη κατηγορία προϊόντων δηλαδή για cat_id = 1

{"cat_id":1,"cat_desc":"COFFEE","mtypes":
[{"m_id":1,"m_type":"FREDDO ESPRESSO","amount":null,"m_price":3},
{"m_id":2,"m_type":"FREDDO CAPPUCCINO","amount":null,"m_price":3.5},
{"m_id":3,"m_type":"ESPRESSO","amount":null,"m_price":2},
{"m_id":4,"m_type":"CAPPUCCINO","amount":null,"m_price":3.5},
{"m_id":5,"m_type":"FRAPPE","amount":null,"m_price":2.5},
{"m_id":6,"m_type":"GREEK","amount":null,"m_price":2},
{"m_id":7,"m_type":"GREEKDOUBLE","amount":null,"m_price":2.5}],
"s_types":[{"s_id":1,"s_type":"SWEET","s_price":0},
{"s_id":2,"s_type":"MEDIUM","s_price":0},{"s_id":3,"s_type":"BLACK","s_price":0},
{"s_id":4,"s_type":"MILK","s_price":2},
{"s_id":21,"s_type":"CHOCOLATE SYRUP","s_price":0},
{"s_id":22,"s_type":"CARAMEL SYRUP","s_price":0},
{"s_id":24,"s_type":"CINNAMON","s_price":0},
{"s_id":33,"s_type":"BLACK SUGAR","s_price":0},
{"s_id":34,"s_type":"1 SACCHARIN","s_price":0},
{"s_id":35,"s_type":"2 SACCHARIN","s_price":0},
{"s_id":36,"s_type":"3 SACCHARIN","s_price":0}]}

Για την κατηγορία COFFEE λοιπόν έχουμε όλα m_id που της ανήκουν από τον πίνακα menu με τις τιμές τους και τις ποσότητες του στο cellar, όπως επίσης και όλα s_id, δηλαδή τις διαθέσιμες επιλογές για κάθε έναν καφέ, με τις τιμές τους επίσης. Αν αναλογιστούμε ότι σε ένα κατάστημα οι κατηγορίες προϊόντων μπορεί να είναι πάρα πολλές καταλαβαίνουμε ότι το μέγεθος του συγκεκριμένου string μπορεί να είναι τεράστιο, όμως η κλήση του json γίνεται μόνο μια φορά και το parsing από τη μεριά του χρήστη πολύ γρηγορότερα από το να καλούσε τρία διαφορετικά services για cat_id, m_id, s_id. Ο

κώδικας με τον οποίο πετυχαίνουμε αυτό είναι ο εξής:

```
$result = executeQuery($con,"SELECT * FROM CATEGORIES");

$response['categories'] = array();

$response['success'] = 0;
$response2['mt_type'] = array();
$product = array();

while ($row=mysqli_fetch_array($result)) {
    // temp user array

    $product ['cat_id'] = (double)$row ['CAT_ID'];
    $cat_id=$product ['cat_id'];
    $product['cat_desc'] = $row['CAT_DESC'];
    $product['mtypes']=array();
    $product['s_types']=array();
    $result2 = executeQuery($con,"SELECT M.M_ID,M.CAT_ID,M.M_TYPE, M.PRICE,
                    C.AMOUNT FROM MENU M LEFT JOIN CELLAR C ON
                    C.PRODUCT_TYPE = M.M_TYPE WHERE M.CAT_ID=".$row['CAT_ID']);

    while ($row2=mysqli_fetch_array($result2)) {
        // temp user array
        $product1['m_id'] = (int)$row2['M_ID'];
        $product1['m_type'] = $row2['M_TYPE'];
        $product1['amount'] = $row2['AMOUNT'];
        $product1['m_price'] = (double)$row2['PRICE'];
        // push single product into final response array

        array_push($product['mtypes'], $product1);

    }
    $result3 = executeQuery($con,"SELECT * FROM SUB_MENU WHERE CAT_ID=".$row['CAT_ID']);

    while ($row3=mysqli_fetch_array($result3)){
            $product2['s_id'] = (int)$row3['S_ID'];
            $product2['s_type'] = $row3['S_TYPE'];
            $product2['s_price'] = (double)$row3['PRICE'];

            array_push($product['s_types'], $product2);
    }

//some products found
$response['success'] = 1;

// push single product into final response array
array_push($response['categories'], $product);
}
```

Εικόνα 74: categories_json.php

Έχουμε έναν πίνακα ο οποίος έχει τα πεδία cat_id, cat_desc, mtypes, s_types, στους οποίους το τρίτο και τέταρτο πεδίο είναι δυσδιάστατοι πίνακες. Έχουμε χρησιμοποιήσει τρία while για αυτό το σκοπό, ένα εξωτερικό στον οποίο συνθέτουμε αυτά τα τέσσερα πεδία του πίνακα που επιστρέφεται και δύο εσωτερικά του τα οποία επιστρέφουν τα m_type και τα s_types με τις

τιμές τους για το συγκεκριμένο cat_id. Τα τρία ερωτήματα δεν είναι πολύπλοκα σε αυτή την περίπτωση και έτσι εκτελούνται σχεδόν ακαριαία από τη MySQL.

5.4. ΠΡΟΣΘΗΚΗ ΝΕΑΣ ΠΑΡΑΓΓΕΛΙΑΣ (add_order_json.php)

Σε αυτό το κεφάλαιο θα γίνει ανάλυση του τρόπου με τον οποίο γίνεται η αποκωδικοποίηση ενός JSON που έχει σταλεί από τον Client στον server για να προσθέσει μια παραγγελία. Θα δείξουμε πως παίρνουμε τη πληροφορία από τον client και πως τη διαχειριζόμαστε. Σε όλες τις προηγούμενες περιπτώσεις διαχειριζόμαστε μόνο το GET από το url. Τώρα πραγματοποιείται POST από τον Client, ένα JSON το οποίο περιλαμβάνει ένα πίνακα με τέσσερα πεδία.

{"order":[{"spec_desc":"","sid":[56],"price":4,"mid":12},{"spec_desc":"milk","sid":[56],"price":5,"mid":14},{"spec_desc":"","sid":[34,35,36,3,33],"price":3,"mid":2},{"spec_desc":"","sid":[66],"price":5,"mid":43}],"waiter_id":1,"table_name":"T7"}

Το πρώτο πεδίο είναι το special description του προϊόντος της παραγγελίας, το δεύτερο είναι ένας πίνακας μετά s_id της παραγγελίας, το τρίτο είναι το συνολικό ποσό της παραγγελίας, και τελευταίο πεδίο το προϊόν της παραγγελίας. Η σειρά με την οποία έρχονται τα προϊόντα δεν παίζει κανένα ρόλο από καμία πλευρά της επικοινωνίας.

Στο service καλούμαστε να κάνουμε parse τον πίνακα αυτό και να τοποθετήσουμε τα προϊόντα στους κατάλληλους πίνακες. Αρχικά, παίρνουμε τα περιεχόμενα του php://input και τα αποθηκεύουμε σε μια μεταβλητή. Στη συνέχεια κάνουμε decode αυτή τη μεταβλητή για να πάρουμε το JSON που χρειαζόμαστε σε έναν πίνακα. Με μια for σβανάρουμε τον πίνακα και τοποθετούμε στον act_orders τα πεδία mid, specialDesc, waiterId, table_num και price. Για να περάσουμε στη βάση τα sid για κάθε order πραγματοποιούμε ένα ακόμη for, εσωτερικό στο προηγούμενο, μέσα στο οποίο για το τρέχων order_id περνάμε όλα sid που υπάρχουν στον s_type_act. Τέλος στέλνουμε πίσω ένα response [‘success’] = 1 για να δείξουμε ότι όλα πήγαν καλά και ότι όλα μπήκαν στη βάση μας. Ο παρακάτω κώδικας είναι αυτός που δίνει ζωή στον client για να αλληλεπιδράσει με τη βάση και κατά συνέπεια δίνει ζωή στο POS.

```php
<?php
    require_once('../data/data.php');

    $con=dbConnect();

    $request = file_get_contents('php://input');

    $input = json_decode($request,true);

    $orders = $input['order'];
    $table = mysqli_real_escape_string($con,$input['table_name']);
    $waiterId = mysqli_real_escape_string($con,$input['waiter_id']);

    //insert each order
    for ($i=0;$i<count($orders);$i++){

        $mid = mysqli_real_escape_string($con,$orders[$i]['mid']);
        $specialDesc = mysqli_real_escape_string($con,$orders[$i]['spec_desc']);
        $sid = $orders[$i]['sid'];

        $price = mysqli_real_escape_string($con,$orders[$i]['price']);

        executeQuery($con,"insert into act_orders(M_ID,SP_DESC,PRICE,U_ID,TABLE_NUM)
                           values('$mid','$specialDesc','$price','$waiterId','$table')");

        $lastId = mysqli_insert_id($con);

        for ($k=0;$k<count($sid);$k++){
            executeQuery($con,"insert into s_type_act values('$lastId','".$sid[$k]."')");
        }
    }

    $response = array();
    $response['success'] = 1;

    echo json_encode($response);
?>
```

Εικόνα 75 : add_order_json.php

ΕΠΙΛΟΓΟΣ

Συνολικά τα services που χρησιμοποιήθηκαν για την καλή λειτουργία του POS είναι οχτώ, εμείς όμως επιλέξαμε να αναλύσουμε πλήρως τη λειτουργία των βασικότερων services που κάνουν βασικές δουλειές με διαφορετικό τρόπο το καθένα για να δείξουμε τις περιπτώσεις τις οποίες μπορεί κάποιος να συναντήσει σε μια ανταλλαγή δεδομένων τύπου Client – Server.

Συμπερασματικά, καταλαβαίνουμε ότι το συγκεκριμένο πρότυπο ανταλλαγής δεδομένων είναι εύκολο στη χρήση και κατανοητό από πολλές γλώσσες προγραμματισμού. Τα αποτελέσματα του δεν είναι μεγάλα σε μέγεθος καθώς περιλαμβάνουν απλό κείμενο και ενδείκνυται για περιπτώσεις ασύρματης επικοινωνίας όπως η δική μας.

Τέλος, με την php μπορούμε να διορθώσουμε τις αδυναμίες που παρουσιάζονται από τα αποτελέσματα της βάσης δεδομένων μας και έτσι μπορούμε να στείλουμε στον client την πληροφορία που χρειάζεται.

6. ΚΕΦΑΛΑΙΟ ΠΕΜΠΤΟ

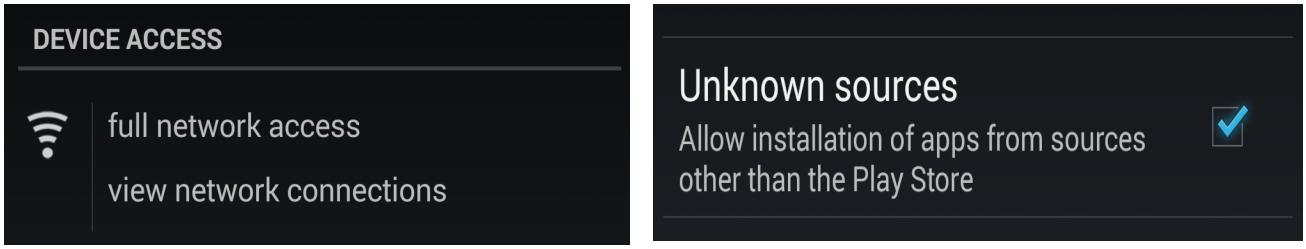
POS (Εφαρμογή Android)

ΕΙΣΑΓΩΓΗ

Στο κεφάλαιο αυτό θα παρουσιαστεί η πλειονότητα των μεθόδων και των κλάσεων που χρησιμοποιούνται για την υλοποίηση του Client σε Android. Θα παρουσιαστούν τα επιμέρους τμήματα κώδικα για την καλύτερη κατανόηση του προγράμματος καθώς και επεξηγήσεις για το τι κάνει και που χρησιμοποιείται το κάθε block κώδικα από αυτά. Με τα απαραίτητα screenshots μέσα από την εφαρμογή, θα δείξουμε πώς συνθέτουμε κάθε οθόνη με την XML και θα αναλύσουμε τη λειτουργικότητα της.

Απαραίτητη προϋπόθεση για την εγκατάσταση της εφαρμογής αποτελεί η ύπαρξη μια συσκευής με λογισμικό Android και κατά προτίμηση έως 6”. Θεωρούμε αυτό το μέγεθος ιδανικό για την γρηγορότερη και άμεση πρόσβαση του χρήστη στις επιλογές του προγράμματος. Μία ακόμα παράμετρος για την εγκατάσταση της εφαρμογής είναι η ενεργοποίηση της επιλογής εγκατάστασης εφαρμογών από άγνωστες πηγές, και η αποδοχή του δικαιώματος πρόσβασης σε Wi-Fi από την εφαρμογή. Το αρχείο εγκατάστασης ονομάζεται POS.apk, και για να λειτουργήσει κανονικά το πρόγραμμα πρέπει να έχει γίνει σύνδεση στο ίδιο δίκτυο με αυτό που βρίσκεται ο server, μιας και η εφαρμογή τρέχει σε LAN. Η IP του server είναι σταθερή και είναι η 192.168.1.13.

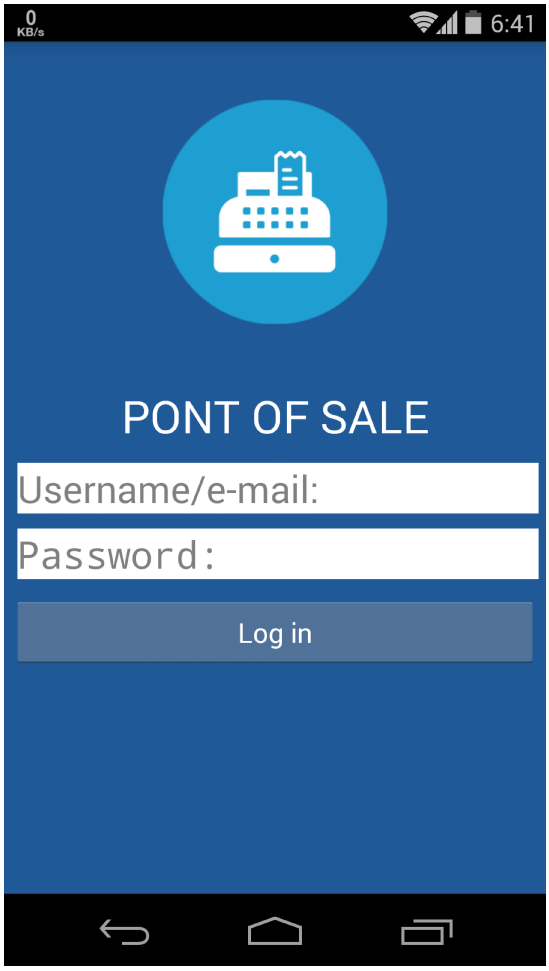
Για τη σύνθεση του κώδικα χρησιμοποιήθηκε το Eclipse με ADT, τόσο για τα αρχεία XML όσο και για τα αρχεία JAVA.[22]



Εικόνα 76: installation requirements

6.1. LOGIN SCREEN

Όπως σε όλα τα προγράμματα έτσι και εμείς χρησιμοποιούμε το δικό μας λογότυπο για την εφαρμογή μας το οποίο και έχουμε παρουσιάσει σε πολλές περιπτώσεις στα προηγούμενα κεφάλαια. Μετά την εγκατάσταση της εφαρμογή μας στο κινητό ή το tablet, ανοίγουμε την εφαρμογή και το πρώτο πράγμα που καλούμαστε να κάνουμε είναι να εισάγουμε τα στοιχεία μας για να κάνουμε Login στην εφαρμογή. Η οθόνη αυτή φαίνεται στην εικόνα 76.



Εικόνα 77 Login Screen

Μπορούμε είτε να εισάγουμε το όνομα χρήστη είτε το e-mail μας για την είσοδο στην εφαρμογή. Αν για κάποιο λόγο δεν έχουμε συνδεθεί με Wi-Fi σε κάποιο δίκτυο τότε θα πεταχτεί στο κάτω μέρος της οθόνης μας ένα toast με το κατάλληλο μήνυμα. Για να δημιουργηθεί η οθόνη αυτή έχουν χρησιμοποιηθεί ένα TextView το οποίο περιέχει τον τίτλο της εφαρμογής, ένα Imageview το οποίο περιέχει το λογότυπο της εφαρμογής, δύο EditText πεδία για την εισαγωγή Username και Password, ένα κουμπί Log in, και όλα αυτά

βρίσκονται μέσα σε LinearLayouts τα οποία μας βοηθούν να ξεχωρίσουμε τα στοιχεία που έχουμε στην οθόνη μας και να επεξεργαστούμε κάθε στοιχείο ξεχωριστά. Τα LinearLayouts θα τα χρησιμοποιήσουμε πολύ στο σύνολο της εφαρμογής μας αλλά δεν θα κουράσουμε με screenshots από κώδικα XML καθώς στις περισσότερες περιπτώσεις μοιάζει αρκετά με αυτόν της αρχικής οθόνης. Το ίδιο και τα υπόλοιπα στοιχεία αυτής της οθόνης. Θα γίνεται αναφορά στον κώδικα της XML μόνο όπου υπάρχει κάτι καινούριο να δείξουμε το οποίο δεν έχουμε αναφέρει.

```
<LinearLayout
    android:id="@+id/l1"
    android:layout_width="fill_parent"
    android:layout_height="0dp"
    android:layout_weight="0.4"
    android:layout_gravity="center"
    android:background="@color/poscolor"
    android:orientation="horizontal"
    android:gravity="center_vertical|center_horizontal">

    <ImageView
        android:id="@+id/logo"
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:background="@drawable/poslogo"
    />

</LinearLayout>
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="PONT OF SALE"
    android:textSize="30sp"
    android:textColor="@color/white"
/>
```

```
<EditText
    android:id="@+id/username"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Username/e-mail:"
    android:textSize="25sp"
    android:layout_marginBottom="10dp"
    android:background="@color/white"
    android:textColor="@color/black"
    android:imeOptions="actionNext"
    android:inputType="textEmailAddress"
/>

<Button
    android:id="@+id/loginbtn"
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:text="Log in"
    android:textColor="@color/white"
/>
```

Εικόνα 78: Μορφή στοιχείων οθόνης στην XML

Η διαχείριση για το κάθε κομμάτι γίνεται με τη χρήση JAVA. Τα αρχεία τα οποία δίνουν ζωή στο πρόγραμμα βρίσκονται στον κατάλογο src\com\pos\pos. το αρχείο Login_Activity.java είναι αυτό που θα αναλύσουμε σε αυτή τη παράγραφο.

Όλη η λειτουργικότητα της οθόνης υφίσταται μόλις δημιουργούνται τα αντικείμενα στην οθόνη με την μέθοδο onCreate(). Τέσσερις είναι οι μεταβλητές που χρειαζόμαστε σε αυτή την κλάση και είναι όλες private. Αντιπροσωπεύει η κάθε μία ένα στοιχείο της οθόνης, δηλαδή ένα κουμπί και δύο EditText. Παίρνουμε τις τιμές που έχει εισάγει ο χρήστης με τη μέθοδο getText() για το Username και το Password και τα εισάγουμε σαν

παραμέτρους στη μέθοδο login() της κλάσης JSONHelper η οποία παίρνει τα αποτελέσματα από τον server στη μορφή που εξηγήσαμε στον προηγούμενο κεφάλαιο και ελέγχει αν τα στοιχεία είναι αυτά που χρειάζεται, κάνοντας parse το string του αποτελέσματος. Αν επιστραφούν success : 1 και το όνομα του χρήστη που έκανε το log in τότε περνάει το πρόγραμμα στη δεύτερη οθόνη. Αν το αποτέλεσμα του json είναι success : 0 τότε θα πάρουμε το μήνυμα σε toast “Wrong username or password”. Τη μεταβλητή loggedInAs τη χρησιμοποιούμε για να αποθηκεύσουμε το όνομα του χρήστη για να εμφανίσουμε το κατάλληλο μήνυμα καλωσορίσματος στη δεύτερη οθόνη. Ο κώδικας που εκτελεί όλα τα παραπάνω είναι ο εξής.

```
login = (Button) findViewById(R.id.loginbtn);
username = (EditText) findViewById(R.id.username);
password = (EditText) findViewById(R.id.password);

login.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        ConnectivityManager connManager = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo mWifi = connManager.getNetworkInfo(ConnectivityManager.TYPE_WIFI);

        if (!mWifi.isConnected()) {
            Toast.makeText(getApplicationContext(), "Wifi is disabled! ", Toast.LENGTH_LONG).show();
        } else {

            try {
                login();
                Intent intent = new Intent(Login_Activity.this, Tables_Activity.class);
                intent.putExtra("waiter", loggedInAs);
                startActivityForResult(intent,1);

            }catch (POSException posE){
                Toast.makeText(getApplicationContext(), posE.getMessage(), Toast.LENGTH_LONG).show();
            } catch (Exception e) {
                Toast.makeText(getApplicationContext(), "Cannot connect to server!", Toast.LENGTH_LONG).show();
            }
        }
    }
});
    }
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {

         switch(requestCode) {
            case (1) : {
              if (resultCode == Activity.RESULT_OK) {

              int flag = data.getIntExtra("exit", 0);
              if (flag ==1) finish();
              // TODO Update your TextView.
              }
              break;
            }
         }


    private void login() throws Exception {
        Log.e("login", "login pressed");
        JSONHelper.setServerURL("http://192.168.1.13");
        loggedInAs = JSONHelper.login("" + username.getText(), "" + password.getText());

        // return !loggedInAs[1].equals("0");
    }
```

Εικόνα 79: Login_Activity.java

Ουσιαστικά με την putExtra(“waiter”, loggedInAs) περνάμε στη δεύτερη οθόνη δεδομένα με το αντικείμενο intent,.το οποίο περιγράφει μια λειτουργία η οποία πρόκειται να εκτελεστεί. Όχι μόνο περνάμε δεδομένα αλλά παίρνουμε κιόλας δεδομένα από την επόμενη οθόνη. Αυτό συμβαίνει σε περίπτωση που από τη δεύτερη οθόνη πατηθεί το πίσω ή το Logout όπου θα εξηγήσουμε στην επόμενη παράγραφο τι συμβαίνει. Αν πατηθεί το κουμπί πίσω τότε θα εκτελεστεί ο κώδικας της onActivityResult αυτόματα. Αυτή είναι προεπιλεγμένη ενέργεια από το Android. Στην πράξη, μόλις καλούμε τη δεύτερη οθόνη τότε διακόπτεται η λειτουργία της πρώτης και δεν συνεχίζεται η λειτουργία της μέχρι να έρθουμε με κάποιο τρόπο σε αυτή την οθόνη πάλι. Έτσι μόλις επιστρέψουμε στην πρώτη οθόνη από τη δεύτερη γίνεται ο έλεγχος με τις μεταβλητές για το ποιο κουμπί πατήθηκε, είτε το Logout είτε το πίσω. Αν έχει πατηθεί το πίσω τότε η εφαρμογή τερματίζει ενώ αν έχει πατηθεί το Logout η οθόνη περιμένει εκ νέου στοιχεία για Login.

```
public static Waiter login(String username, String password) throws Exception {
    String success = new String();
    Waiter waiter = new Waiter();
    url = urlpart + "/json/userlogin_json.php?username=" + username + "&" + "password=" + password;
    Log.e("login url", url);

    JSONObject json = new JSONParser().getJSONFromUrl(url);
    Log.e("JSON response", json.toString());

    success = json.getString("success");
    if (success.equals("0")) {
        throw new POSException("Wrong user name or password!");
    }
    waiter.setName(json.getString("name"));
    waiter.setId(json.getInt("u_id"));

    return waiter;
}
```

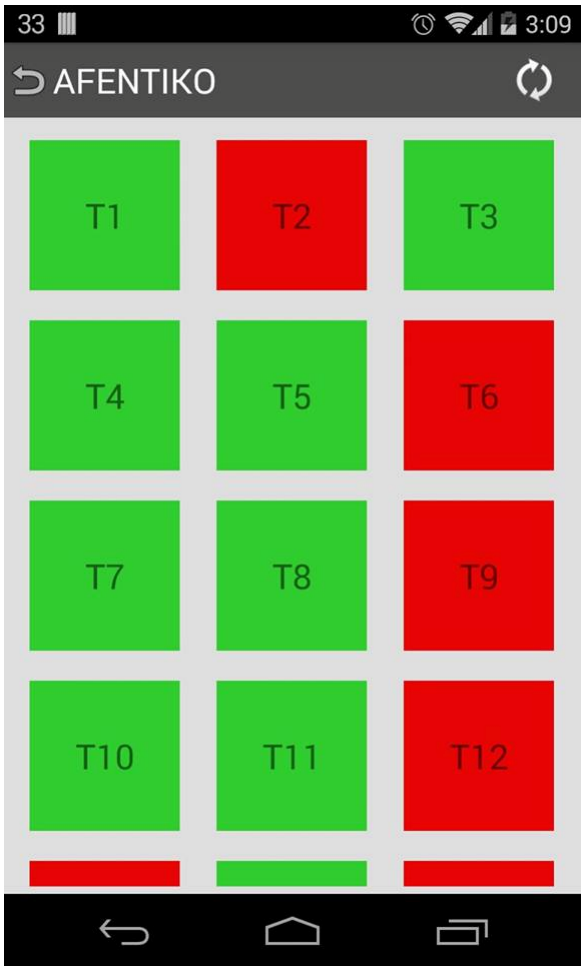
Εικόνα 80: login() της κλάσης JSONHelper

Η κλάση JSONHelper βρίσκεται σε ένα φάκελο ποιο «βαθιά» στο πρόγραμμα, στο φάκελο json. Εκεί δημιουργείται ένα αντικείμενο τύπου JSONObject το οποίο γεμίζει με τα αποτελέσματα της JSONParser (κεφ 6.6) μιας άλλης κλάσης η οποία βρίσκεται στον ίδιο φάκελο με τη JSONHelper. Οι τιμές που μας επιστρέφει η μέθοδος αυτή είναι 0 αν τα στοιχεία που εισήγαγε ο χρήστης δεν είναι σωστά και το όνομα του χρήστη με το id του αν τα στοιχεία είναι σωστά. Στη συνέχεια το όνομα και το id του χρήστη περνούν

σαν παράμετροι στο αντικείμενο waiter τύπου Waiter και επιστρέφονται. Αν πάρουμε 0 από τη JSONHelper τότε επιστρέφουμε ένα POSException με κείμενο “Wrong username or password!”.

6.2. TABLES STATE SCREEN

Μετά από επιτυχημένο login στην εφαρμογή οδηγούμαστε στη δεύτερη οθόνη του προγράμματος μας. Εκεί θα δούμε πολλά τετραγωνάκια που το καθένα αντιπροσωπεύει ένα τραπέζι του μαγαζιού. Μπορούμε να διακρίνουμε με πράσινο χρώμα τα τραπέζια τα οποία είναι ακόμα ελεύθερα στο μαγαζί και με κόκκινο χρώμα τα τραπέζια που είναι ακόμα απλήρωτα.



Εικόνα 81 : Table state screen

Όλοι οι σερβιτόροι που χρησιμοποιούν το Point of Sale μπορούν να είναι ενημερωμένοι για το ποια τραπέζια είναι πιασμένα και ποια ελεύθερα στο μαγαζί ανεξαρτήτου πόστου. Η συγκεκριμένη οθόνη με τα τραπέζια ενημερώνεται μόνη της κάθε δύο λεπτά και υπάρχει και η δυνατότητα

χειροκίνητης ενημέρωσης με το κουμπάκι Refresh που βρίσκεται στο πάνω δεξιά μέρος της οθόνης. Πατώντας το κουμπάκι με το όνομα του σερβιτόρου στο πάνω αριστερά μέρος της οθόνης πραγματοποιείται Logout. Με το πάτημα του κουμπιού “Πίσω” δεν πραγματοποιείται Logout από το πρόγραμμα, αλλά τερματισμός του προγράμματος. Πρέπει για να τερματίσουμε το πρόγραμμα να πατήσουμε το κουμπάκι δύο φορές όπως θα δούμε και στο σχετικό toast που θα εμφανιστεί.

6.2.1. ΓΕΜΙΣΜΑ ΟΘΟΝΗΣ

Με το άνοιγμα της οθόνης, αυτό που υπάρχει είναι ένα LinearLayout, το οποίο περιέχει ένα Button και ένα Imageview για το πάνω μέρος της οθόνης, και ένα GridView. Αυτά και μόνο τα δύο στοιχεία υπάρχουν αρχικά στην οθόνη μας. Με το άνοιγμα της οθόνης τρέχει η μέθοδος onCreate() όπως γίνεται σε κάθε οθόνη που ξεκινάει να εκτελείται. Μέσα εκεί ελέγχεται εάν υπάρχει αποθηκευμένο Instance των τραπεζιών και αν όχι τότε ξεκινάει η δημιουργία τους, μέσω της παραμέτρου τύπου Bandle της onCreate(). Αν σε περίπτωση γυρίσουμε την οθόνη μας οριζόντια από κάθετα ή το ανάποδο, τότε θεωρούμε ότι το Instance υπήρχε και δεν κάνουμε την ίδια διαδικασία, αλλά μόνο ζωγραφίζουμε τα τραπέζια στο GridView.

Με την εκκίνηση της onCreate ελέγχουμε τη λειτουργία των κουμπιών της οθόνης μας, για Logout ή Exit που θα αναλύσουμε στο υποκεφάλαιο 6.2.3 παρακάτω. Στη συνέχεια δημιουργούμε μια Timer μεταβλητή για την οποία κάνουμε Override την run(). Έτσι θα τρέξει η Timer_method() η οποία με τη σειρά της θα καλέσει την Runnable μεταβλητή Timer_tick η οποία κάνει όλη τη δουλειά από κει και κάτω. Μέσα σε αυτή τη μέθοδο συλλέγουμε την πληροφορία που μας χρειάζεται για τα τραπέζια που υπάρχουν στο μαγαζί, ποια είναι ελεύθερα ή όχι και το όνομα του κάθε τραπεζιού, με τη βοήθεια της JSONHelper κλάσης και πάλι, που θα τρέξει το service json/tables_json.php, το οποίο θα επιστρέψει ένα πίνακα με τα τραπέζια και ποια από αυτά είναι πιασμένα.

Η Timer_tick αφού γεμίσει ένα ArrayList με data από το JSON, τότε θα τρέξει την InitializeTablesGridView(), η οποία θα ζωγραφίσει τα τραπέζια στην οθόνη μας με το κατάλληλο χρώμα και όνομα. Παρακάτω βλέπουμε τον

κώδικα της Timer_tick και της InitializeTablesGridView().

```
public void InitializeTablesGridView() {

    gridView = (GridView) findViewById(R.id.TablesGridView);
    customGridAdapter = new TablesGridViewAdapter(this, R.layout.tables_row_layout, tables);
    gridView.setAdapter(customGridAdapter);

    gridView.setOnItemClickListener(new OnItemClickListener() {
        public void onItemClick(AdapterView<?> parent, View v, int position, long id) {

            Intent intent = new Intent(Tables_Activity.this, CurrentOrder_Activity.class);

            // pass the table name to the next activity
            intent.putExtra("tablename", ((Table) parent.getItemAtPosition(position)).getTitle());
            startActivity(intent);
        }

    });
private Runnable Timer_Tick = new Runnable() {

    @Override
    public void run() {
        try {
            tables = getData();
        } catch (Exception e) {
            Toast.makeText(getApplicationContext(),
            "Cannot connect to server! Tables did not refresh!!!",
                    Toast.LENGTH_LONG).show();
        }
        InitializeTablesGridView();
    }
};
```

Εικόνα 82: Timer_tick() και InitializeTablesGridView()

Αυτό που κάνει η μέθοδος αυτή είναι να δημιουργεί ένα δικό της customGridAdapter, με παραμέτρους το στοιχείο XML που θα χρησιμοποιήσει για να γεμίσει την οθόνη, και την πηγή των δεδομένων. Επίσης η συγκεκριμένη μέθοδος θα ελέγξει την ενέργεια του πατήματος ενός κουμπιού τραπέζιού σύμφωνα με το οποίο θα μεταφερθούμε στην επόμενη οθόνη μας παίρνοντας μαζί το όνομα του τραπεζιού το οποίο πατήθηκε. Αυτό φυσικά γίνεται αφού γεμίσει η οθόνη μας με τραπέζια πιασμένα, ελεύθερα ή και τα δύο. Η TablesGridViewAdapter() η οποία κάνει extend την ArrayAdapter<Table> είναι αυτή που πρέπει να κάνουμε Override για να πάρει τιμή τελικά η customGridAdapter που αναφέραμε πριν. Η συγκεκριμένη μέθοδος χρησιμοποιεί το XML της εικόνας 82 για να γεμίσει το GridView, το οποίο δεν είναι κάτι άλλο παρά ένα TextView. Την ποσότητα των τραπεζιών

που θα δημιουργηθούν την ξέρει η TablesGridViewAdapter από το ArrayList που της δίνουμε.

```
<TextView
    android:id="@+id/TableName"
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:layout_marginTop="5dp"
    android:gravity="center"
    android:textSize="20sp" >
</TextView>
```

Εικόνα 83: TextView τραπεζιού

Αυτό που πρέπει να κάνουμε πλέον είναι Override την μέθοδο getView() και χρησιμοποιώντας την getSystemService() να αποκτήσουμε πρόσβαση στο αντικείμενο τύπου LayoutInflater της οθόνης μας, για να γεμίσουμε το GridView. Το όνομα του κάθε τετραγώνου που θα εμφανιστεί είναι το όνομα του τραπεζιού και αυτό γίνεται με την getTitle() της μεταβλητής table_Name.

```
// custom GridViewAdapter implemented in an inner class for simplicity as
// it's going to only be used in this Activity
private class TablesGridViewAdapter extends ArrayAdapter<Table> {
    // private Context context;
    private int layoutResourceId;
    private ArrayList<Table> data = new ArrayList<Table>();

    public TablesGridViewAdapter(Context context, int layoutResourceId, ArrayList<Table> data) {
        super(context, layoutResourceId, data);
        this.layoutResourceId = layoutResourceId;
        // this.context = context;
        this.data = data;
    }

    private class ViewHolder {
        TextView tableName;
    }

    public View getView(int position, View convertView, ViewGroup parent) {
        View row = convertView;
        ViewHolder holder = null;

        if (row == null) {
            LayoutInflater inflater = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            row = inflater.inflate(layoutResourceId, parent, false);
            holder = new ViewHolder();
            holder.tableName = (TextView) row.findViewById(R.id.TableName);

            row.setTag(holder);
        } else {
            holder = (ViewHolder) row.getTag();
        }

        Table item = (Table) data.get(position);
        holder.tableName.setText(item.getTitle());

        // if (item.getAvailable())
        holder.tableName.setBackgroundColor
        (Color.parseColor(item.getAvailable() ? "#32cd32" : "#E60606"));
        // else
        // holder.tableName.setBackgroundColor(Color.parseColor("#E60606"));
        return row;
    }

}
```

Εικόνα 84: Δημιουργία customGridAdapter

Το χρώμα του background του κάθε TextView που εκφράζει ένα τραπέζι, θα γίνει κόκκινο αν το τραπέζι είναι πιασμένο, πράσινο αν όχι ή αν είναι πληρωμένο. Η setBackgroundColor() κάνει αυτή τη δουλειά στη τελευταία γραμμή της getView().

6.2.2. ΛΕΙΤΟΥΡΓΙΑ REFRESH

Κατά την εκκίνηση της οθόνης και μετά το γέμισμα της αυτό που γίνεται είναι η ανανέωση της οθόνης των τραπεζιών αυτόματα κάθε δύο λεπτά. Η χειροκίνητη ανανέωση της κατάστασης των τραπεζιών πραγματοποιείται με το κουμπάκι που βρίσκεται στο πάνω δεξιά μέρος της οθόνης. Ο χρόνος των δύο λεπτών έχει επιλεγεί τυχαία αν και θα μπορούσε να είναι μεγαλύτερος μιας και υπάρχει χειροκίνητο refresh. Το χειροκίνητο refresh της κατάστασης των τραπεζιών εκτελεί τον κώδικα της Timer_tick που εξηγήσαμε στη προηγούμενη παράγραφο. Ουσιαστικά καλείται το κατάλληλο service από τον server και στη συνέχεια η InitializeTablesGridView() που επαναδημιουργεί τα τραπέζια.

```
// create a new timer that refreshes the tables state every 2
// minutes
// tables = getData();
Timer refreshTimer = new Timer();
refreshButton = (ImageButton) findViewById(R.id.tablesRefreshButton);
refreshButton.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View v) {
        Timer_Tick.run();
    }
});
refreshTimer.schedule(new TimerTask() {

    @Override
    public void run() {
        // TODO Auto-generated method stub
        TimerMethod();
    }
}, 0, 120000);
```

Εικόνα 85: Αυτόματο και χειροκίνητο refresh τραπεζιών

6.2.3. ΛΕΙΤΟΥΡΓΙΑ ΕΞΟΔΟΥ – LOGOUT

Δύο είναι οι τρόποι με τους οποίους μπορούμε να βγούμε από την

εφαρμογή. Η διαφορά του Logout από την έξοδο όπως αναφέραμε και στην πρώτη παράγραφο αυτού του κεφαλαίου, είναι ότι με την έξοδο κλείνει το πρόγραμμα ενώ με το Logout εμφανίζεται η πρώτη οθόνη για εισαγωγή εκ νέου στοιχείων Login.

Πατώντας το κουμπί με το όνομα του χρήστη και το βελάκι στο πάνω αριστερά μέρος της οθόνης πραγματοποιείται το Logout. Αυτό που πραγματικά συμβαίνει στον κώδικα είναι ότι κάνουμε finish() τη συγκεκριμένη οθόνη χωρίς να περνάμε κάποια παράμετρο στο intent με αποτέλεσμα το android να μας πηγαίνει στην προηγούμενη οθόνη και να περιμένει.

Αντίθετα στην περίπτωση που πατηθεί δύο φορές το Πίσω τότε ναι μεν μεταφερόμαστε στο Login Screen αλλά περνάμε σαν παράμετρο στο αντικείμενο intent τη τιμή exit = 1. Η τιμή αυτή ελέγχεται (βλ. εικόνα 78) κατά τη συνέχεια εκτέλεσης της πρώτης οθόνης και τότε γίνεται finish() και η πρώτη οθόνη, αν οι τιμές των exit ταιριάζουν. Να τονίσουμε ότι πρέπει να πατηθεί το κουμπί Πίσω δύο φορές για να πραγματοποιηθεί exit από την εφαρμογή. Το κατάλληλο toast μας προτρέπει για κάτι τέτοιο. Η μέθοδος onBackPressed() που φαίνεται παρακάτω κάνει όλα όσα είπαμε παραπάνω.

```
private Boolean exit = false;

@Override
public void onBackPressed() {
    if (exit) {
        Intent back = new Intent(Tables_Activity.this, Tables_Activity.class);
        back.putExtra("exit", 1);
        setResult(Activity.RESULT_OK, back);
        finish();
    } else {
        Toast.makeText(this, "Press Back again to Exit.", Toast.LENGTH_SHORT).show();
        exit = true;
        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                exit = false;
            }
        }, 3 * 1000);

    }
}
```

Εικόνα 86: onBackPressed() στο tables_activity.java

6.3. CURRENT ORDER SCREEN

Πατώντας σε κάποιο τραπέζι της δεύτερης οθόνης μεταφερόμαστε αυτόματα στην τρίτη οθόνη του προγράμματος μας. Αν το τραπέζι που πατήσαμε ήταν ελεύθερο ή πληρωμένο, δηλαδή με πράσινο χρώμα, τότε θα δούμε μια κενή λίστα. Αν το τραπέζι που πατήσαμε ήταν αρχικά κόκκινο τότε θα δούμε όλα τα προϊόντα που περιέχει το συγκεκριμένο τραπέζι και δίπλα σε κάθε προϊόν ένα κουμπάκι info, μια κλεψύδρα ή ένα πράσινο tick αν το προϊόν έχει εκτελεστεί από κάποιον σεφ, και το ποσό που κοστίζει. Στο πάνω αριστερό μέρος υπάρχει η επιλογή All για να επιλεχθούν όλα τα checkbox μαζί για την παραγγελία, στο κέντρο φαίνεται το όνομα του τραπεζιού που πατήθηκε και στο δεξί μέρος το συνολικό ποσό της παραγγελίας. Στο κάτω μέρος έχουμε τοποθετήσει τρία κουμπιά. Το πρώτο από αριστερά είναι το Add το οποίο μπορεί να προσθέσει προϊόντα στην υπάρχουσα παραγγελία (επόμενη οθόνη). Αυτό θα μας μεταφέρει και στην επόμενη οθόνη την οποία θα αναλύσουμε στο επόμενο κεφάλαιο. Το δεύτερο κουμπί είναι το Delete με το οποίο μπορούμε να διαγράψουμε κάποιο προϊόν της παραγγελίας και το τρίτο είναι εκείνο με το οποίο μπορούμε να πληρωθούμε τα επιλεγμένα προϊόντα της λίστας.

Εικόνα 87: Current order screen

Το xml το οποίο γεμίζει αυτή την οθόνη σε αυτή τη περίπτωση είναι ένα ListView που είναι κενό αρχικά αλλά εν συνεχεία γεμίζει με γραμμές που περιέχουν κάποιο προϊόν με βάση του current_order_list_layout.xml. Το γέμιασμα κάθε γραμμής του LIstView γίνεται με τον παρακάτω τρόπο.

```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <CheckBox
        android:id="@+id/checkBox"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="2"
        android:singleLine="true"
        android:text="CheckBox"
        android:textSize="25sp" />

    <ImageButton
        android:id="@+id/infoButton"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:src="@drawable/info"
        android:background="@android:color/transparent"
        android:scaleType="fitCenter"
        android:adjustViewBounds="true"/>

    <ImageView
        android:id="@+id/infoDone"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:src="@drawable/not_done_yet"
        android:scaleType="fitCenter"
        android:adjustViewBounds="true"/>

    <TextView
        android:id="@+id/Price"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="right"
        android:text="Price"
        android:textAlignment="textEnd"
        android:textSize="22sp" />
</LinearLayout>
```

Εικόνα 88: ListView στο current_order_layout.xml

Περιέχουμε μέσα σε κάθε γραμμή ένα checkbox, ένα ImageButton που αν πατηθεί θα εμφανιστούν οι λεπτομέρειες του προϊόντος της παραγγελίας, ένα ImageView για την πληροφόρηση του σερβιτόρου αν έχουν εκτελεστεί από τους σεφ τα προϊόντα της παραγγελίας και ένα text view με το ποσό που κοστίζει το προϊόν της παραγγελίας.

6.3.1. ΓΕΜΙΣΜΑ ΟΘΟΝΗΣ

Στην onCreate() η οποία τρέχει με το άνοιγμα της οθόνης, ελέγχουμε και πάλι το instance, όπως και στην προηγούμενη οθόνη, και σε περίπτωση που δεν έχουμε μπει στην οθόνη για πρώτη φορά απλώς ζωγραφίζουμε και πάλι τα δεδομένα μας. Αν ερχόμαστε εδώ για πρώτη φορά τότε η getData() είναι αυτή που μας επιστρέφει τα δεδομένα που χρειαζόμαστε για να γεμίσουμε το ListView που χρησιμοποιούμε. Θέτουμε ένα layoutDimmer στην οθόνη μας για να το χρησιμοποιήσουμε αργότερα στο popup window του delete. Στη συνέχεια εμφανίζουμε στην οθόνη το όνομα του τραπεζιού το οποίο πατήθηκε και αν το τραπέζι που πατήθηκε ήταν κενό τότε δεν καλούμε κανένα service

γεμίσματος με data, απλώς εμφανίζουμε στην οθόνη τις μπάρες με τα κουμπιά τους. Στην εικόνα που φαίνεται παρακάτω φαίνεται ότι από τη JSONHelper μέθοδο μας καλούμε την getCurrentOrder () η οποία μας επιστρέφει τα απαραίτητα data.

```
private void getData() throws Exception {

    Log.e("called", "getData() (currentOrder)");
    if (!table.getAvailable()) {
        currentOrder = JSONHelper.getCurrentOrder(table.getTitle());
        Collections.sort(currentOrder);
    }
}
```

Εικόνα 89: getData()

Η initializeListView() η οποία καλείται πρώτη τρέχει τη μέθοδο setTotalPrice() η οποία με τη σειρά της δίνει την κατάλληλη τιμή στο TextView στο πάνω δεξιά μέρος της οθόνης. Εκεί εάν δεν έχουμε επιλέξει τίποτα από τα items που βρίσκονται στη λίστα μας, η οποία θα δημιουργηθεί και πάλι με custom adapter, εμφανίζεται η συνολική τιμή της παραγγελίας. Μόλις αλλάξουμε το selection αλλάζει και η τιμή στο πάνω δεξί μέρος της οθόνης μας.

```
// Calculates and sets total price of selected or all if none is selected
public void setTotalPrice() {

    TextView totalPrice = (TextView) findViewById(R.id.totalPrice);
    float totalSelected = 0;
    float totalUnselected = 0;

    for (CurrentOrderItem item : currentOrder) {
        if (item.isSelected()) {
            totalSelected += item.getPrice();
        } else {
            totalUnselected += item.getPrice();
        }
    }
    totalPrice.setText(new DecimalFormat("##0.00").
    format((totalSelected == 0 ? totalUnselected : totalSelected))+ "€");
}
```

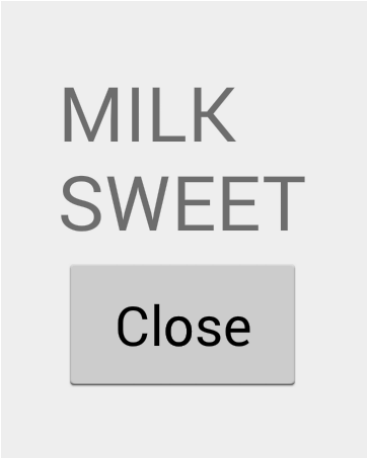
Εικόνα 90: setTotalPrice()

Στον dataAdapter που δημιουργούμε δίνουμε σαν παράμετρο το current_order_list_layout το οποίο είναι το XML αρχείο που θα γεμίζει τη κάθε

γραμμή του ListView, και το όνομα του ListView που θα γεμίζει. Η διαφορά με τον προηγούμενο adapter του κεφαλαίου 6.2 είναι ότι εδώ αναφερόμαστε σε ένα ListView το οποίο έχει μέσα και περισσότερα αντικείμενα όπως δύο εικονίδια από τα οποία το ένα clickable και ένα TextView για την εμφάνιση της τιμή του κάθε προϊόντος της παραγγελίας, και ένα popup που θα αναλύσουμε στο υποκεφάλαιο 6.3.3 αυτού του κεφαλαίου. Τέλος η initializeUI() είναι η μέθοδος που θα δημιουργήσει όλα τα στοιχεία της οθόνης που δεν αφορούν το ListView, όπως ολόκληρη τη μπάρα πάνω και κάτω, και τρέχει πάντα με την onCreate().

6.3.2. INFO BUTTON

Πατώντας το κουμπί του Info στο μέσο της λίστας εμφανίζεται ένα popup το οποίο μας ενημερώνει για τις λεπτομέρειες του προϊόντος, δηλαδή για τα επιλεγμένα s_types του πίνακα sub_menu. Ο κώδικα που υλοποιεί το συγκεκριμένο popup βρίσκεται μέσα στον customAdapter όπου έχει γίνει override η onClick() μέθοδος. Ουσιαστικά τραβάει τα δεδομένα από το JSON και τα τοποθετεί το ένα κάτω από το άλλο για να μας ενημερώσει για τα περιεχόμενα του προϊόντος. Με το κουμπάκι Close το οποίο εμφανίζεται μέσα στο popup απλά κάνουμε ένα dismiss το popup και επαναφέρουμε το alpha του dimmer στο 0 και πάλι για να πάμε πίσω στην οθόνη μας.



Εικόνα 91: Popup window

Ο κώδικας του αν και μεγάλος κάνει τόσα λίγα εμφανισιακά αλλά παρ όλα αυτά όλα τόσο σημαντικά.

6.3.3. DELETE BUTTON

Με το κουμπί αυτό εμφανίζεται ένα ακόμα popup το οποίο μας επιτρέπει να κάνουμε delete ένα ήδη υπάρχον προϊόν μέσα σε ένα order. Αυτός είναι και ο μόνος τρόπος εξαπάτησης ενός συστήματος POS, όταν κάποιος ο οποίος δεν πρέπει έχει στα χέρια το Security Code, αποκτήσει πρόσβαση σε αυτό. Ο τρόπος αυτός είναι η διαγραφή ενός προϊόντος από το Current Order, εν αγνοία του πελάτη και του Administrator. Ως αποτέλεσμα, ο πελάτης θα πληρώσει κανονικά το αντίτιμο του προϊόντος το οποίο κατανάλωσε και ο waiter θα εισπράξει αυτό το ποσό για τον εαυτό του!

```
popupDeleteDeleteButton.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View v) {
        StringBuffer orders = new StringBuffer();
        for (int i = 0; i < itemsToDelete.size(); i++) {
            orders.append(itemsToDelete.get(i).getOrderId());
            if (i < itemsToDelete.size() - 1) {
                orders.append(",");
            }
        }
        EditText securityCode = (EditText) popupView.findViewById(R.id.securityCode);
        try {
            int response = JSONHelper.deleteOrders(securityCode.getText().toString(), orders.toString());
            Toast.makeText(getApplicationContext(),
                    response == 1 ? "Items Deleted!" : "An error occured!", Toast.LENGTH_LONG).show();
        } catch (Exception e) {
            Toast.makeText(getApplicationContext(), "Cannot connect to server!", Toast.LENGTH_LONG)
                    .show();
        }
        popupWindow.dismiss();
        CurrentOrder_Activity.this.layoutDimmer.getForeground().setAlpha(0);
        try {
            getData();
            initializeListView();
        } catch (Exception e) {
            Toast.makeText(getApplicationContext(), "Cannot connect to server!", Toast.LENGTH_LONG)
                    .show();
        }
    }
});
CurrentOrder_Activity.this.layoutDimmer.getForeground().setAlpha(190);
// popupWindow.showAtLocation(popupView, Gravity.CENTER, 0, 0);

EditText securityCode = (EditText) popupView.findViewById(R.id.securityCode);
securityCode.setOnEditorActionListener(new OnEditorActionListener() {

    @Override
    public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
        popupDeleteDeleteButton.callOnClick();
        return true;
    }
});
```

Εικόνα 92: onClickListener() του deleteButton στο popup

Στα προηγούμενα κεφάλαια για το Admnistrator UI είδαμε ότι υπάρχει ένα κουμπί το οποίο ονομάζεται Security Code το οποίο να πατηθεί εμφανίζει έναν

κωδικό ο οποίος είναι περασμένος στη βάση από τους προγραμματιστές του POS. με αυτόν τον κωδικό μπορεί μόνο ο Administrator να κάνει διαγραφή μιας παραγγελίας. Το popup θα εμφανιστεί μόνο όταν είναι επιλεγμένο κάποιο στοιχείο ή όλα της παραγγελίας αλλιώς με κατάλληλο μήνυμα θα μας προτρέψει το POS να επιλέξουμε κάποιο στοιχείο. Η υλοποίηση του popup γίνεται μέσα στην initializeUI( ) και όχι σε κάποιον Adapter γιατί το κουμπί είναι μέρος της κάτω μπάρας που δημιουργείται ανεξάρτητα από το ListVIew της οθόνης αυτής. Αν εισάγουμε λάθος κωδικό τότε θα πάρουμε το μήνυμα "An error occured!" και επίσης το μήνυμα "Cannot connect to server!" αν για κάποιο λόγο δεν μπορεί να γίνει η ταυτοποίηση του Security Code με τη βάση λόγω απώλειας σύνδεσης, μήνυμα το οποίο θα συναντήσουμε σε πολλά σημεία για να προστατέψουμε την εφαρμογή από κατάρρευση. Αν εισάγουμε σωστό Security Code το μήνυμα που θα λάβουμε είναι το "Items Deleted!" και θα επανέλθουμε στην οθόνη με το current order όπου θα καλέσουμε τις getData() και initializeListView() για να δούμε και μπροστά μας τις αλλαγές.

6.3.4. PAY BUTTON

Η διαφορά στην υλοποίηση του Pay με το Delete έγκειται στο γεγονός ότι το Pay περνάει στο server διαφορετικές παραμέτρους μιας και με την πληρωμή του κάθε προϊόντος μπορεί να μας εξάγει σοβαρά στατιστικά στοιχεία. Έτσι λοιπόν στο service που καλείται εκτός από τα id του κάθε προϊόντος προς πληρωμή εισάγεται και το id του χρήστη ο οποίος εκτελεί την πληρωμή. Αυτό γίνεται για να μπορεί το αφεντικό στο τέλος της βάρδιας ή της ημέρας να αναζητήσει από τους κατάλληλους υπαλλήλους τα κατάλληλα ποσά. Δεν παίζει πραγματικό ρόλο ποιος σερβιτόρος πήρε την παραγγελία αλλά ποιος πληρώθηκε την παραγγελία. Αυτά είναι δύο ξεχωριστά στατιστικά δεδομένα.

"Payment completed!" είναι το μήνυμα σε περίπτωση που όλα πάνε καλά στη συναλλαγή με το server και "An error occured!" αν όχι. Επίσης δεν μπορούμε να εκτελέσουμε καμία συναλλαγή με το Pay αν πρώτα δεν έχουμε επιλέξει τουλάχιστον ένα προϊόν από τη λίστα μας. "You didn't select any orders!" είναι το μήνυμα που θα λάβουμε σε μια τέτοια περίπτωση

Να επισημάνουμε εδώ πως έχουμε πάρει και πάλι ένα μέτρο ασφάλειας σε

περίπτωση που το κουμπί Pay πατηθεί κατά λάθος από το χρήστη με επιλεγμένο προϊόν στην οθόνη του. Για να ολοκληρωθεί η ενέργεια Pay πρέπει ο χρήστης υποχρεωτικά να πατήσει δύο φορές το κουμπί Pay μέσα σε τρία δευτερόλεπτα. Έτσι λοιπόν μπορούμε να αποφύγουμε τη λανθασμένη πληρωμή κάποιου προϊόντος. Αυτό θα φανεί και στο Toast που θα πάρουμε αν πατηθεί μια φορά το Pay.

Εικόνα 93: Payment confirmation

6.4. NEW ORDER SCREEN

Στην οθόνη αυτή μπορεί να δημιουργήσει νέα παραγγελία ο χρήστης ή να προσθέσει μια καινούρια σε μια ήδη υπάρχουσα. Αυτό που θα παρατηρήσουμε στην οθόνη αυτή είναι ότι είναι χωρισμένη στα τρία και κάθε κομμάτι της οθόνης περιέχει ένα ξεχωριστό κομμάτι του Menu του καταστήματος.

Εικόνα 94 : New Order Screen

Στην αριστερή λίστα η οποία είναι και αυτή ένα ListView όπως και η προηγούμενη οθόνη, βρίσκονται όλοι οι τύποι προϊόντων που υπάρχουν στη λίστα. Στη δεύτερη λίστα στα δεξιά μας βλέπουμε τα προϊόντα που υπάρχουν για κάθε κατηγορία και έχουμε τη δυνατότητα να επιλέξουμε ένα κάθε φορά όπως και στη λίστα με τις κατηγορίες. Με κόκκινο χρώμα γραμμάτων βλέπουμε τα προϊόντα τα οποία είναι στο cellar μας σε ποσότητα μικρότερη από πέντε τεμάχια. Αφήνουμε το χρήστη να επιλέξει αν θα προσθέσει στη παραγγελία του προϊόν με κόκκινα γράμματα, και με δική του ευθύνη. Αυτό συμβαίνει για καθαρά πρακτικούς λόγους της δουλειάς που γίνεται σε μαγαζιά εστίασης. Το τρίτο κομμάτι της οθόνης είναι ένα GridView και πάλι όπως και στη δεύτερη οθόνη με τα τραπέζια. Εκεί περιλαμβάνονται οι επιλογές για κάθε κατηγορία και πάλι. Σε αυτό το κομμάτι του μενού έχουμε τη δυνατότητα πολλαπλής επιλογής εφ’ όσον το επιθυμούμε.

Στο πάνω μέρος θα δούμε και πάλι τη μπάρα με το όνομα του τραπεζιού και το συνολικό ποσό του προϊόντος που επιλέχθηκε. Στο κέντρο της μπάρας αυτή τη φορά είναι το εικονίδιο το οποίο θα μας μεταφέρει στην επόμενη οθόνη η οποία περιέχει έναν πίνακα με τα προϊόντα που έχουμε επιλέξει κατά την δημιουργία νέας παραγγελίας. Στο κάτω μέρος θα δούμε ότι έχει τοποθετηθεί spinner για να μπορούμε να επιλέξουμε παραπάνω από ένα όμοια προϊόντα, χωρίς να χρειάζεται να συνθέσουμε την ίδια παραγγελία για το προϊόν δεύτερη φορά. Υπάρχει στο μέσο της μπάρας ένα καρτελάκι στο οποίο μπορούμε να γράψουμε κάποια extra πληροφορία που αφορά το προϊόν της παραγγελίας και ένα κουμπάκι save το οποίο προσθέτει στη λίστα της επόμενης οθόνη του προϊόντος που συνθέσαμε. Το XML που αποτελεί αυτή την οθόνη είναι παρόμοιο με τις προηγούμενες οθόνης απλώς εδώ η διαφορά είναι ότι η οθόνη είναι κομμένη στα τρία και διαχειριζόμαστε κάθε κομμάτι της οθόνης ξεχωριστά. Αυτό που αξίζει να δείξουμε είναι ο τρόπος με το οποίο δημιουργείται ένα custom spinner και όχι κάποιο έτοιμο του Android. Στον κώδικα της XML παρακάτω φαίνεται αυτό ακριβώς. Ουσιαστικά χρησιμοποιούμε δύο ImageButtons και ένα TextView ανάμεσα τους και αυξομειώνουμε τη τιμή του TextView με το πάτημα των κουμπιών συν και πλην.

```
<LinearLayout
    android:layout_width="0dp"
    android:layout_height="fill_parent"
    android:layout_weight="1"
    android:orientation="horizontal" >

    <ImageButton
        android:id="@+id/newOrderSpinnerMinusButton"
        android:layout_width="0dp"
        android:layout_height="fill_parent"
        android:layout_weight="1.5"
        android:adjustViewBounds="true"
        android:background="@android:color/transparent"
        android:scaleType="fitCenter"
        android:src="@drawable/spinner_minus" />

    <TextView
        android:id="@+id/newOrderSpinnerText"
        android:layout_width="0dp"
        android:layout_height="fill_parent"
        android:layout_weight="1"
        android:paddingTop="13dp"
        android:text="1"
        android:textAlignment="center"
        android:textSize="20sp" />

    <ImageButton
        android:id="@+id/newOrderSpinnerPlusButton"
        android:layout_width="0dp"
        android:layout_height="fill_parent"
        android:layout_weight="1.5"
        android:adjustViewBounds="true"
        android:background="@android:color/transparent"
        android:scaleType="fitCenter"
        android:src="@drawable/spinner_plus" />
</LinearLayout>
```

Εικόνα 95 : Custom Spinner XML

6.4.1. ΓΕΜΙΣΜΑ ΟΘΟΝΗΣ

Με την εκκίνηση του προγράμματος όπως και όλες μας τις οθόνες έτσι και εδώ τρέχει πρώτη η onCreate(). Έχουμε θέση μέσα σε αυτήν ένα layoutDimmer με μηδενικό alpha για να το χρησιμοποιήσουμε με το πάτημα του κουμπιού για την εισαγωγή extra πληροφορίας στην παραγγελία. Καλούμε τη μέθοδο getData() η οποία με τη σειρά της θα καλέσει το service με τη βοήθεια της JSONHelper και έτσι έχουμε όλα τα δεδομένα της βάσης μας από ένα service το οποίο καλέσαμε μόλις μια φορά, στη μεταβλητή menu. Η τοποθέτηση αντικειμένων στην οθόνη ξεκινάει καλώντας της initializeUI η οποία ζωγραφίζει στη οθόνη μόνο την πάνω και την κάτω μπάρα και τίποτα από τα υπόλοιπα στοιχεία της οθόνης. Αυτά είναι ένα το όνομα του τραπεζιού, το κουμπάκι για redirect στην επόμενη οθόνη, το συνολικό ποσό της επιλογής του order, το spinner, το κουμπάκι special description της παραγγελίας και το doneButton με το οποίο προσθέτουμε ότι έχει επιλεχθεί σε ένα ArrayList.

```
private void initializeUI() {
    tableName = getIntent().getStringExtra("tablename");
    ((TextView) findViewById(R.id.newOrderTableName)).setText(tableName);
    reviewButton = (ImageButton) findViewById(R.id.newOrderReviewButton);
    spinnerMinusButton = (ImageButton) findViewById(R.id.newOrderSpinnerMinusButton);
    spinnerPlusButton = (ImageButton) findViewById(R.id.newOrderSpinnerPlusButton);
    specialDescriptionButton = (ImageButton) findViewById(R.id.newOrderSpecialDescriptionButton);
    doneButton = (ImageButton) findViewById(R.id.newOrderDoneButton);
    spinnerText = (TextView) findViewById(R.id.newOrderSpinnerText);
}
```

Εικόνα 96 : initializeUI ()

Μόλις ολοκληρωθεί ο σχεδιασμός αυτής της μεθόδου, καλείται στην onCreate() η initializeCategoriesList(), η οποία είναι υπεύθυνη για το σχεδιασμό της υπόλοιπης οθόνης. Μέσα σε αυτή την μέθοδο δημιουργούμε και πάλι ένα δικό μας custom adapter με όνομα categoriesDataAdapter στον οποίο δίνουμε σαν παράμετρο το new_order_category_layout με το οποίο θα γεμίσει την λίστα με τις κατηγορίες στα αριστερά μας. Δεν διαφέρει σε τίποτα ο συγκεκριμένος adapter με τον adapter που χρησιμοποιούμε στη δεύτερη και στην τρίτη οθόνη του προγράμματος μας, γι αυτό δεν θα τον δούμε αναλυτικά. Αυτό που αξίζει να σημειώσουμε στην initializeCategoriesList() είναι ότι Choice_Mode_Single και με το πάτημα ενός category πλέον καλείται μέσα στην onItemClick() η initializeTypesLists() για το συγκεκριμένο item που είναι selected.

```
private void initializeCategoriesList() {
    categoriesDataAdapter = new CategoriesListView_Adapter(this, R.layout.new_order_category_layout, menu);
    final ListView categoriesListView = (ListView) findViewById(R.id.newOrderCategoriesList);
    categoriesListView.setAdapter(categoriesDataAdapter);
    categoriesListView.setChoiceMode(ListView.CHOICE_MODE_SINGLE);
    categoriesListView.setClickable(true);
    categoriesListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> arg0, View arg1, int position, long arg3) {

            MenuCategory cat = (MenuCategory) categoriesListView.getItemAtPosition(position);
            makeSelected(cat);
            arg1.setSelected(true);
            initializeTypesLists();
        }

    });
}
```

Εικόνα 97 : initializeCategoriesList()

Εκεί θα δημιουργηθούν τα επόμενα κομμάτια της οθόνης μας, ένα ListView με τα m_types και τα s_types για το επιλεγμένο category. Επειδή κάθε s_type ανήκει και αυτό σε ένα category και όχι σε ένα m_type βάση του σχεδιασμού της βάσης, δεν χρειαζόμαστε άλλα initialize μέθοδο για να κάνει περισσότερη δουλειά. Με το πάτημα του κουμπιού ενός category θα δούμε ότι δημιουργούνται αυτόματα λίστες στα δεξιά και σε όλα τα σημεία της οθόνης.

Χρησιμοποιώντας εκ νέου δύο καινούριους adapters τον ένα για ListView και τον άλλο για GridView γεμίζουμε τα υπόλοιπα κομμάτια της οθόνης μας με τα δεδομένα της getData(). Αυτό που αξίζει να αναφερθεί και πάλι είναι ότι το πρόγραμμα μας δίνει τη δυνατότητα πολλαπλής επιλογής στα s_types μιας

και έτσι είναι το σωστό, πχ κάποιος θέλει τον καφέ του γλυκό ελαφρύ με γάλα, ενώ δε συμβαίνει το ίδιο για τα m_type όπου πάλι έχουμε CHOICE_MODE_SINGLE. Με τη μέθοδο makeSelected() αυτό που καλούμε ουσιαστικά, κάνουμε το background του αντικειμένου που επιλέξαμε να αλλάξει φόντο και να από-επιλεγεί το item που είχαμε επιλέξει νωρίτερα. Για να δουλέψει το multiple choice στα s_type με τον τρόπο που εμείς θέλαμε έχουμε παραμετροποιήσει την CheckableLinearLayout κλάση να κάνει select πολλά αντικείμενα χωρίς να χρειάζεται long-pressed click για να ενεργοποιηθεί αυτό το functionality το οποίο είναι και το default στον Android, κάνοντας implement το Checkable interface.

6.4.2. ΥΠΟΛΟΓΙΣΜΟΣ ΤΙΜΗΣ

Μέσα στην initializeTypesLists() το τελευταίο πράγμα που γίνεται είναι ο υπολογισμός της τιμής και του order που συνθέτουμε και φαίνεται στο πάνω δεξιό μέρος της οθόνης. Για το σκοπό αυτό καλούμε την setTotalItemPrice().

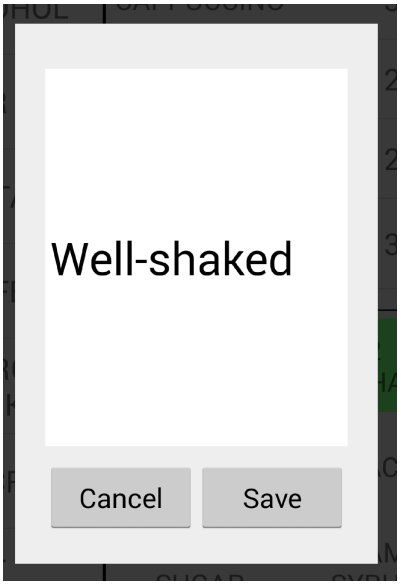
```
public void setTotalItemPrice() {

    TextView totalPriceText = (TextView) findViewById(R.id.newOrderItemPrice);
    TextView quantityText = (TextView) findViewById(R.id.newOrderSpinnerText);

    totalPrice = 0;
    int quantity = Integer.parseInt(quantityText.getText().toString());
    // float totalUnselected = 0;

    for (MenuCategory category : menu) {
        if (category.isSelected()) {
            for (MType mType : category.getMTypes()) {
                if (mType.isSelected()) {
                    totalPrice += mType.getPrice();
                }
            }
            for (SType sType : category.getSTypes()) {
                if (sType.isSelected())
                    totalPrice += sType.getPrice();
            }
        }
    }
    totalPrice *= quantity;
    totalPriceText.setText(new DecimalFormat("##0.00").format(totalPrice) + "€");
}
```

Εικόνα 98 : setTotalItemPrice()

Εδώ θα πάρουμε για τα επιλεγμένα mtypes και stypes τις τιμές που έχουν στο price με τη getPrice() και θα τα προσθέσουμε για να δημιουργήσουμε ένα καινούριο price, το οποίο θα πολλαπλασιάσουμε με την τιμή που έχουμε δώσει στο spinner (quantity) για να είμαστε εντάξει στην παραγγελία που θα στείλουμε. Σε αυτό το σημείο να επισημάνουμε ότι τη δουλειά αυτή την κάνει ο Client για τον server και είναι το μοναδικό σημείο της εφαρμογής που γίνεται αυτό. Αρχικά είχε σχεδιαστεί ένα service το οποίο υπολόγιζε τη τιμή των s_types και των m_types σε php. Για να αποφύγουμε αυτό να γίνεται δύο φορές αποφασίσαμε ότι θα την κάνει αυτή τη δουλειά ο client γιατί έτσι και αλλιώς πρέπει να είναι ενήμερος για την αξία των προϊόντων πριν στείλει την παραγγελία για να μπορεί να ενημερώσει τον πελάτη αν χρειαστεί για το ποσό της παραγγελίας και να ενημερωθεί και ο ίδιος.

6.4.3. ΣΥΝΘΕΣΗ ΠΙΝΑΚΑ ΠΡΟΪΟΝΤΟΣ

Για την ολοκλήρωση του order μπορούμε αν θέλουμε να επιλέξουμε την εισαγωγή κάποιου special description με το κουμπάκι που βρίσκεται στο κέντρο της κάτω μπάρας. Θα εμφανιστεί ένα popup με ένα TextView στο οποίο μπορούμε να γράψουμε ότι θέλουμε σαν επεξήγηση στη συγκεκριμένη παραγγελία.



Εικόνα 99: SpecialDescription popup

Τέλος με το κλικ Done για κάποιο order που ολοκληρώσαμε συμπληρώνεται ένα ArrayList με όνομα newOrder με τα αντικείμενα order που

δημιουργήσαμε, το οποίο και θα περάσει στην επόμενη και τελευταία οθόνη μας όπου θα μπορούμε να κάνουμε Review την παραγγελία μας και να την επεξεργαστούμε αν το επιλέξουμε. Το κάθε αντικείμενο περιλαμβάνει μια ένα m_id που είναι ο κωδικός του επιλεγμένου m_type, το id του χρήστη που έκανε τη παραγγελία, το συνολικό κόστος του προϊόντος, το special description που μπορεί να έχει βάλει κάποιος σερβιτόρος και έναν πίνακα με τα s_types για το συγκεκριμένο m_type που επιλέχθηκε.

```
doneButton.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        if (totalPrice!=0){
        NewOrder newOrderItem = new NewOrder(selectedMType.getMType(), selectedMType.getId(),
                specialDescription, totalPrice);
        newOrder.add(newOrderItem);
        }
        else{
            Toast.makeText(getApplicationContext(), "Invalid order!!", Toast.LENGTH_LONG).show();
        }
    }
});
```

Εικόνα 100 : Δημιουργία αντικειμένου newOrderItem

6.5. REVIEW AND SEND ORDER SCREEN

Η τελευταία οθόνη στο πρόγραμμα μας είναι αυτή στην οποία ελέγχουμε τα προϊόντα που έχουμε προσθέσει κατά τη σύνθεση της παραγγελίας στην προηγούμενη οθόνη και μπορούμε να τα στείλουμε στο server ή να διαγράψουμε κάποιο προϊόν, αν αυτό το προσθέσαμε εσφαλμένα. Μπορούμε επίσης να πάμε πίσω στην οθόνη σύνθεσης της νέας παραγγελίας για να προσθέσουμε όσα ακόμη προϊόντα επιθυμούμε. Μιας και η συγκεκριμένη οθόνη αποτελείται και αυτή από ένα ListView δεν κρίνεται αναγκαία η επεξήγηση κώδικα XML σε αυτή την περίπτωση, ο οποίος μοιάζει σε μεγάλο ποσοστό με τον κώδικα της τρίτης οθόνης της εφαρμογή μας. Το συγκεκριμένο ListView περιλαμβάνει αναλυτικά τα orders που εισήχθησαν από τον χρήστη και μπορούμε να στείλουμε επιλεκτικά όποια orders θέλουμε για να καταχωρηθούν στη βάση. Τα κατάλληλα μηνύματα σε Toast είναι αυτά που θα μας προτρέψουν να επιλέξουμε κάποιο προϊόν πριν στείλουμε κατά λάθος ίσως κενή παραγγελία ή αν κατά λάθος προσπαθήσουμε να πιέσουμε το κουμπί της διαγραφής χωρίς να επιλέξουμε κάποιο προϊόν για διαγραφή.

Εικόνα 101: ReviewNewOrder Screen

Στην οθόνη βλέπουμε και πάλι μια μπάρα στο επάνω μέρος, η οποία περιλαμβάνει το checkbox All για να επιλέξουμε όλα τα προϊόντα της παραγγελίας, το νούμερο του τραπεζιού στο οποίο βρισκόμαστε και το συνολικό ποσό της παραγγελίας που συνθέσαμε. Στο κάτω μέρος της οθόνης η άλλη μπάρα η οποία υπάρχει, περιλαμβάνει το κουμπάκι στα αριστερά το οποίο μας γυρίζει στην προηγούμενη οθόνη για να συμπληρώσουμε ενδεχομένως την υπάρχουσα παραγγελία, το κουμπάκι Delete στο μέσο της οθόνης το οποίο μας επιτρέπει να σβήσουμε προϊόντα τα οποία έχουν επιλεχθεί εσφαλμένα και το κουμπάκι Send το οποίο ολοκληρώνει τη νέα μας παραγγελία στέλνοντας τη στο Server. Με το πάτημα του κουμπιού Send μεταφερόμαστε στην οθόνη του Current Order του τραπεζιού η οποία γίνεται εκ νέου initialize και μας δείχνει την ανανεωμένη – συμπληρωμένη παραγγελία του συγκεκριμένου τραπεζιού. Αν δεν είχαμε επιλέξει να στείλουμε στο Server όλα τα προϊόντα αλλά κάποια από αυτά, αυτό σημαίνει πως η σύνθεση της παραγγελίας έχει μείνει στη μέση και έτσι θα ενημερωθούμε για αυτό με το κατάλληλο Toast στην οθόνη του Current Order σε περίπτωση που

προσπαθήσουμε να τερματίσουμε τη παραγγελία για να μεταφερθούμε στην οθόνη Table State. Το ListView το οποίο φαίνεται στο μέσο της οθόνης, αποτελείται όπως φαίνεται και στην εικόνα παραπάνω από τα επιλεγμένα προϊόντα, κατά τη σύνθεση της παραγγελίας. Μπορούμε να δούμε αναλυτικά τα προϊόντα του μενού και του υπομενού που έχουμε επιλέξει και το ποσό το οποίο κοστίζει κάθε προϊόν.

6.6. ΓΕΜΙΣΜΑ ΟΘΟΝΗΣ

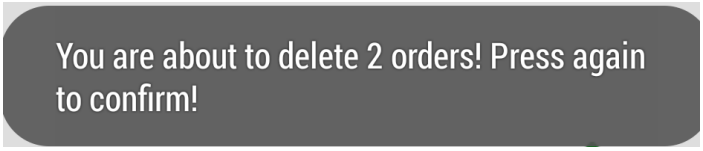
Όπως και στις προηγούμενες οθόνες μας έτσι και στη τελευταία κατά την έναρξη της οθόνης τρέχει η μέθοδος onCreate() η οποία καλεί την getData() αρχικά και στη συνέχεια την initializeUI() η οποία ζωγραφίζει τα πάντα στη οθόνη μας εκτός από το ListView το οποίο βρίσκεται στο μέσο της οθόνης. Η getData () διαφέρει από τις υπόλοιπες μεθόδους που τραβούν δεδομένα από το server στις προηγούμενες οθόνες. Εδώ, παίρνει τα δεδομένα της από τον πίνακα new_order τον οποίο συνθέτουμε στην προηγούμενη οθόνη και είναι τύπου static. Static τον ορίσαμε για να του προσθέσουμε αυτή ακριβώς τη λειτουργία. Να είναι δηλαδή προσβάσιμος σε οποιοδήποτε σημείο της εφαρμογής. Έτσι λοιπόν η initializeListView() η οποία καλείται μετά τη initializeUI(), υπολογίζει αρχικά το συνολικό ποσό της παραγγελίας στο επάνω δεξί μέρος της οθόνης το οποίο αρχικά είναι το συνολικό ποσό της παραγγελίας αν δεν έχουμε επιλέξει κάποιο προϊόν και αλλάζει κάθε φορά που επιλέγουμε ή αποκλείουμε ένα προϊόν. Δημιουργώντας και πάλι ένα δικό μας adapter τύπου ReviewNewOrder_Adapter αυτή τη φορά, δίνουμε σαν παράμετρο στη μέθοδο το layout με το οποίο θα γεμίσει η οθόνη μας, το review_new_order_list_layout που είναι τύπου XML. Τέλος γεμίζουμε αυτή την οθόνη με τον adapter τον οποίο δημιουργήσαμε μόλις πριν.

```
private void initializeListView() {
    Log.e("called", "initializeListView()");

    setTotalPrice();
    dataAdapter = new NewOrderListView_Adapter(this, R.layout.review_new_order_list_layout, newOrder);
    ListView listView = (ListView) findViewById(R.id.reviewOrderListView);

    // Assign adapter to ListView
    listView.setAdapter(dataAdapter);
}
```

Εικόνα 102: initialize UI ()

6.7. ΚΟΥΜΠΙ DELETE

Μπορούμε είτε να επιλέξουμε τη διαγραφή όλων των προϊόντων επιλέγοντας τα με το All είτε να επιλέξουμε μερικά και να τα κάνουμε Delete. Βασική προϋπόθεση είναι να επιλέξουμε τουλάχιστον ένα από τα προϊόντα της λίστας για να δουλέψει το κουμπάκι Delete. Έτσι λοιπόν αυτό που κάνει το Delete είναι να διαγράφει από τον static πίνακα new_order το αντικείμενο το οποίο επιλέξαμε και να φορτώνει εκ νέου την οθόνη η οποία, θα μας δείξει τα προϊόντα που απέμειναν στην παραγγελία κάνοντας refresh και καλώντας την initializeListView(). Αν διαγράψουμε όλα τα αντικείμενα από τον πίνακα new_order μας πάει στην προηγούμενη οθόνη στην οποία θα εισάγουμε νέα παραγγελία. Για να ολοκληρώσουμε την ενέργεια του Delete πρέπει να πατήσουμε το κουμπί δύο φορές μέσα σε τρία δευτερόλεπτα για να επιβεβαιώσουμε την ενέργεια πράγμα για το οποίο μας ενημερώνει το κατάλληλο Toast. Αυτό είναι ένα μικρό μέτρο ασφαλείας που έχουμε θέσει για την αποφυγή ανεπιθύμητων ενεργειών.



Εικόνα 103: Delete Confirmation Toast

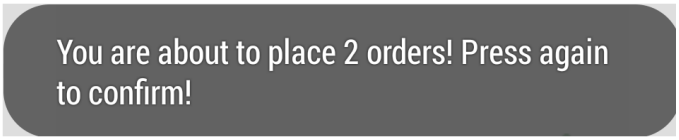
Ο κώδικας οποίος υλοποιεί το Delete είναι ο εξής.

```
deleteButton = (ImageButton) findViewById(R.id.reviewDeleteButton);
deleteButton.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        final ArrayList<NewOrder> itemsToDelete = (selectedAll ? newOrder : getSelectedItems());
        if (itemsToDelete.size() != 0) {
            if (confirmDelete) {
                newOrder.removeAll(itemsToDelete);
                if (newOrder.size() == 0) {
                    finish();
                } else {
                    initializeListView();
                }
            } else {
                Toast.makeText(
                        getApplicationContext(),
                        "You are about to delete " + itemsToDelete.size()
                                + (itemsToDelete.size() > 1 ? " orders!" : " order!")
                                + " Press again to confirm!", Toast.LENGTH_LONG).show();
                confirmDelete = true;
                new Handler().postDelayed(new Runnable() {
                    @Override
                    public void run() {
                        confirmDelete = false;
                    }
                }, 3 * 1000);
            }
        } else {
            Toast.makeText(getApplicationContext(), "You didn't select any items!", Toast.LENGTH_LONG).show();
        }
    }
});
```

Εικόνα 104: Delete Button onClickListener()

6.8. SEND BUTTON

Το κουμπάκι αυτό είναι ίσως το πιο περίπλοκο σε λειτουργία κουμπί όλου το POS όσον αφορά στο Client σε Android. Όπως και το Delete έτσι και το Send, για να λειτουργήσει, πρέπει να έχει επιλεχθεί τουλάχιστον ένα προϊόν της λίστας. Επίσης πρέπει να το πατήσουμε και αυτό το κουμπί μέσα σε τρία δευτερόλεπτα δύο φορές για να επιβεβαιώσουμε την αποστολή της παραγγελίας των επιλεγμένων προϊόντων.



Εικόνα 105: Send Confirmation Toast

6.8.1. sendOrder() METHOD

Το Send Button αφού πατηθεί δύο φορές ολοκληρώνει την αποστολή της παραγγελίας για τα επιλεγμένα προϊόντα. Αν τα προϊόντα για τα οποία πατήσαμε Send ήταν όλα τα προϊόντα της λίστας, τότε μεταφερόμαστε στην τρίτη οθόνη του προγράμματος, στην Current Order Screen, την οποία και κάνουμε initialize εκ νέου για να μας δείξει τις αλλαγές στη βάση, και να επιβεβαιώσουμε ότι οι παραγγελίες μας όντως έχουν σταλεί. Αν επιλέξουμε να στείλουμε ορισμένα από τα προϊόντα της λίστας μας, τότε μετά την αποστολή των προϊόντων παραμένει ανοιχτή η οθόνη Review για να αποφασίσουμε τι θα κάνουμε με τα υπόλοιπα προϊόντα της παραγγελίας. Αυτό που είναι σημαντικό να κατανοήσουμε είναι ο τρόπος με τον οποίο επιτυγχάνεται η επικοινωνία με τον server και τις ενέργειες που γίνονται για να επιτευχθεί η επικοινωνία. Στην παράγραφο 5.4 αναλύσαμε τον τρόπο με τον οποίο ο server διαχειρίζεται ένα http post και πώς παίρνει από αυτό τα δεδομένα. Εδώ θα δούμε πώς δημιουργούνται.

Η δουλεία σε αυτή τη περίπτωση δε γίνεται με τη JSONParser η οποία καλείται από τη JSONHelper συνήθως αλλά εν ολίγοις γράφουμε και πάλι την κλάση JSONParser την οποία και θα εξηγήσουμε στο επόμενο κεφάλαιο. Ο client καλεί το service add_order_json.php και του δίνει σαν δεδομένα, ένα ArrayList το οποίο έχει σε κάθε γραμμή του, το id του σερβιτόρου ο οποίος

εκτελεί την παραγγελία, το όνομα – αριθμό του τραπεζιού στο οποίο ανήκει η παραγγελία, και στο τρίτο πεδίο ένα άλλο ArrayList, το οποίο περιλαμβάνει σε κάθε του γραμμή, το m_id του προϊόντος στο οποίο αναφέρετε η εγγραφή, τα s_id τα οποία συνοδεύουν αυτό το m_id, το πεδίο specialDesctription το οποίο είναι η επεξήγηση της παραγγελίας, και το συνολικό ποσό του συγκεκριμένου προϊόντος. Το δεύτερο ArrayList το οποίο περιλαμβάνεται μέσα στο μεγάλο ArrayList το συνθέτουμε στη κλάση NewOrder μαζί με τα αντικείμενα τύπου NewOrder το οποίο είναι πρακτικά μια γραμμή της λίστας του Review Screen. Επειδή δεν είμαστε σε θέση να γνωρίζουμε εξ' αρχής όλα τα πιθανά ενδεχόμενα μιας παραγγελίας του πελάτη, ούτε φυσικά και το μέγεθος της ανά τραπέζι, είμαστε υποχρεωμένοι να χρησιμοποιήσουμε αυτό τον τρόπο επικοινωνίας για να στείλουμε μεγάλο ενδεχομένως όγκο πληροφορίας με μιας.

```
public static void sendOrder(final ArrayList<NewOrder> newOrder, final String tableName, final int waiterId)
        throws Exception {
    HttpClient client = new DefaultHttpClient();
    HttpConnectionParams.setConnectionTimeout(client.getParams(), 10000);
    HttpResponse response;
    JSONObject json = new JSONObject();
    JSONArray order = new JSONArray();
    url = urlpart + "/json/add_order_json.php";

    HttpPost post = new HttpPost(url);
    json.put("table_name", tableName);
    json.put("waiter_id", waiterId);

    order = new JSONArray();
    for (NewOrder item : newOrder) {
        order.put(item.getJSONObject());
    }
    json.put("order", order);
    Log.e("JSON OBJECT: ", json.toString());
    StringEntity se = new StringEntity(json.toString());
    se.setContentType(new BasicHeader(HTTP.CONTENT_TYPE, "application/json"));
    post.setEntity(se);
    response = client.execute(post);
    if (response != null) {
        InputStream in = response.getEntity().getContent();
    }

}
```

Εικόνα 106 : sendOrder ()της JSONHElper

Το κεφάλαιο που ακολουθεί εξηγεί όλα τα παραπάνω στοιχεία που χρησιμοποιούνται στον κώδικα για τη επικοινωνία Client – Server με τη μέθοδο post, διότι η σύνταξη του κώδικα μοιάζει στα περισσότερα σημεία με τη JSONParser.

6.9. CLASS JSONPARSER

Αξίζει να δούμε τη δουλειά η οποία γίνεται στη JSONParser. Και αυτό γιατί η συγκεκριμένη μέθοδος παίρνει σαν είσοδο το url του service που υπάρχει στον server και επεξεργάζεται κατάλληλα το αποτέλεσμα σε μορφή JSON. Χρησιμοποιείται κατά κόρον στη κλάση JSONHelper και είναι υπεύθυνη για την επιστροφή του JSONObject το οποίο έχει μέσα τα στοιχεία που χρειαζόμαστε.[20][22]

```
public JSONObject getJSONFromUrl(String url) {
    StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();

    StrictMode.setThreadPolicy(policy);
    // Making HTTP request
    try {
        // defaultHttpClient
        DefaultHttpClient httpClient = new DefaultHttpClient();
        HttpPost httpPost = new HttpPost(url);

        HttpResponse httpResponse = httpClient.execute(httpPost);
        HttpEntity httpEntity = httpResponse.getEntity();
        is = httpEntity.getContent();

    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    } catch (ClientProtocolException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }

    try {
        BufferedReader reader = new BufferedReader(new InputStreamReader(is, "iso-8859-1"), 8);
        StringBuilder sb = new StringBuilder();
        String line = null;
        while ((line = reader.readLine()) != null) {
            sb.append(line + "\n");
        }
        is.close();
        json = sb.toString();
    } catch (Exception e) {
        Log.e("Buffer Error", "Error converting result " + e.toString());
    }

    // try to parse the string to a JSON object
    try {
        jObj = new JSONObject(json);
    } catch (JSONException e) {
        Log.e("JSON Parser", "Error parsing data " + e.toString());
    }

    // return JSON String
    return jObj;

}
```

Εικόνα 107: Κλάση JSONParser

Αυτό που πρέπει να δημιουργηθεί είναι ένα αντικείμενο τύπου InputStream για να το δώσουμε σαν παράμετρο στη μέθοδο BufferedReader. Για να γεμίσουμε με δεδομένα τη μεταβλητή InputStream παίρνουμε τα περιεχόμενα του httpEntity αντικειμένου το οποίο με τη σειρά του περιέχει δεδομένα από το αντικείμενο httpPost στο οποίο και εισάγουμε το url του service που βρίσκεται στο server. Αφού έχουμε ότι χρειαζόμαστε σε ένα αντικείμενο τύπου InputStream, γεμίζουμε ένα άλλο αντικείμενο τύπου StringBuilder με κείμενο για όσες γραμμές επιστραφούν από το αντικείμενο reader τύπου BufferedReader. Ουσιαστικά κάνουμε append τις γραμμές του reader αντικειμένου μας σε ένα String.

Τέλος αποθηκεύουμε το String αυτό στη μεταβλητή json τύπου String και βάση αυτού του String δημιουργούμε ένα αντικείμενο τύπου JSONObject το οποίο και κάνουμε return με το τέλος της μεθόδου. Τα υπόλοιπα από κει και πέρα είναι δουλεία της κάθε μεθόδου να κρατήσει ή να πετάξει πράγματα από το αντικείμενο αυτό, αναλόγως με τη δουλειά που θέλει να κάνει. Με άλλα λόγια μπορεί ο server να επιστρέφει πληροφορία η οποία δεν είναι αναγκαία για τον client οπότε στις μεθόδους της JSONHelper παίρνουμε την πληροφορία που χρειαζόμαστε από κάθε service που καλούμε.

ΕΠΙΛΟΓΟΣ

Σε αυτό το κεφάλαιο παρουσιάστηκε λεπτομερώς η λειτουργία και η υλοποίηση του Client σε Android. Φυσικά μια τέτοια εφαρμογή είναι ότι πιο σημαντικό για τη λειτουργία ενός Point Of Sale. Καταλαβαίνουμε ότι με τη χρήση της XML και της Java μπορεί κάποιος να δημιουργήσει μια εφαρμογή σε Android μόνο με τη χρήση των κατάλληλων εργαλείων και τεχνολογιών που προσφέρονται. Τα περιθώρια βελτίωσης μιας τέτοια εφαρμογής είναι μεγάλα με τη προσθήκη νέων features για την ευκολότερη και γρηγορότερη πρόσβαση του χρήστη στις επιλογές του προγράμματος, όπως ο ανασχηματισμός του κώδικα σε fragments και η υλοποίηση της επικοινωνίας Client – Server σε ξεχωριστό thread για να γίνει πιο optimized η εφαρμογή.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Αυτό που μένει από αυτή την πτυχιακή εργασία δεν είναι άλλο από τη γνώση που πήραμε πάνω σε αντικείμενα προγραμματισμού που παλαιότερα δεν είχαμε ανακαλύψει ή γνωρίζαμε ελάχιστα. Η δημιουργία εφαρμογής σε Android αλλά και η δημιουργία services σε php από τη μεριά του server ήταν κάτι άγνωστο πριν από τη συγκεκριμένη πτυχιακή εργασία.

Οι τεχνολογίες που χρησιμοποιήθηκαν, οι γλώσσες προγραμματισμού που γέμισαν χιλιάδες γραμμές, και τα εργαλεία που δούλεψαν για αυτό το αποτέλεσμα πολλά και διάφορα, με βασικά εργαλεία την PHP, τη MySQL, την Java και την XML.

Με την εφαρμογή για τον Administrator, τους σεφ και τους σερβιτόρους, δημιουργήσαμε ένα ολοκληρωμένο σύστημα POS το οποίο μπορεί να λειτουργήσει χωρίς πρόβλημα σε μια επιχείρηση μαζικής εστίασης. Με τη χρήση του συστήματος κατανοήσαμε ότι μόνο μέσα από τον προγραμματισμό δεν γίνεται να προβούμε στην σωστή αποσφαλμάτωση των προγραμμάτων μας αλλά μόνο μέσα από συχνές δοκιμές και πολλά crashes των εφαρμογών.

Οι δυνατότητες ενός τέτοιου προγράμματος στα πλαίσια της πτυχιακής μας εργασίας ξεπέρασαν τις αρχικές προσδοκίες αλλά στα πλαίσια του επαγγελματισμού που μας περιμένει από δω και στο εξής υπάρχουν τεράστια περιθώρια βελτίωσης.

Η χρήση των τεχνολογιών OLAP, για την εξόρυξη πληροφορίας από τα δεδομένα της βάσης μας, η χρήση fragments για την καλύτερη λειτουργία του προγράμματος Client σε android, ο υπολογισμός της αποθήκης προϊόντων με καλύτερο τρόπο για ποσότητες χύμα και όχι σε τεμάχια, η δημιουργία εφαρμογής για PC για την κουζίνα και τον Admin, η εκτύπωση ημερησίου δελτίου κίνησης (Z) από την ταμειακή μηχανή αποδείξεων ακόμα και η online σύνδεση του POS με τους προμηθευτές για αυτόματες παραγγελίες, είναι κάποιες από τις κύρια features που μας έρχονται στο μυαλό.

7. ΒΙΒΛΙΟΓΡΑΦΙΑ

ΒΙΒΛΙΑ

1. Μελέτη, σχεδιασμός και ανάπτυξη ηλεκτρονικού καταστήματος, Μαυράκης Κωσταντίνος, 2011
2. Pro Android 3, Hashimi-Komatineni-MacLean, APress, 2011
3. The Busy Coder's Guide To Android Development, Murphy, CommonsWare, 2009
4. Android In Action, Ableson-Sen-King-Ortiz, Manning Publications, 2011
5. Εισαγωγή στις αποθήκες δεδομένων, Πάνος Βασιλειάδης, Εξόρυξη δεδομένων, 2008

WEBSITES

6. http://source.android.com- Το site που μπορείτε να κατεβάσετε τον πηγαίο κώδικα του Android και να αναμιχθείτε ενεργά με το project Android
7. http://developer.android.com - Απευθύνεται αποκλειστικά σε developers και εδώ μπορείτε να προμηθευτείτε το Android SDK αλλά και να βρείτε πηγές που θα σας βοηθήσουν στην ανάπτυξη των εφαρμογών σας
8. http://en.wikipedia.org/wiki/Point_of_sale
9. http://www.w3schools.com/css/
10. http://www.w3schools.com/php/
11. http://phpmailer.worxware.com/index.php?pg=sf&p=sfp
12. http://stackoverflow.com
13. http://dev.mysql.com/downloads/workbench/
14. http://php.net/manual/en/
15. http://www.softwareadvice.com/retail/
16. http://www.phpjabbers.com/php--mysql-select-data-and-split-on-pages-php25.html
17. http://stackoverflow.com/questions/1478295/what-does-async-false-do-in-JQuery-ajax
18. http://json.org/json-el.html

19. http://api.JQuery.com
20. http://stackoverflow.com/questions/23503847/json-parser-in-android
21. http://www.w3schools.com/php/func_mysqli_insert_id.asp
22. http://hmkcode.com/android-send-json-data-to-server

ΕΠΙΣΤΗΜΟΝΙΚΑ ΠΡΟΓΡΑΜΜΑΤΑ

23. XAMPP (https://www.apachefriends.org/download.html)
24. phpMyAdmin (https://www.apachefriends.org/download.html)
25. Eclipse με ADT (https://developer.android.com/sdk/index.html?hl=i)
26. Notepad ++ (http://notepad-plus-plus.org/)
27. Apache Server (https://www.apachefriends.org/download.html)
28. MySQL Workbench (http://downloads.mysql.com/archives/workbench/)