Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

“Εφαρμογή σε Android με διαχείριση απομακρυσμένης βάσης δεδομένων αποθήκης.”

Σπουδαστής: **Μαυροφοράκης Μιχαήλ (ΑΜ:3846).**

Επιβλέπων Καθηγητής: **Αϊβαλής Κώστας.**

Υπεύθυνη Δήλωση

Βεβαιώνω ότι είμαι ο συγγραφέας της παρούσας πτυχιακής εργασίας και ότι οποιαδήποτε βοήθεια έλαβα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται. Ακόμα έχω αναφορά σε πηγές από τις οποίες έκανα χρήση δεδομένων, λέξεων ή ιδεών. Επίσης, βεβαιώνω ότι η παρούσα εργασία προετοιμάστηκε από εμένα προσωπικά για τις απαιτήσεις του προγράμματος σπουδών του τμήματος Πληροφορικής του Τ.Ε.Ι Κρήτης.

ΟΝΟΜΑΤΕΠΩΝΥΜΟ ΣΠΟΥΔΑΣΤΗ ΥΠΟΓΡΑΦΗ

Μαυροφοράκης Μιχαήλ

Περίληψη

Ο σκοπός αυτής της πτυχιακής αρχικά, είναι να παρουσιάσει με ποιο τρόπο μπορούν να αλληλοεπιδράσουν μεταξύ τους ένας server ο οποίος έχει την βάση με όλα τα δεδομένα κάποιου οργανισμού η κοινότητας, και μια απλή σχετικά εφαρμογή η οποία επικοινωνεί με την βάση δεδομένων και διαχειρίζεται τα στοιχεία της.

Η εφαρμογή αυτή σε συνεργασία με την βάση δεδομένων σε ένα απομακρυσμένο server, αναπαριστά και καλύπτει τις βασικές ανάγκες μίας αποθήκης η οποία πρέπει να κρατά βασικά στοιχεία προς εξυπηρέτηση της επιχείρησης. Η χρήση εφαρμογής για αυτό το σκοπό, προσφέρει πολλές λιτότητες και διευκολύνσεις, καθώς επιτρέπει στον χρήστη να διαχειρίζεται τα δεδομένα του από οποιοδήποτε σημείο και δίκτυο χωρίς να χρειάζεται να είναι στον κεντρικό, για παράδειγμα υπολογιστή της αποθήκης.

Κλείνοντας, πρέπει να τονίσουμε ότι η χρήση των έξυπνων συσκευών έχει διεισδύσει για τα καλά στην ζωή μας. Έτσι, μπορούμε να χρησιμοποιήσουμε αυτές τις συσκευές για να μπορούμε να έχουμε άμεση πρόσβασή στην επιχείρηση μας ή στην προσωπική βάση δεδομένων μας.

Abstract

The purpose of this dissertation, is to show how a server that has the basis of all the data of an organization or community can interact with a simple application that communicates with the database and manages the data.

This application, in conjunction with the database in a remote server, represents and covers the basic needs of a warehouse that has to hold basic data to serve the business. The use of an application for this purpose offers many austerity and convenience as it allows the user to manage his data from anywhere and without having to be on the central warehouse computer.

In closing, we must emphasize that the use of smart devices has penetrated well into our lives, so we can use these devices to have direct access to our business or our personal database.

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω τον εισηγητή της πτυχιακής εργασίας, κύριο Αϊβαλή Κώστα, καθηγητή του τμήματος μηχανικών πληροφορικής του ΤΕΙ Κρήτης, για την βοήθεια που πρόσφερε, για την διεκπεραίωση αυτής της πτυχιακής εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια και τους φίλους μου οι οποίοι με υποστήριξαν καθ' όλη την διάρκεια εκπόνησης και συγγραφής της πτυχιακής εργασίας.

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1
1.1Εισαγωγή

Σε αυτό το κεφάλαιο θα κάνουμε μία περιγραφή της εργασίας και μια αναφορά για τα Smartphones. Για να υλοποιήσουμε την εφαρμογή πρέπει πρώτα να κατασκευάσουμε μία βάση δεδομένων, η οποία θα κρατά τα στοιχεία της επιχείρησης ή του οργανισμού μας και συγκεκριμένα της αποθήκης μας. Για αυτό το σκοπό, επέλεξα να χρησιμοποιήσω ένα raspberry pi 3 το όποιο θα αντιπροσωπεύει τον Server της αποθήκης μας. Παρακάτω θα αναφερθούν και οι λόγοι για τους οποίους επέλεξα αυτόν τον μικροϋπολογιστή καθώς και μερικά χαρακτηριστικά του.

Για το raspberry pi χρησιμοποιήσαμε το official λογισμικό της εταιρίας, NOOBS, για το οποίο θα πούμε μερικά λόγια σε επόμενο κεφάλαιο. Στην συνέχεια, εγκατέστησα μερικά εργαλεία τα οποία αποτελούν σημαντικό παράγοντα για την εκτέλεση της εργασίας. Ο server μας πρέπει να υποστηρίζει και να είναι συμβατός με PHP και MySQL και φυσικά να είναι ενωμένος στο διαδίκτυο ώστε να είναι προσβάσιμος από οποιοδήποτε browser και να αποτελεί ένα ολοκληρωμένο σύστημα web server, παρακάτω θα γίνει ακριβής περιγραφή για το πως επιτεύχθηκε αυτό.

Για την εφαρμογή επέλεξα να χρησιμοποιήσω το Android Studio, για τον λόγο ότι μού είναι πιο οικείο και έχω ασχοληθεί στο παρελθόν με μικρό-εφαρμογές. Το Android Studio, είναι ένα πολύ δυνατό εργαλείο για Development εφαρμογών Android και έχει μία μεγάλη κοινότητα χρηστών, με αποτέλεσμα να υπάρχει σε μεγάλο βαθμό υποστήριξη.

1.1 Δομή Εργασίας

Στο δεύτερο κεφάλαιο, θα αναφερθούμε στα τεχνικά χαρακτηριστικά και εργαλεία πού χρησιμοποιήσαμε για την εργασία. Στην συνέχεια, στο κεφάλαιο 3 θα παρουσιάσουμε βήμα-βήμα την δημιουργία του server. Το κεφάλαιο 4, θα έχει αποκλειστικό σκοπό την

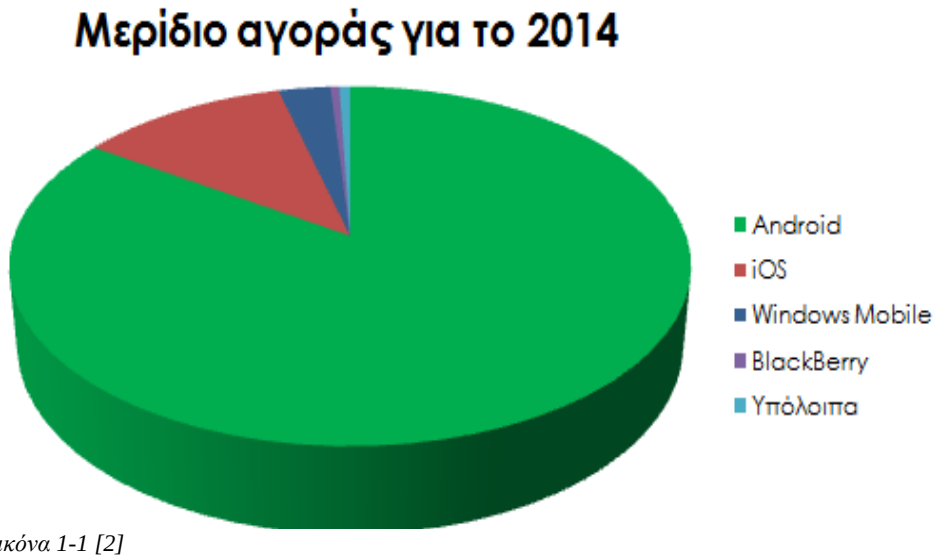
αναπαράσταση και παρουσίαση της βάσης δεδομένων μας, καθώς και μερικά σημαντικά στοιχεία, τα οποία μας βοηθούν να επιτεύξουμε την σύνδεση της εφαρμογής με την βάση. Το κεφάλαιο 5, θα αναλύει τα κομμάτια της εφαρμογής και τις δυνατότητες της, καθώς και με ποιο τρόπο επιτυγχάνετε ο σκοπός της εργασίας αυτής. Τέλος, θα αναφέρονται στην βιβλιογραφία όλες οι πηγές πού χρησιμοποίησα για να φέρω εις πέρας αυτή την εργασία.

1.2 Smartphones (“Εξυπνα τηλέφωνα”)

Το smartphone ή έξυπνο τηλέφωνο, είναι ένα κινητό τηλέφωνο πού βασίζεται σε ένα λειτουργικό σύστημα με περισσότερες υπολογιστικές ικανότητες σε σχέση με ένα απλό κινητό τηλέφωνο. Τα πρώτα έξυπνα τηλέφωνα συνδύαζαν τις λειτουργίες ενός προσωπικού ψηφιακού βοηθού (PDA) και ενός κινητού τηλεφώνου. Σε μεταγενέστερα μοντέλα συνδυάστηκαν οι λειτουργίες ψηφιακών φωτογραφιών, βιντεοκάμερες τσέπης, φορητών media players, καθώς και GPS, τα οποία αποτελούσαν μία πολύ-χρηστική συσκευή.

Τα πιο γνωστά λειτουργικά συστήματα που χρησιμοποιούνται από τα σύγχρονα έξυπνα τηλέφωνα είναι το Android της Google, το iOS της Apple, το BlackBerry OS της RIM, το Symbian της Nokia, τα Windows Phone της Microsoft, καθώς και διανομές Linux όπως το Ubuntu Phone μεταξύ αυτόν και πολλά άλλα.

Το μεγαλύτερο μερίδιο αγοράς για το 2014, χαρακτηρίζεται κυρίως από Android. [1]



Εικόνα 1-1 [2]

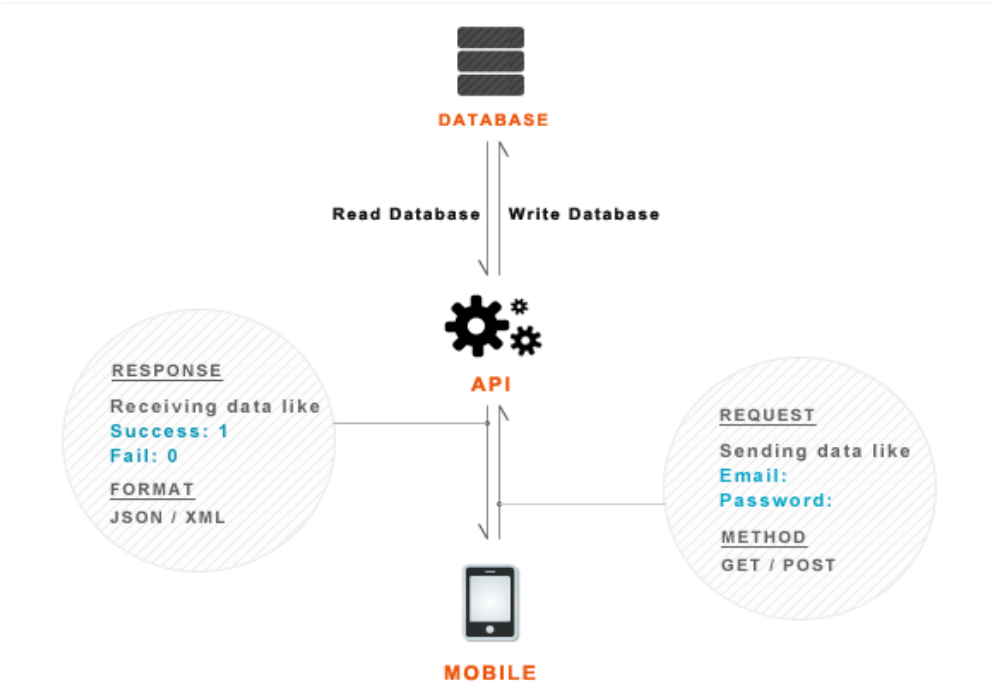
ΚΕΦΑΛΑΙΟ 2

2.1 Εισαγωγή

Σε αυτό το κεφάλαιο, θα περιγράψουμε τα εργαλεία που θα χρησιμοποιήσουμε για την πτυχιακή εργασία. Για να αλληλοεπιδράσουμε με την βάση δεδομένων, πρέπει πρώτα να χτίσουμε ένα REST API. Η δουλειά του θα είναι να παίρνει το αίτημα από το χρήστη και στην συνέχεια να επικοινωνεί με την βάση δεδομένων και να απαντάει στον χρήστη. Έτσι, εκτός από την επικοινωνία με τον server, επιτυγχάνουμε και την ασφάλεια των δεδομένων επικοινωνίας, αφού ο server είναι αυτός που αποφασίζει για τις απαντήσεις του. Για αυτό τον σκοπό, θα δημιουργήσουμε ένα απλό PHP, MySQL API.

Το API μας είναι υπεύθυνο για:

- Να αποδεχτεί τα αιτήματα σε μεθόδους POS/GET.
- Να αλληλοεπιδράσει με την βάση δεδομένων κάνοντας εισαγωγή ή ανάκτηση δεδομένων.
- Και τέλος να δώσει απάντηση σε μορφή JSON.



Σχετικό διάγραμμα επικοινωνίας:

2.2 “PHP” Τι είναι;

Το PHP ή αλλιώς Hypertext Preprocessor είναι μια πολύ διαδεδομένη γλώσσα λογισμικού, που είναι κατάλληλη για ανάπτυξη ιστοσελίδων και μπορεί να ενσωματωθεί σε HTML.

Για παράδειγμα:

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>Example</title>
    </head>
    <body>

        <?php
            echo "Hi, I'm a PHP script!";
        ?>

    </body>
</html>
```

Εικόνα 2-1 [3]

Εικόνα 2-2 [4]

Ο κώδικας PHP περικλείεται σε οδηγίες έναρξης και λήξης <? php και?> που μας επιτρέπουν να μεταβούμε μέσα και έξω από την λειτουργία του PHP.

Αυτό που διακρίνει την PHP από κάτι σαν το JavaScript από την πλευρά του χρήστη, είναι ότι ο κώδικας εκτελείται στον server, και στην συνέχεια αποστέλλεται στον client(“πελάτη”). Ο client, θα λάβει τα αποτελέσματα της εκτέλεσης του κώδικα, αλλά δεν θα ξέρει τι είναι ο υποκείμενος κώδικας.

Το πιο αναγνωρισμένο σημείο της PHP, είναι το ότι αποτελεί μια εξαιρετικά απλή γλώσσα για κάποιον νέο, παράλληλα όμως προσφέρει πολλά προηγμένα χαρακτηριστικά για έναν επαγγελματία προγραμματιστή. Παρόλο που η PHP επικεντρώνεται στην δημιουργία server-site, μπορούμε να την χρησιμοποιήσουμε και σε πολλές άλλες εφαρμογές. [4][5]

2.3 MySQL

Το MySQL είναι ένα δωρεάν σύστημα διαχείρισης και σχεδίασης βάσης δεδομένων που χρησιμοποιεί Structure Query Language (SQL).

Η SQL είναι η πιο διάσημη γλώσσα για την πρόσβαση, την προσθήκη και την διαχείριση περιεχομένου σε μια βάση δεδομένων. Είναι αρκετά γνώστη για την ταχύτητα που προσφέρει, την ευελιξία της και την αξιοπιστία της. Η MySQL αποτελεί σημαντικό κομμάτι σχεδόν κάθε εφαρμογής PHP.

Η MySQL μπορεί να τρέξει εικονικά σε όλες τις πλατφόρμες, συμπεριλαμβανομένου του Linux, UNIX και Windows. Παρόλο που μπορεί να χρησιμοποιηθεί σε πολλές εφαρμογές, τείνει να παρουσιάζει μεγαλύτερη προτίμηση σε περισσότερο διαδικτυακές εφαρμογές. [6]

Η SQL αναπτύχθηκε στην IBM από τους Andrew Richardson, Donald C. Messerly και Raymond F. Boyce, την δεκαετία του 1970. Η συγκεκριμένη έκδοση, που αποτελούνταν αρχικά από SEQUEL, είχε σκοπό να χειρίζεται και να ανακτά τα δεδομένα που αποθηκεύτηκαν στο πρώτο RDBMS της IBM , το System R. Το πρώτο εργαλείο διαχείρισης σχεσιακών βάσεων, ήταν το RDMBS που αναπτύχθηκε στο MIT, στις αρχές του 1970 και η Ingres, που αναπτύχθηκε το 1974 στο Πανεπιστήμιο Berkeley. Η Ingres εφάρμοσε μία γλώσσα διατύπωσης ερωτήσεων, γνωστή ως QUEL, το οποίο αντικαταστάθηκε αργότερα στην αγορά από την SQL. Προς το τέλος της δεκαετίας του 70, η Relational Software (τώρα Oracle Corporation) είδε τη δυνατότητα αυτών που περιεγράφηκαν από Codd, Chamberlin και Boyce και ανέπτυξε την SQL βασισμένη στο RDBMS, με τις φιλοδοξίες πώλησης του στο Αμερικανικό ναυτικό, την Κεντρική Υπηρεσία Πληροφοριών και άλλες Αμερικανικές Υπηρεσίες. Το καλοκαίρι του 1979, η Relational Software εισήγαγε την πρώτη διαθέσιμη στο εμπόριο εφαρμογή του SQL και νίκησε την IBM με τη διάθεση του πρώτου εμπορικού RDBMS για μερικές εβδομάδες.

Η SQL χωρίζετε σε διάφορα στοιχεία που περιλαμβάνουν:

- ***Expressions*** που μπορούν να κάνουν παραγωγή, είτε κλιμακωτές τιμές είτε πίνακες.
- ***Queries*** που κάνουν ανάκτηση στοιχείων και βασίζονται σε κάποια ειδικά κριτήρια.
- ***Statements*** που επιδρούν στα στοιχεία και στα σχήματα, ή έχουν την δυνατότητα να ελέγξουν τις συνδέσεις από άλλα προγράμματα και την ροή του προγράμματος.
- ***Predicates*** τα οποία διευκρινίζουν τούς όρους που μπορούν να αξιολογηθούν σαν σωστό ή λάθος.
- ***Clauses,*** τα οποία είναι προαιρετικά σε μερικές περιπτώσεις, αλλά απαραίτητα συστατικά των ερωτήσεων και δηλώσεων. [7]

2.4 Android

Το πρώτο εμπορικό Android λογισμικό που κυκλοφόρησε ήταν το Android 1.0 το οποίο κυκλοφόρησε τον Σεπτέμβριο του 2008. Το Android συνεχίζει να αναπτύσσετε από την Google και την Open Handset Alliance (OHA). Από τον Απρίλιο του 2009, οι εκδόσεις του Android έχουν θέμα ζαχαροπλαστικής στην κωδική ονομασία τους, και κυκλοφόρησαν σε αλφαβητική σειρά, εκτός των εκδόσεων 1,0 και 1,1 που δεν είχαν τεθεί υπό κωδικό όνομα.[8][9]

Κωδικό όνομα	Νούμερο έκδοσης	Ημερομηνία αρχικής κυκλοφορίας	Επίπεδο API
N/A	1.0	23 Σεπτεμβρίου 2008	1
	1.1	9 Φεβρουάριου 2009	2
Cupcake	1.5	27 Απριλίου 2009	3
Donut	1.6	15 Σεπτεμβρίου 2009	4
Eclair	2.0 – 2.1	26 Οκτωβρίου 2009	5–7
Froyo	2.2 – 2.2.3	20 Μαΐου 2010	8
Gingerbread	2.3 – 2.3.7	6 Δεκεμβρίου 2010	9–10
Honeycomb	3.0 – 3.2.6	22 Φεβρουάριου 2011	11–13
Ice Cream Sandwich	4.0 – 4.0.4	18 Οκτωβρίου 2011	14–15
Jelly Bean	4.1 – 4.3.1	9 Ιουλίου 2012	16–18
KitKat	4.4 – 4.4.4	31 Οκτωβρίου 2013	19–20
Lollipop	5.0 – 5.1.1	12 Νοεμβρίου 2014	21–22
Marshmallow	6.0 – 6.0.1	5 Οκτωβρίου 2015	23
Nougat	**7.0 - 7.1.1**	**22 Αυγούστου 2016**	**24**

Πίνακας 2-3 [8]

2.5 Android Studio

Το Android Studio είναι η επίσημη έκδοση περιβάλλοντος δημιουργίας (IDE) Integrated Development Environment για το Android λογισμικό της Google, βασισμένο στο JetBrains' IntelliJ IDEA software και σχεδιασμένο για να εξυπηρετεί τους Android Developers.

Το Android Studio ανακοινώθηκε 16 Μαΐου του 2013, στο συνέδριο Google I/O. Ήταν ακόμα σε αρχικά στάδια, ξεκινώντας από την έκδοση 0.1 τον Μάιο του 2013, στη συνέχεια προχώρησε σε Beta στάδιο ξεκινώντας από την έκδοση 0.8 η οποία εμπορικοποιήθηκε τον Ιούνιο του 2014. Η πρώτη σταθερή έκδοση αφέθηκε τον Δεκέμβριο του 2014 ξεκινώντας από την έκδοση 1.0. Σήμερα η σταθερή έκδοση είναι η 2.3.3 του Ιουνίου 2017. [10]

Η γλώσσα προγραμματισμού που χρησιμοποιεί είναι η Java και μπορεί να εγκατασταθεί στα λειτουργικά συστήματα Windows, macOS και Linux. Είναι ένα σχετικά απαιτητικό πρόγραμμα, αλλά με πάρα πολλά εργαλεία τα οποία προσφέρουν ευελιξία και αμέτρητες εφαρμογές.

Criterion	Description
OS version	Windows 7 or later Mac OS X 10.9.5 or later GNOME or KDE desktop
RAM	3 GB RAM minimum, 8 GB RAM recommended; plus 1 GB for the Android Emulator[17]
Disk space	500 MB disk space for Android Studio, at least 1.5 GB for Android SDK, emulator system images, and caches
Java version	Java Development Kit (JDK) 8
Screen resolution	1280x800 minimum screen resolution

Πίνακας 2-4 [10]

2.6 Raspberry Pi 3

Το Raspberry Pi είναι μία σειρά ολοκληρωμένου υπολογιστικού συστήματος, το οποίο αποτελείται από μία πλακέτα η οποία περιέχει όλα τα βασικά συστατικά ενός κανονικού υπολογιστικού συστήματος. Κατασκευάστηκε στην Αγγλία από την Raspberry Pi Foundation, για να προωθήσει τη τεχνική της βασικής επιστήμης των υπολογιστών στα σχολεία και σε αναπτυσσόμενες χώρες.

Σύμφωνα με την Raspberry Pi Foundation, πάνω από πέντε εκατομμύρια μικροϋπολογιστές έχουν πουληθεί μέχρι και το Φεβρουάριο του 2015, κάνοντάς το, το πιο διάσημο σε πωλήσεις βρετανικό υπολογιστή. Μέχρι το Νοέμβριο του 2016 πουλήθηκαν πάνω από έντεκα εκατομμύρια συσκευές, καθώς και τον Μάρτιο του 2017 πουλήθηκαν περισσότερα από δώδεκα εκατομμύρια υπολογιστές κάνοντάς το, το τρίτο καλύτερο σε πωλήσεις υπολογιστή γενικής χρήσης. [11]

Για τη συγκεκριμένη εργασία, προτίμησα το μοντέλο *RASPBERRY PI 3 MODEL B* το οποίο είναι το νεότερο μοντέλο στην αγορά, μέχρι και σήμερα. Οι προδιαγραφές του είναι οι εξής:

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 40-pin extended GPIO
- 4 USB 2 ports
- 4 Pole stereo output and composite video port
- Full size HDMI
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A [12]

Εικόνα 2-5 [12]

2.7 Noobs- Raspbian (raspberry pi)

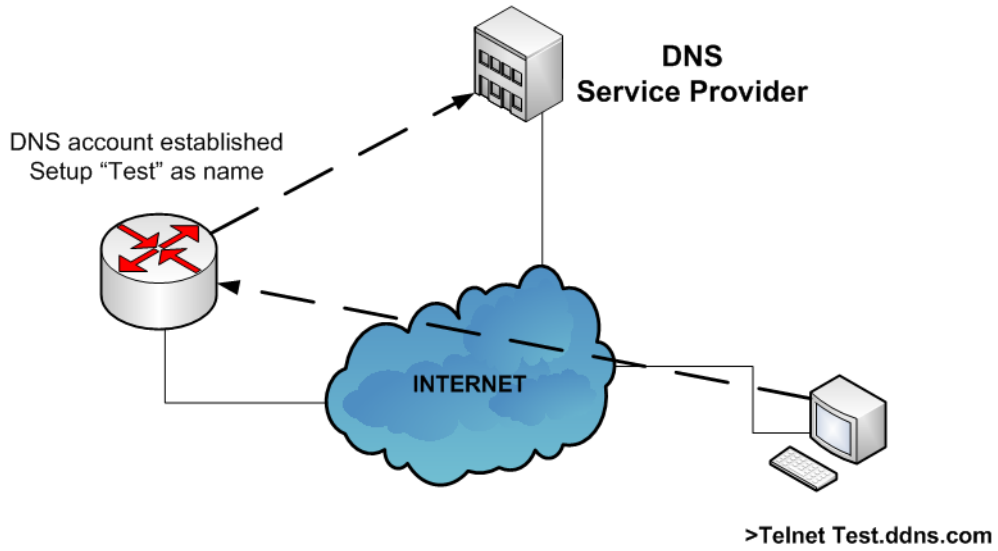
Το NOOBS είναι ένα εύκολο λογισμικό σύστημα εγκατάστασης το οποίο περιέχει το Raspbian λογισμικό, το οποίο και χρησιμοποιήσαμε για την παρούσα εργασία ως βασικό λογισμικό για τον Server μας. Επίσης, παρέχει ένα εύρος από διάφορα λογισμικά συστήματα τα οποία μπορεί κάποιος να τα κατεβάσει από το διαδίκτυο εκείνη τη στιγμή.[13]

Το Raspbian συμπεριλαμβάνει προ εγκατεστημένα προγράμματα για εκπαίδευση, προγραμματισμό και άλλες χρήσεις. Το Raspbian είναι βασισμένο σε Debian λογισμικό το οποίο είναι ένα Unix-like λογισμικό σύστημα που παρέχεται δωρεάν. Από το 2015 αποτελεί και επίσημα το βασικό λειτουργικό σύστημα για την οικογένεια των μικροϋπολογιστών Raspberry Pi.[14][20]

2.8 No-IP

Το No-IP ξεκίνησε τον Οκτώβριο του 2000 προσφέροντας ένα δωρεάν δυναμικό DNS και URL ανακατεύθυνσης (redirection). Οι χρήστες μπορούν να δημιουργούν sub-domain κάτω από κάποιους άλλους domains που τους προσφέρει το No-IP.

Το βασικό προϊόν του No-IP είναι οι δυναμικές υπηρεσίες DNS (“DDNS”). Οι βασικές δυναμικές υπηρεσίες (DNS) που χρησιμοποιούνται από το No-IP είναι δωρεάν για χρήση μέχρι ο λογαριασμός του χρήστη να είναι ενεργός. Η δωρεάν υπηρεσία επιτρέπει στους χρήστες να ρυθμίζουν μεταξύ ενός έως και τριών ονομάτων που παρέχονται από το No-IP. Στη συνέχεια, το όνομα του κεντρικού υπολογιστή θα επιλυθεί στη τρέχουσα διεύθυνση IP του χρήστη. Το No-IP παρέχει επίσης υπηρεσίες DNS, e-mail και παρακολούθησης δικτύου. [15]


DNS
Service Provider
DNS account established
Setup “Test” as name
INTERNET
>Telnet Test.ddns.com

Εικόνα 2-6 [16]

2.9 JAVA

Η Java είναι μια γλώσσα προγραμματισμού γενικής χρήσης, βασισμένη στην χρήση κλάσεων, η διαφορά της με άλλες γλώσσες είναι η ανεξαρτησία που προσφέρει στο λειτουργικό σύστημα και στην πλατφόρμα που την χρησιμοποιεί. Ο κώδικάς της Java μπορεί να τρέξει το ίδιο αποτελεσματικά σε Windows, Linux και Macintosh. Στο άμεσο μέλλον η Java θα μπορεί να τρέξει και σε κονσόλες παιχνιδιών, χωρίς να χρειάζεται κάποια αλλαγή στον κώδικα, για κάθε διαφορετικό λειτουργικό σύστημα. Για να επιτευχθεί κάτι τέτοιο πρέπει να τα προγράμματα που είναι γραμμένα σε java να μπορούν να διαβαστούν από κάθε υπολογιστή ανεξαρτήτως τύπου και είδος επεξεργαστή. Η δυσκολία σε αυτή την ιδέα είναι ότι η κεντρική μονάδα επεξεργασίας του επεξεργαστή, διαβάζει και ‘καταλαβαίνει’ σε διαφορετικό κώδικα μηχανής. Ένας τρόπος παράκαμψης του προβλήματος είναι η χρήση εικονικής μηχανής (Virtual Machine).

Η java είναι μια γλώσσα με πολλές δυνατότητες και ευελιξίες αλλά, ένα δυνατό χαρακτηριστικό της είναι η χρήση της σε εφαρμογές διαδικτυακές και ανεξαρτήτως μεγέθους. Έχει την ικανότητα να δημιουργεί δυναμικές ιστοσελίδες χρησιμοποιώντας το πρωτόκολλο επικοινωνίας HTTP, ανοίγοντας δρόμους για αμέτρητες δυνατότητες και χρήσεις. [17]

2.10 RESTful Web Services

Representational state transfer (REST) ή αλλιώς και RESTful web services είναι μια μέθοδος παροχής διαλειτουργικότητας (interoperability) μεταξύ διαφόρων υπολογιστικών συστημάτων στο Διαδίκτυο. Οι "Διαδικτυακοί πόροι" (Web resources) αρχικά, ορίστηκαν ως αρχεία ή έγγραφα που είναι προσβάσιμα μέσω των προσωπικών τους διευθύνσεων (URLs), σήμερα όμως έχουν ένα εντελώς διαφορετικό και γενικευμένο ορισμό, που περιλαμβάνει κάθε πράγμα ή οντότητα που μπορεί να αναγνωριστεί, να ονομαστεί ή να αντιμετωπιστεί με οποιοδήποτε τρόπο στο διαδίκτυο.

Σε μία RESTful διαδικτυακή υπηρεσία, τα αιτήματα που γίνονται στο URL μίας πηγής προκαλούν μία απάντηση πού μπορεί να είναι σε μορφή XML, HTML, JSON ή κάποια άλλη μορφή. Η απάντηση μπορεί να είναι μία επιβεβαίωση για παράδειγμα των αλλαγών που έγιναν στους πόρους της συγκεκριμένης διεύθυνσης. Χρησιμοποιώντας λοιπόν HTTP αιτήματα, το ίδιος λειτουργιών περιλαμβάνει εκείνα που είναι προκαθορισμένα από τις μεθόδους HTTP, GET, PUT, POST, DELETE κ.ο.κ.

Χρησιμοποιώντας λοιπόν τυποποιημένες λειτουργίες και stateless πρωτόκολλο, τα REST συστήματα καταφέρνουν να έχουν γρήγορη ανταπόκριση, αξιοπιστία και δυνατότητα εξέλιξης. Οι διαδικτυακές υπηρεσίες APIs που υποστηρίζουν την αρχιτεκτονική και τους περιορισμούς των REST, ονομάζονται RESTful API. Τα RESTful APIs που είναι βασισμένα σε HTTP ορίζονται με τις παρακάτω παραμέτρους:

- Έχουν διεύθυνση URL.
- Διαθέτουν ένα διαδικτυακό μέσο το οποίο ορίζει τα δεδομένα μεταφοράς.
- Διαθέτουν βασικές μεθόδους HTTP (POST, GET, DELETE, κ.α.).

ΚΕΦΑΛΑΙΟ 3

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα περιγράψουμε όλη την διαδικασία για να κάνουμε το raspberry pi μας Web Server. Αρχικά, θα ορίσουμε το raspberry pi ως τοπικό διακομιστή (Server) εγκαθιστώντας τον Apache, PHP, και MySQL. Αφού γίνει αυτό, ο Server μας μπορεί να μετατραπεί σε web server με την τεχνική προώθησης θύρας από το δρομολογητή (router) μας. Μετά την εκτέλεση της διαδικασίας προώθησης των θυρών (Port forwarding), το Raspberry Pi μας θα είναι προσβάσιμο μέσω διαδικτύου, εφόσον ο δρομολογητής μας θα το δείχνει.

3.2 Εγκατάσταση Apache, PHP, και MySQL για το Raspberry Pi

Προτού ξεκινήσουμε, πρέπει να σιγουρευτούμε ότι το λογισμικό μας (Raspbian) είναι πλήρες ενημερωμένο. Για να το κάνουμε αυτό, ανοίγουμε ένα τερματικό και πληκτρολογούμε την εντολή "*sudo apt-get update"*. Στη συνέχεια, θα ξεκινήσουμε να εγκαθιστάμε όλα τα απαραίτητα πακέτα για τον Server μας. Μπορούμε επίσης να πληκτρολογήσουμε στο τερματικό την εντολή "*sudo bash*", έτσι δεν θα χρειάζεται να γράφουμε την *'sudo'* εντολή συνέχεια και θα είμαστε συνεχώς root χρήστης.

Τώρα, για να εκτελεστούν όλες οι ενημερώσεις που κάναμε προηγουμένως για τα πακέτα, γράφουμε την εντολή "*apt-get upgrade*" και στην συνέχεια θα μας ρωτήσει αν θέλουμε να συνεχίσουμε, πληκτρολογούμε "*y*" και πατάμε enter. Η διαδικασία αυτή μπορεί να κρατήσει μερικά λεπτά ίσως και ώρες, ανάλογα το μέγεθος των πακέτων και την ταχύτητα του δικτύου που χρειάζονται για να ενημερωθούν.

Μόλις όλα τελειώσουν με την προηγούμενη διαδικασία, έχει σειρά η εγκατάσταση του Apache, ένα από τα βασικά συστατικά της εργασίας μας. Για αυτό το σκοπό χρησιμοποιούμε την εντολή "*apt-get install apache2 apache2-doc apache2-utils*".

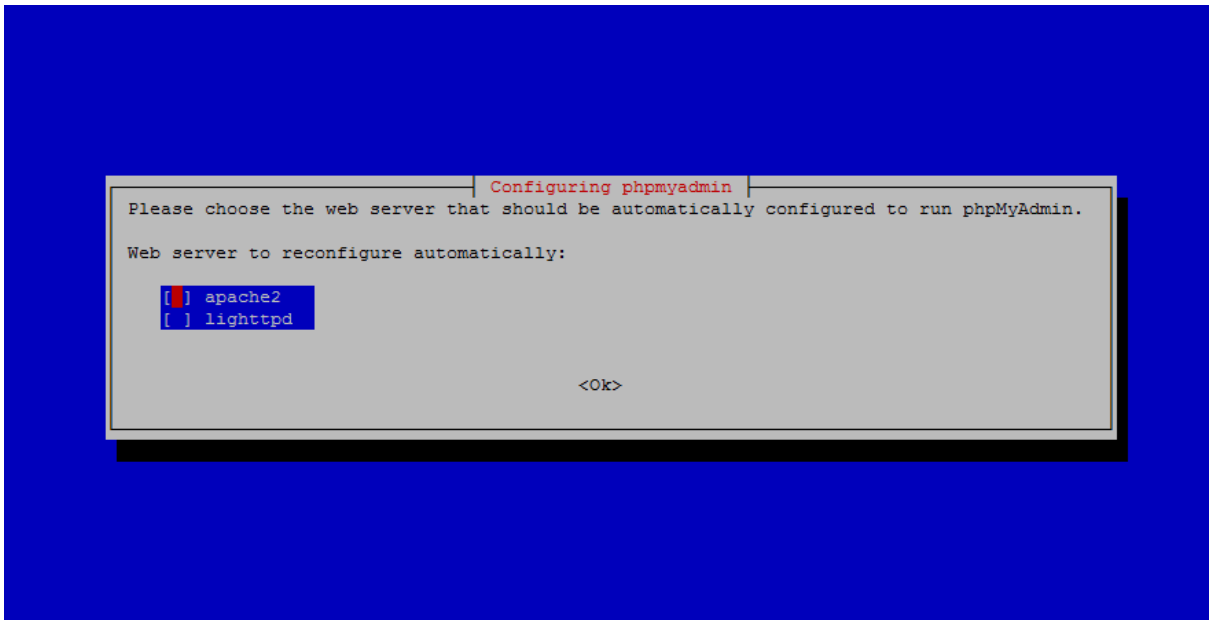
Συνεχίζοντας, όπως αναφέραμε και πιο πάνω, ο server μας πρέπει να υποστηρίζει PHP και MySQL. Επομένως, αρχικά πρέπει να κάνουμε εγκατάσταση την γλώσσα PHP. Αυτό επιτυγχάνεται με την εντολή "*apt-get install libapache2-mod-php5 php5 php-pear php5-xcache*". Τώρα πρέπει να εγκαταστήσουμε τα πακέτα σύνδεσης για την βάση δεδομένων (PHP->SQL), γράφοντας στο τερματικό " *apt-get install php5-mysql* ".

Είναι σημαντικό όμως να γίνει και η εγκατάσταση του MySQL server, ο οποίος αποθηκεύει τα δεδομένα μας στην βάση δεδομένων. Γράφουμε την εντολή "*apt-get install mysql-server mysql-client*". Κατά την διάρκεια αυτής της εγκατάστασης, μας ρωτάει για την εισαγωγή ενός κωδικού για τον server μας. Επομένως πληκτρολογούμε κάποιον κωδικό που θυμόμαστε. Τέλος, κάνουμε μία επανεκκίνηση το Raspberry pi μας και ελέγχουμε αν όλα δουλεύουν σωστά, γράφοντας 'localhost' ή '127.0.0.1' στον browser του Pi , και βλέποντας το παρακάτω αποτέλεσμα. [18]

Εικόνα 3-1

3.3 Εγκατάσταση του phpMyAdmin για το Raspberry Pi Web Server

Το phpMyAdmin είναι ένα δωρεάν εργαλείο γραμμένο σε PHP, που έχει σκοπό να χειρίζεται την διαχείριση του MySQL με την χρήση του web browser. Αυτό το καθιστά ευκολόχρηστο ως προς την διαχείριση της βάσης δεδομένων ακόμα και για κάποιον που δεν το έχει χρησιμοποιήσει ή δει ποτέ. Για να το εγκαταστήσουμε στο Raspberry Pi Web Server γράφουμε στο τερματικό "*sudo apt-get install phpmyadmin*". Αυτό θα ξεκινήσει την εγκατάσταση των πακέτων που χρειαζόμαστε. Κατά την εκτέλεση, μας ρωτάει τι είδους web server θα χρησιμοποιηθεί, επιλέγουμε 'apache2'.

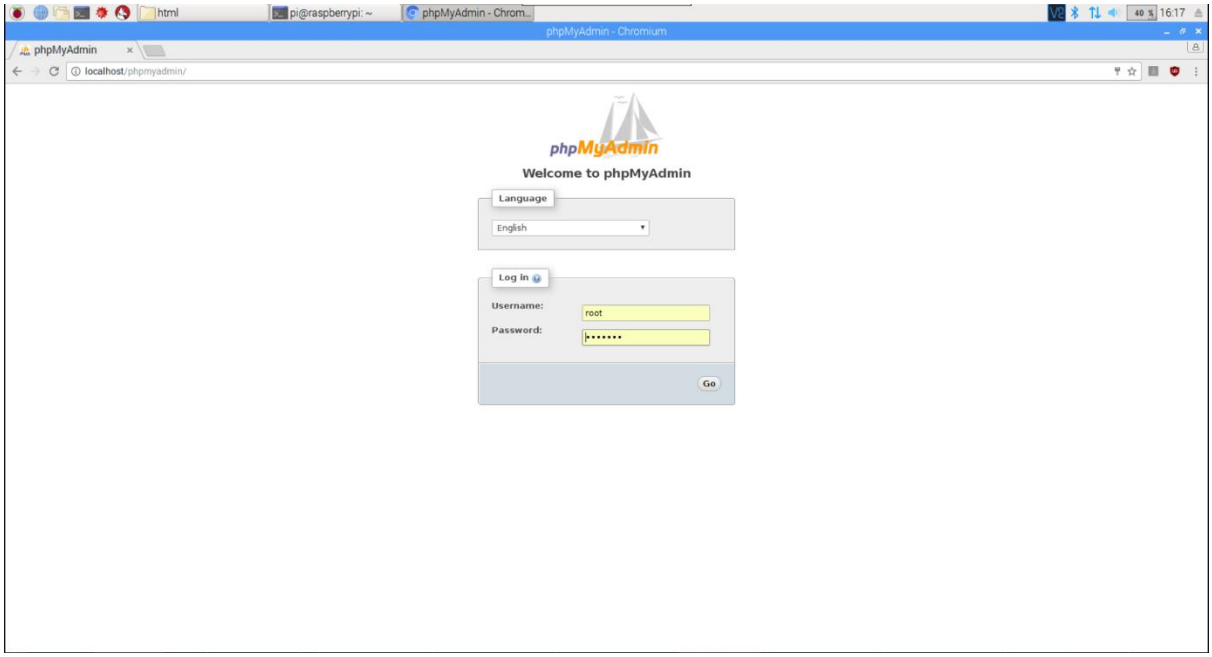


Εικόνα 3-2

Στην συνέχεια, πρέπει να διαμορφώσουμε τη βάση δεδομένων που θα χρησιμοποιήσει το phpMyAdmin, γι' αυτό επιλέγουμε 'YES ' όταν μας ζητηθεί. Εν συνεχεία, πρέπει να πληκτρολογήσουμε κωδικό διαχειριστή, εδώ πληκτρολογούμε τον κωδικό που εισήγαγε προηγουμένως για την MySQL βάση δεδομένων. Τώρα μας ζητάει έναν άλλο κωδικό. Αυτός ο κωδικός είναι για την πρόσβαση στο phpMyAdmin. Προσωπικά προτίμησα να βάλω τον ίδιο κωδικό, καθώς είναι ευκολότερο να το θυμάμαι. [18]

Τώρα πρέπει να ενώσουμε τον Apache με το phpMyAdmin. Για να γίνει αυτό, πρέπει να πειράξουμε το αρχείο ‘apache2.conf’. Έτσι, γράφουμε “*sudo nano /etc/apache2/apache2.conf*”.

Στο τέλος του αρχείου, προσθέτουμε την γραμμή ‘*Include /etc/phpmyadmin/apache.conf’*, αποθηκεύουμε και κάνουμε επανεκκίνηση τον server μας. Γράφοντας ‘localhost/phpmyadmin’ browser του Pi, βλέπουμε το παρακάτω αποτέλεσμα της εικόνας.



Εικόνα 3-3

Για username βάζουμε ‘root’, και για Password τον κωδικό που θέσαμε προηγουμένως κατά την εγκατάσταση του MySQL server. [18]

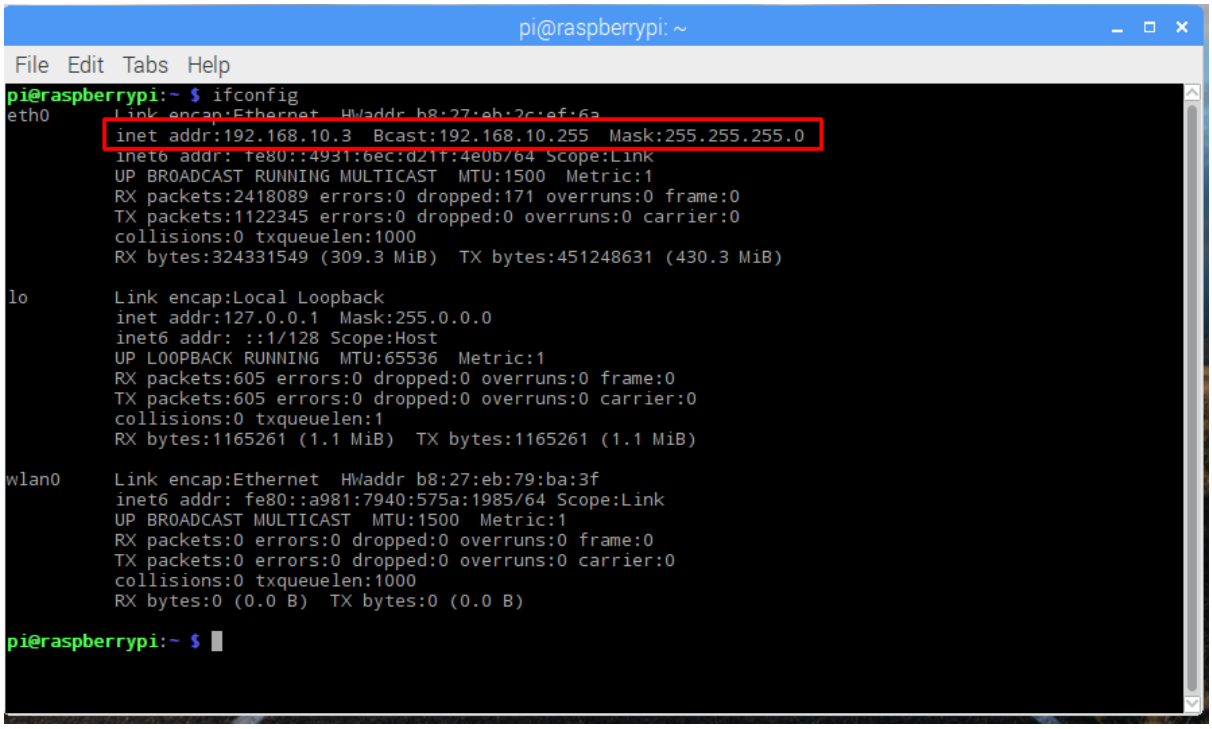
3.4 Static IP Address για το Raspberry Pi

Σε αυτό το σημείο, ακόμα δεν μπορούμε να έχουμε πρόσβαση στον server μας από διαφορετικό δίκτυο, μόνο από τοπικό. Επομένως, θα πρέπει να κάνουμε το server μας προσβάσιμο από οποιοδήποτε δίκτυο. Το Modem στο σπίτι μας, δίνει έναν αριθμό σε όλες τις συνδεμένες συσκευές σε αυτό, έτσι ώστε να μπορεί να της καταλάβει και να τις αναγνωρίσει. Αυτός ο αριθμός ονομάζετε *Internet Protocol* (IP) *Address.* Υπάρχουν δύο είδη από αυτά,

Internal IP Address (Εσωτερική διεύθυνση IP) και *External IP Address* (Εξωτερική διεύθυνση IP). Επίσης, το Modem παίρνει έναν αριθμό, ο οποίος ορίζετε από τον παροχέα υπηρεσιών διαδικτύου μας ή αλλιώς I*nternet Services Provider* (ISP), ο οποίος μπορεί να είναι για παράδειγμα η cyta, ote, forthnet ή κάποιος άλλος, για να μπορεί να αναγνωρίσει διάφορες συσκευές και συστήματα. Αυτός ο αριθμός είναι ο *External IP Address.*

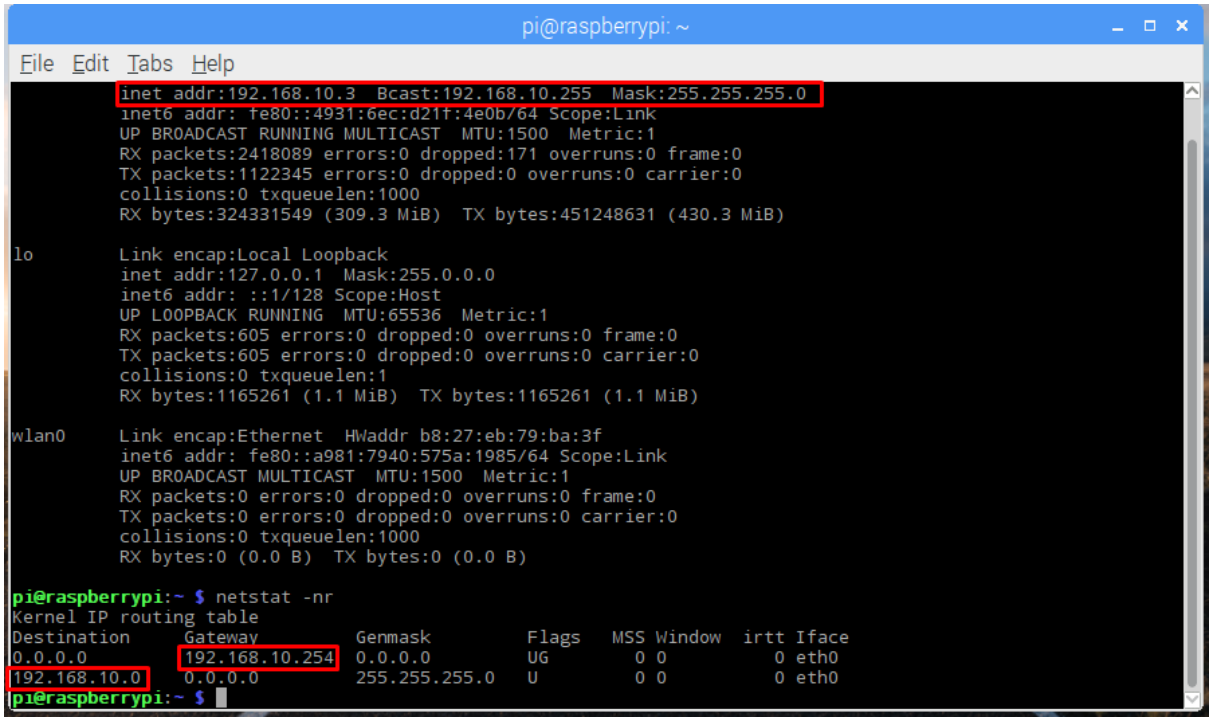
Εμείς θα μιλήσουμε για τον εσωτερικό αριθμό. Το Modem μας δίνει έναν τυχαίο αριθμό, συνήθως μοιάζει με αυτόν, 192.168.XXX.XXX. Αυτός ο τυχαίος αριθμός αναφέρεται με το όνομα DHCP. Χρειάζεται να το αλλάξουμε αυτό σε έναν σταθερό αριθμό (static IP address). Για αυτό το σκοπό καλούμαστε να αλλάξουμε ένα αρχείο στο Raspberry Pi μας, προτού όμως κάνουμε αυτό πρέπει να μαζέψουμε κάποιες πληροφορίες σχετικά με το δίκτυο μας.

Γράφοντας στο τερματικό “ifconfig” παίρνουμε τις παρακάτω πληροφορίες:



Εικόνα 0-1

Τώρα χρειαζόμαστε το gateway και destination address. Για αυτό εκτελούμε την εντολή "*netstat -nr*", παίρνοντας κάτι σαν το παρακάτω στιγμιότυπο (Χρειαζόμαστε τα σημειωμένα στοιχεία):



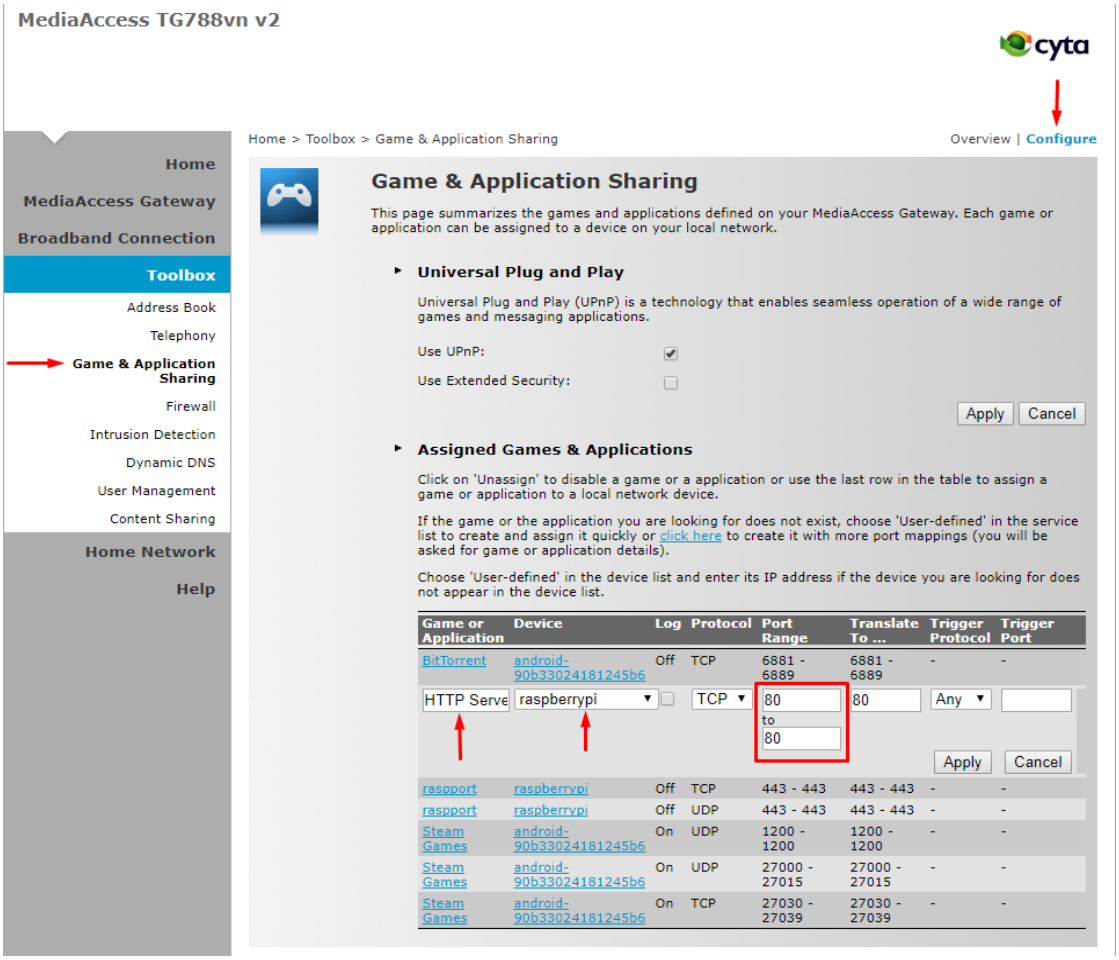
Εικόνα 0-3

Πάμε λοιπόν να επεξεργαστούμε τις διεπαφές δικτύου. Γράφουμε στο τερματικό "*sudo nano /etc/network/interfaces*", και βρίσκουμε την γραμμή που λέει 'iface eth0 inet dhcp' και την αλλάζουμε σε " iface eth0 inet static". Στη συνέχεια, κάτω από την γραμμή που αλλάξαμε βάζουμε τα στοιχεία που συλλέξαμε προηγουμένως, (βλέπε εικόνα 3.6-2). Τέλος, κάνουμε αποθήκευση των αλλαγών μας (Ctr+X). [18]

3.5 Ρύθμιση της προώθησης θύρας και του DDNS στον διακομιστή Web Raspberry Pi

Εικόνα 0-2

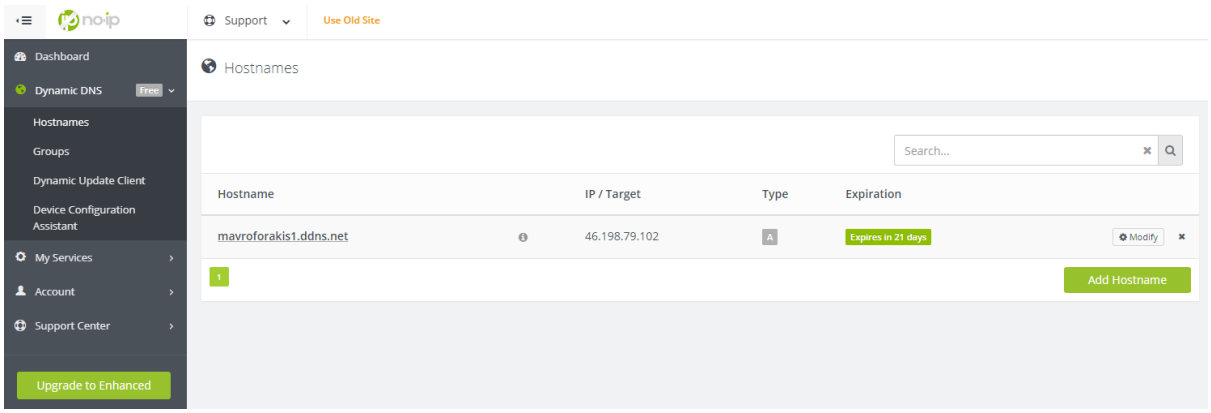
Σε αυτό το σημείο πρέπει να κάνουμε αλλαγές εκτός του Raspberry Pi, θα ανοίξουμε την θύρα 80 για να κάνουμε το Raspberry Pi προσβάσιμο από οποιοδήποτε δίκτυο. Για να ανοίξουμε κάποια θύρα στο δίκτυο μας πρέπει να έχουμε πρόσβαση στο Modem του σπιτιού μας. Ανοίγουμε το browser του υπολογιστή μας ή του Raspberry Pi, και πληκτρολογούμε την διεύθυνση 'broadcast', που βρήκαμε προηγουμένως. Στην δική μου περίπτωση, η εισαγωγή της πόρτας για το raspberry pi είναι όπως της εικόνας παρακάτω. Αυτό, αλλάζει ανάλογα με την έκδοση λογισμικού και εταιρία κατασκευής του Modem, αλλά η μέθοδος παραμένει η ίδια. Επιλέγουμε HTTP Server που αντιπροσωπεύει την πόρτα 80 και το όνομα της συσκευής μας που θα εμφανιστεί στο dropdown στην στήλη Device, γράφουμε την θύρα που θέλουμε να ανοίξουμε και πατάμε Apply. [18]



Εικόνα 3.5-1

Κατόπιν ρυθμίζουμε το DDNS, το οποίο αναφέρετε σε Dynamic Domain Name Setup. Χρησιμοποίησα μια σελίδα που λέγετε No-IP, η οποία μας βοηθάει στο να ορίσουμε ένα URL για την διεύθυνση του Modem μας, στο οποίο ανοίξαμε την πόρτα 80 και την κατευθύναμε στο Raspberry Pi.

Αφού δημιουργήσουμε έναν λογαριασμό, το site μας καθοδήγει στις ενέργειες και στα βήματα που πρέπει να προβούμε για να θέσουμε ένα domain όνομα για τον server μας. [18] [19]



Εικόνα 3.5-2

ΚΕΦΑΛΑΙΟ 4

4.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναλύσουμε και θα περιγράψουμε την διαδικασία δημιουργίας βάσης δεδομένων με την βοήθεια του εργαλείου phpMyAdmin το οποίο και έχουμε εγκαταστήσει στον server μας προηγουμένως.

Αφού έχουμε φτάσει μέχρι εδώ ο server είναι online και το δικό μου domain όνομα είναι "*mavroforakis1.ddns.net*" επομένως, για να μπούμε στο phpMyAdmin γράφουμε στο browser " http://mavroforakis1.ddns.net/phpmyadmin/", κάνουμε login και ξεκινάμε την επεξεργασία της βάσης. Επίσης, πρέπει να αναφέρουμε ότι η συγκεκριμένη βάση που δημιούργησα έχει ως σκοπό να επιδείξει κάποια βασικά χαρακτηρίστηκα των βάσεων δεδομένων και να καταστίσει λειτουργική την εφαρμογή που θα περιγράψουμε στο επόμενο κεφάλαιο. Για την χρήση της ως εμπορική εφαρμογή απαιτεί περισσότερα δεδομένα.

4.2 Δημιουργία βάσης και πινάκων

Για να δημιουργήσουμε μια καινούργια βάση, πάμε από τα δεδομένα δομής στα αριστερά της σελίδας και κάνουμε κλικ στο 'Νέα'. Στη συνέχεια, βάζουμε το όνομα της βάσης δεδομένων μας και πατάμε 'εκτέλεση'. Το όνομα της βάσης μου είναι 'UserDB' και περιέχει όλους τους βασικούς πίνακες που χρειαζόμαστε για την εφαρμογή.

Ο πρώτος πίνακας που φτιάχνουμε είναι ο 'Users' ο οποίος περιέχει τα δεδομένα όλων των χρηστών του συστήματος μας. Τα πεδία του πίνακα μας είναι τα εξής:

1. **Type**: όπου παίρνει τιμές 0 ή 1 για να μπορούμε να ξεχωρίσουμε αν είναι απλός χρήστης (0) ή αν είναι διαχειριστής (1) και είναι τύπου 'tinyint'.
2. **UserID**: που είναι auto increment και είναι τύπου 'tinyint'.
3. **LastName**: που θα περιέχει το επίθετο της εγγραφής, τύπου 'char'.
4. **FirstName**: που θα περιέχει το όνομα της εγγραφής, τύπου 'char'.
5. **UserName**: που περιέχει το username του χρήστη, τύπου 'varchar'.

6. **Email**: για το email του χρήστη, τύπου ‘varchar’.
7. **Password**: για κωδικό σύνδεσης, τύπου ‘varchar’.
8. **Id_Storage**: που περιγράφει σε ποια αποθήκη ανήκει ο χρήστης, τύπου ‘tinyint’.

Ο δεύτερος πίνακας είναι ο ‘Storages’, που είναι υπεύθυνος να αποθηκεύει τις διάφορες αποθήκες που ανήκουν στην παρούσα εταιρία. Για παράδειγμα, αποθήκη Ηρακλείου και αποθήκη Αθήνας, καθώς πολλές εταιρίες έχουν περισσότερες από μία αποθήκη για προϊόντα ή ακόμα και διάφορα καταστήματα σε διαφορετικά μέρη. Ο πίνακας έχει τα εξής πεδία:

1. **id**: που είναι auto increment και τύπου ‘tinyint’.
2. **Sname**: για το όνομα της αποθήκης, τύπου ‘varchar’.
3. **Saddress**: για την διεύθυνση της αποθήκης, τύπου ‘varchar’.

Ο τρίτος πίνακας είναι ο ‘Shelfs’, υπεύθυνος για τα ράφια της αποθήκης, που περιέχουν τα προϊόντα μας, έχοντας πεδία του τα εξής:

1. **Id**: που είναι auto increment και τύπου ‘tinyint’.
2. **Sname**: για το όνομα του ραφιού, τύπου ‘varchar’.
3. **Sdiscription**: όπου περιγράφουμε το προϊόν, για παράδειγμα Laptop,TVs, κ.ά., τύπου ‘varchar’.
4. **Id_Storage**: που περιγράφει σε ποια αποθήκη ανήκει το ράφι, τύπου ‘tinyint’.

Ο τέταρτος πίνακας μας περιέχει τα προϊόντα της αποθήκης και ονομάζεται ‘Products’ και αποτελείτε από:

1. **id**: που είναι auto increment και τύπου ‘tinyint’.
2. **Pnumber**: για τον αριθμό προϊόντος που είναι μοναδικός, τύπου ‘varchar’.
3. **Pname**: για το όνομα προϊόντος (π.χ. ASUS), τύπου ‘varchar’.
4. **Pmodel**: για το μοντέλο προϊόντος (π.χ. Predator Z97), τύπου ‘varchar’.
5. **Pqnt**: που περιγράφει την ποσότητα σε stock στο ράφι μας, τύπου ‘int’.
6. **Id_shelf**: που αναφέρετε σε ποιο ράφι ανήκει το κάθε προϊόν, τύπου ‘tinyint’.

Στην συνέχεια, ο πέμπτος και προ τελευταίος πίνακας μας είναι εκείνος που αποθηκεύουμε τις παραγγελίες μας και δείχνει όλες της παραγγελίες από όλους τους αποθηκάριους. Περιέχει τα εξής πεδία:

1. **id**: που είναι auto increment και τύπου ‘tinyint’.
2. **ProductNumber**: που αναφέρετε στον αριθμό προϊόντος και είναι τύπου ‘varchar’.
3. **Qnt**: περιγράφει την ποσότητα των κομματιών παραγγελίας, τύπου ‘int’.
4. **Status**: που περιγράφει την τρέχουσα κατάσταση της παραγγελίας, αν δηλαδή έχει ολοκληρωθεί ή αν ακόμα είναι υπό εκτέλεση, τύπου ‘varchar’.

Τέλος, χρειαζόμαστε ένα πίνακα για να αποθηκεύουμε τις κινήσεις που γίνονται από τους χρήστες, τον ‘LogFile’, έτσι ώστε ο διαχειριστής να μπορεί να ελέγχει και να καταγράφει το τί κάνει ο κάθε υπάλληλος του (αποθηκάριος). Ο πίνακας αυτός αποτελείτε από τα εξής πεδία:

1. **id**: που είναι auto increment και τύπου ‘tinyint’.
2. **LoginUser**: όπου περιγράφει το «ποιος» σε αυτή την εκτέλεση, τύπου ‘varchar’.
3. **Action**: όπου περιγράφει το «τι έκανε» (π.χ. Διαγραφή, Login), τύπου ‘varchar’.
4. **Description**: το οποίο είναι εφεδρικό πεδίο αλλά το χρησιμοποιούμε για να περιγράφουμε αν ήταν επιτυχής ή αποτυχής η δράση του χρήστη, τύπου ‘text’.
5. **Date_Time**: που περιγράφει την ώρα και ημερομηνία της δράσης που παρουσιάστηκε, τύπου ‘timestamp’.

4.3 Συσχετίσεις πινάκων

Σε κάθε σειρά ενός πίνακα, μπορεί να οριστεί ένα μοναδικό κλειδί. Η κάθε σειρά σε ένα πίνακα έχει την δυνατότητα να αναφέρει γραμμές από άλλους πίνακες σε μία βάση δεδομένων, προσθέτοντας μια στήλη για το μοναδικό κλειδί της αναφερόμενης γραμμής. Αυτές οι στήλες, είναι γνωστές ως εξωτερικά ή ξένα κλειδιά (Foreign Keys).

Ένα μέρος αυτής της διαδικασίας συμπεριλαμβάνει την συνεχή δυνατότητα τροποποίησης ή επιλογής μιας μόνο σειράς, σε ένα πίνακα. Συνεπώς, οι πιο πολλές εφαρμογές έχουν ένα μοναδικό πρωτεύων κλειδί (PK) για κάθε πίνακα. Κατά τη δημιουργία μιας νέας σειράς σε κάποιο πίνακα, δημιουργείτε και μία μοναδική τιμή για το πρωτεύων κλειδί του πίνακα, το οποίο είναι αυτό που χρησιμοποιείτε για την πρόσβαση σε αυτόν.

Τα πρωτεύοντα κλειδιά σε μία βάση δεδομένων χρησιμοποιούνται για να προσδιορίσουν τις σχέσεις μεταξύ πινάκων. Όταν ένα πρωτεύων κλειδί μεταφερθεί ή ‘μεταναστεύσει’ σε ένα άλλο πίνακα , τότε ορίζεται ως ξένο κλειδί του άλλου πίνακα, αυτό το σχεσιακό μοτίβο μπορεί να εκφραστεί είτε ως σχέση ένα προς ένα, είτε ως σχέση ένα προς πολλά. Επιπρόσθετα, πολλές βάσεις δεδομένων παρουσιάζουν περιπτώσεις πολλά προς πολλά, δημιουργώντας έναν πίνακα που εμπεριέχει όλα τα σχεσιακά κλειδιά (και των δύο οντοτήτων) των υπολοίπων πινάκων.

Για παράδειγμα, έστω ότι έχουμε μία βάση δεδομένων με δύο πίνακες. Ένας πίνακας είναι ο ‘Προμηθευτές’ που περιέχει όλα τα δεδομένα των προμηθευτών για τα προϊόντα μας και έναν πίνακα ‘Παραγγελίες’ που εκφράζει όλες τις παραγγελίες που έχουν γίνει για κάθε προμηθευτή. Υποθέτουμε ότι η επιχείρηση χρειάζεται κάθε παραγγελία να αναφέρεται σε έναν και μόνο προμηθευτή που μας φέρνει την παραγγελία. Για την απεικόνιση αυτού του μοτίβου στην βάση δεδομένων, βάζουμε μία έξτρα στήλη ξένου κλειδιού στον πίνακα ‘Παραγγελίες’ (π.χ. id_prom), όπου γίνετε αναφορά στο κλειδί του προμηθευτή (π.χ. id). Αφού, το πρωτεύων κλειδί κάποιου πίνακα χρειάζεται να είναι μοναδικό και επειδή το

‘id_prom’ έχει τιμές μόνο από τα πρωτεύοντα κλειδιά, λέμε ότι όταν έχει κάποια τιμή, το id_prom’ θα αναγνωρίσει τον συγκεκριμένο προμηθευτή που ανήκει η συγκεκριμένη παραγγελία από id του πίνακα παραγγελιών. Αυτό είναι ένα παράδειγμα σχέσης ένα προς ένα (σε κάθε προμηθευτή ανήκει μία παραγγελία). Αντίθετα, αν είχαμε έναν προμηθευτή που δεχόταν πολλές παραγγελίες από την επιχείρηση μας τότε το σχεσιακό μοτίβο θα ήταν ένα προς πολλά. [21]

4.4 Χρήσιμα Queries

Εδώ θα αναφέρουμε κάποια χρήσιμα ερωτήματα (Queries) για την MySQL βάση και τις βασικές τους λειτουργίες. Αυτά τα ερωτήματα τα εκφράζουμε σε php κώδικα και με την βοήθεια τους επιτυγχάνουμε τη διαχείριση της βάσης δεδομένων μας. Στα παρακάτω παραδείγματα, εκφράζεται ο βασικός κώδικας για την εκάστοτε εργασία και όχι η εκτελέσιμη μορφή, αφού αναφερόμαστε με παραδείγματα.

- Πριν να έχουμε πρόσβαση στα δεδομένα της βάσης, πρέπει να επιχειρήσουμε να συνδεθούμε με την βάση. Αυτό επιτυγχάνετε με τον εξής τρόπο:

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);



?>
```

Εικόνα 4.4-1 [22]

- Για να μπορούμε να συνδεθούμε όμως σε μία βάση πρέπει πρώτα να την δημιουργήσουμε, αυτό επιτυγχάνετε ως εξής:

```
// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . $conn->error;
}
```

Εικόνα 4.4-2 [22]

- Για την δημιουργία πίνακα στην βάση δεδομένων που φτιάξαμε, εκτελούμε κάτι σαν αυτό:

```
CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)
```

Εικόνα 4.4-3 [22]

- Μετά από την δημιουργία βάσης και πίνακα μπορούμε να ξεκινήσουμε να βάζουμε στον πίνακα μας δεδομένα:

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

Εικόνα 4.4-4 [22]

- Ο όρος SELECT χρησιμοποιείτε για την επιλογή στοιχείων από κάποιο πίνακα και ο όρος '*' χρησιμοποιείτε για την επιλογή όλων τον πεδίων του

```
SELECT column_name(s) FROM table_name
```

4.4-5 [22]

πίνακα:

```
SELECT * FROM table_name
```

Εικόνα 4.4-6 [22]

- Για την διαγραφή στοιχείων από κάποιο πίνακα χρησιμοποιείτε ο όρος

```
DELETE FROM table_name
WHERE some_column = some_value
```

Εικόνα 4.4-7 [22]

‘DELETE’:

- Επίσης υπάρχει ο όρος ‘UPDATE’ που κάνει ενημέρωση των στοιχείων

κάποιου πίνακα:

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

Εικόνα 4.4-8 [22]

Για την δημιουργία των πινάκων στην βάση δεδομένων χρησιμοποιήσαμε την εξής μορφή queries αναλυτικά:

```
--
-- Δομή πίνακα για τον πίνακα `LogFile`
--

CREATE TABLE IF NOT EXISTS `LogFile` (
`id` int(11) NOT NULL,
  `LoginUser` varchar(30) NOT NULL,
  `Action` varchar(30) NOT NULL,
  `Description` text NOT NULL,
  `Date_Time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
) ENGINE=InnoDB AUTO_INCREMENT=35 DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Δομή πίνακα για τον πίνακα `Orders`
--

CREATE TABLE IF NOT EXISTS `Orders` (
`id` tinyint(4) NOT NULL,
  `ProductNumber` varchar(30) DEFAULT NULL,
  `Qnt` int(11) NOT NULL,
  `Status` varchar(30) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=19 DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Δομή πίνακα για τον πίνακα `Products`
--

CREATE TABLE IF NOT EXISTS `Products` (
`id` tinyint(4) NOT NULL,
  `Pnumber` varchar(16) NOT NULL,
  `Pname` varchar(20) NOT NULL,
  `Pmodel` varchar(20) NOT NULL,
  `Pqnt` int(11) NOT NULL,
  `id_shelf` tinyint(4) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=13 DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Δομή πίνακα για τον πίνακα `Shelfs`
--

CREATE TABLE IF NOT EXISTS `Shelfs` (
`id` tinyint(4) NOT NULL,
  `Sname` varchar(5) NOT NULL,
  `Sdiscription` varchar(30) NOT NULL,
  `id_Storage` tinyint(4) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=latin1;
```

Εικόνα 4.4-9

```
--
-- Δομή πίνακα για τον πίνακα `Storages`
--

CREATE TABLE IF NOT EXISTS `Storages` (
`id` tinyint(4) NOT NULL,
  `Sname` varchar(5) NOT NULL,
  `Saddress` varchar(30) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Δομή πίνακα για τον πίνακα `Users`
--

CREATE TABLE IF NOT EXISTS `Users` (
  `Type` tinyint(4) NOT NULL,
`UserID` tinyint(4) NOT NULL,
  `LastName` char(36) NOT NULL,
  `FirstName` char(36) NOT NULL,
  `UserName` varchar(36) NOT NULL,
  `Email` varchar(45) NOT NULL,
  `Password` varchar(35) NOT NULL,
  `id_Storage` tinyint(4) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=59 DEFAULT CHARSET=utf8;

--
-- Ευρετήρια για άχρηστους πίνακες
--

--
-- Ευρετήρια για πίνακα `LogFile`
--
ALTER TABLE `LogFile`
 ADD PRIMARY KEY (`id`);

--
-- Ευρετήρια για πίνακα `Orders`
--
ALTER TABLE `Orders`
 ADD PRIMARY KEY (`id`);

--
-- Ευρετήρια για πίνακα `Products`
--
ALTER TABLE `Products`
 ADD PRIMARY KEY (`id`), ADD UNIQUE KEY `Pnumber` (`Pnumber`), ADD KEY `id_shelf` (`id_shelf`);

--
-- Ευρετήρια για πίνακα `Shelfs`
--
ALTER TABLE `Shelfs`
 ADD PRIMARY KEY (`id`), ADD KEY `id_Storage` (`id_Storage`);
```

Εικόνα 4.4-10

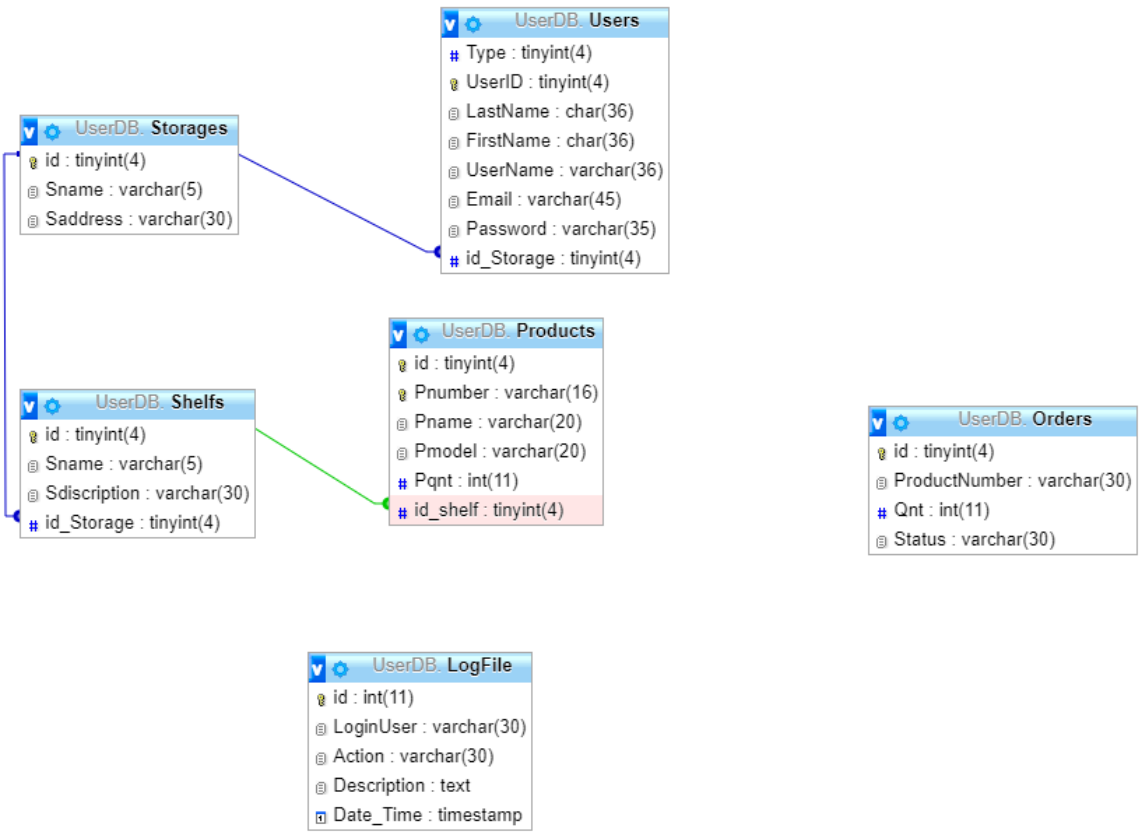
```
--
-- Ευρετήρια για πίνακα `Storages`
--
ALTER TABLE `Storages`
 ADD PRIMARY KEY (`id`);

--
-- Ευρετήρια για πίνακα `Users`
--
ALTER TABLE `Users`
 ADD PRIMARY KEY (`UserID`), ADD KEY `id_Storage` (`id_Storage`);

--
-- AUTO_INCREMENT για άχρηστους πίνακες
--

--
-- AUTO_INCREMENT για πίνακα `LogFile`
--
ALTER TABLE `LogFile`
MODIFY `id` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=35;
--
-- AUTO_INCREMENT για πίνακα `Orders`
--
ALTER TABLE `Orders`
MODIFY `id` tinyint(4) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=19;
--
-- AUTO_INCREMENT για πίνακα `Products`
--
ALTER TABLE `Products`
MODIFY `id` tinyint(4) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=13;
--
-- AUTO_INCREMENT για πίνακα `Shelfs`
--
ALTER TABLE `Shelfs`
MODIFY `id` tinyint(4) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=7;
--
-- AUTO_INCREMENT για πίνακα `Storages`
--
ALTER TABLE `Storages`
MODIFY `id` tinyint(4) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=3;
--
-- AUTO_INCREMENT για πίνακα `Users`
--
ALTER TABLE `Users`
MODIFY `UserID` tinyint(4) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=59;
```

Εικόνα 4.4-11

```
--
-- Περιορισμοί για άχρηστους πίνακες
--

--
-- Περιορισμοί για πίνακα `Products`
--
ALTER TABLE `Products`
ADD CONSTRAINT `Products_ibfk_1` FOREIGN KEY (`id_shelf`) REFERENCES `Shelfs` (`id`) ON DELETE CASCADE;

--
-- Περιορισμοί για πίνακα `Shelfs`
--
ALTER TABLE `Shelfs`
ADD CONSTRAINT `Shelfs_ibfk_1` FOREIGN KEY (`id_Storage`) REFERENCES `Storages` (`id`) ON DELETE CASCADE;

--
-- Περιορισμοί για πίνακα `Users`
--
ALTER TABLE `Users`
ADD CONSTRAINT `Users_ibfk_1` FOREIGN KEY (`id_Storage`) REFERENCES `Storages` (`id`) ON DELETE CASCADE;
```

Εικόνα 4.4-12

4.5 Γραφική απεικόνιση βάσης δεδομένων



Εικόνα 4.5-13

Σε αυτό το διάγραμμα γίνετε γραφική απεικόνιση της βάσης δεδομένων μας, με όλα τα χρήσιμα χαρακτηριστικά της. Οι γραμμές που ενώνουν τους πίνακες μας αναπαριστούν τις συσχετίσεις μεταξύ των πινάκων. Μεταξύ του πίνακα ‘Users’ και του πίνακα ‘Storages

παρατηρούμε μια μπλε γραμμή η οποία συσχετίζει το πεδίο ‘id’ του πίνακα storages και το πεδίο ‘id_Storage’. Όπως αναφέραμε στο παράδειγμα συσχετίσεων παραπάνω, υπάρχουν δύο βασικές σχεσιακές μορφές, ένα προς ένα και ένα προς πολλά. Στο συγκεκριμένο σημείο, η γραμμή αυτή περιγράφει την περίπτωση του ένα προς πολλά, που φαίνεται στο δεξιό άκρο της μπλε γραμμής και μοιάζει με ημικύκλιο. Το λογικό σχεσιακό σκεπτικό είναι ότι μια αποθήκη μπορεί να έχει πολλούς χρήστες, δηλαδή πολλούς αποθηκάριους.

Μεταξύ των πινάκων ‘Storages’ και ‘Shelfs’, βλέπουμε το ημικύκλιο στο άκρο να βρίσκετε στο πεδίο του πίνακα shelfs δείχνοντας το πεδίο ‘id_Storage’, γιατί μία αποθήκη μπορεί να έχει πολλά ράφια αλλά, η αντίθετη προϋπόθεση δεν υφίσταται ικανή. Παρομοίως για τους πίνακες Shelfs και Products ισχύει ότι ένα ράφι μπορεί να έχει πολλά προϊόντα.

ΚΕΦΑΛΑΙΟ 5

5.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναλύσω αρχικά το API της εφαρμογής και την λειτουργία του. Μετέπειτα, θα αναφερθώ στην περιγραφή της εφαρμογής.

Η εφαρμογή αυτή δημιουργήθηκε με σκοπό ένας αποθηκάριος να μπορεί να διαχειριστεί τα προϊόντα της αποθήκης του απομακρυσμένα· για παράδειγμα, με την βοήθεια κάποιου smartphone που θα έχει στην διάθεση του, χωρίς να χρειάζεται να βρίσκεται στον κεντρικό υπολογιστή. Ας φανταστούμε ένα σενάριο ενός αποθηκάριου σε μία τεράστια αποθήκη με προϊόντα τεχνολογίας μίας εταιρίας e-shop. Ο αποθηκάριος αυτός πρέπει να γνωρίζει πόσα προϊόντα έχει σε stock στην αποθήκη του και που βρίσκεται το καθένα. Θα πρέπει λοιπόν να έχει την δυνατότητα αναζήτησης οποιουδήποτε προϊόντος και να βλέπει σε ποιο ράφι βρίσκετε για να μπορέσει να προβεί σε όποια ενέργεια επιθυμεί. Από την άλλη μεριά έχουμε και τον υπεύθυνο της αποθήκης, που μπορεί να είναι κάποιος υπάλληλος ή ακόμα και ο ίδιος ο ιδιοκτήτης. Αυτός, θα πρέπει να έχει την δυνατότητα όλων των ενεργειών ενός αποθηκάριου όπως και κάποιες μοναδικές για αυτόν επιλογές όπως, η πρόσληψη ή η απόλυση κάποιου άλλου αποθηκάριου. Θα πρέπει επίσης, να είναι ικανός να παρακολουθεί όλες τις ενέργειες των αποθηκάριων για να υπάρχει κάποιος έλεγχος και τάξη.

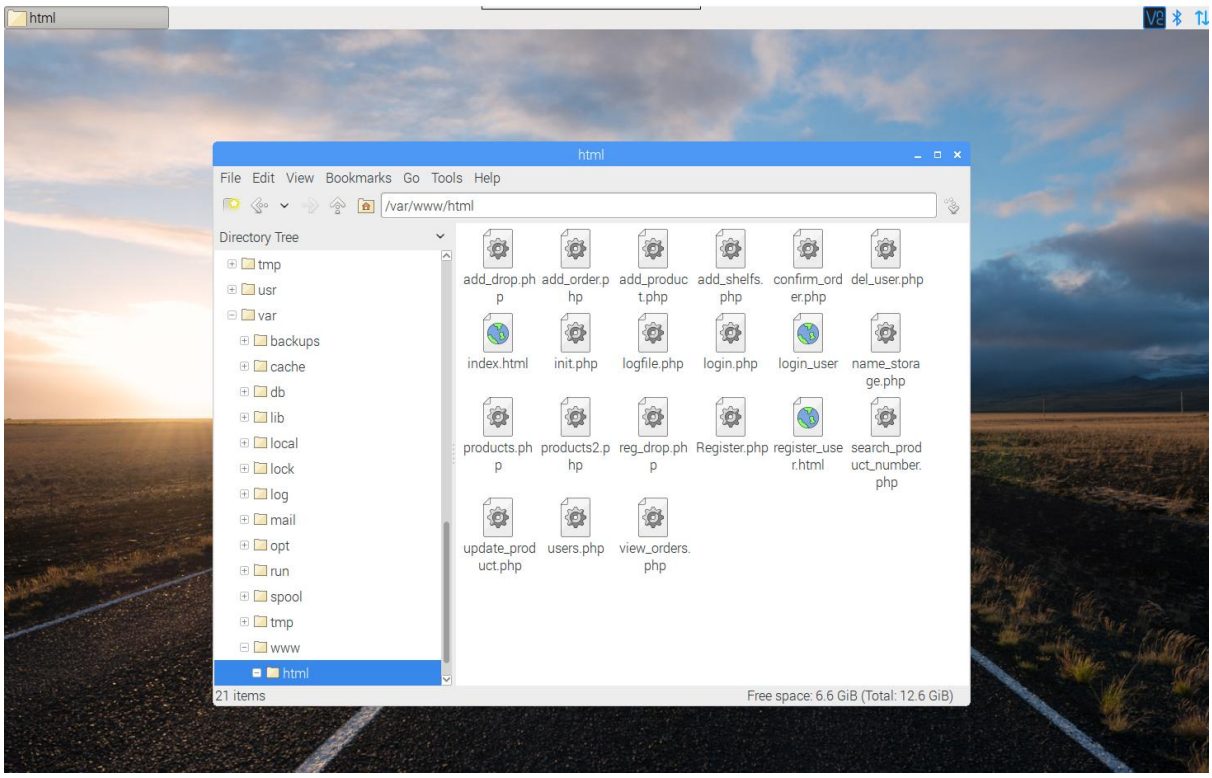
5.2 Περιγραφή του API (PHP)

Στον προγραμματισμό application programming interface (API) ή στα ελληνικά «διεπαφή προγραμματισμού εφαρμογών», λέμε το σύνολο ορισμών, πρωτοκόλλων και εργαλείων για την κατασκευή ή δημιουργία εφαρμογών λογισμικού. Γενικότερα είναι ένα σύνολο μεθόδων επικοινωνίας μεταξύ διάφορων λογισμικών. Η χρησιμότητα ενός API καθορίζετε από το κατά πόσο διευκολύνει την ανάπτυξη ενός προγράμματος, προσφέροντας όλα τα στοιχεία για την δομή που χρειάζεται ο προγραμματιστής. [23]

Ένα API μπορεί να είναι ένα σύστημα βασισμένο σε web εφαρμογή είτε σε λογισμικό σύστημα είτε ένα σύστημα βάσης δεδομένων ή λογισμικό βιβλιοθήκης ή ακόμα και hardware. Ένα API μπορεί να πάρει πολλές μορφές. Στην δική μας περίπτωση το API που δημιουργήσαμε είναι ένα Web Application API, αφού η δουλειά του είναι να διαχειρίζεται και να επικοινωνεί ως “μεσάζοντας” της βάσης δεδομένων και της εφαρμογής μας.

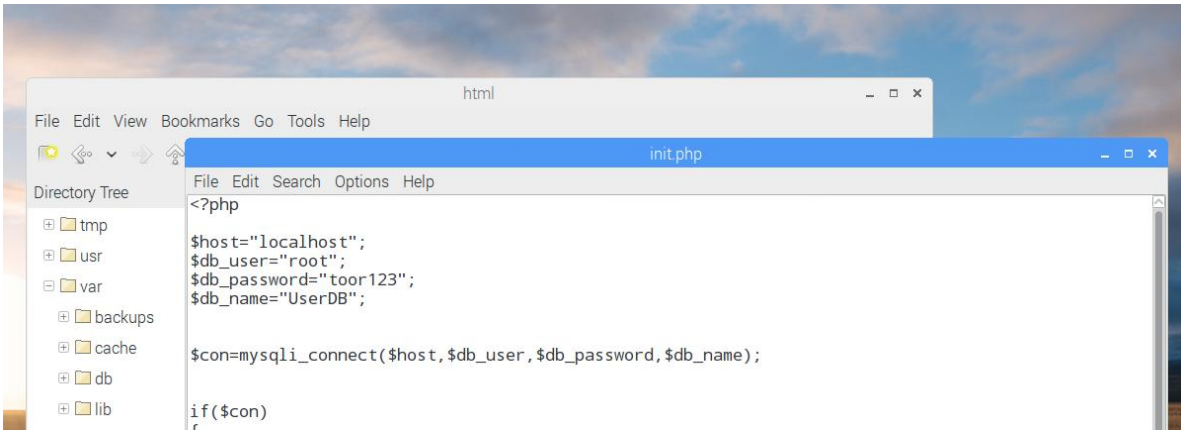
Τα Web API είναι διεπαφές με τις οποίες γίνονται διεργασίες, για παράδειγμα μιας επιχείρησης και των εφαρμογών που χρησιμοποίει αυτή. Ένα API (και στην περίπτωση μας RESTful API) τυπικά ορίζεται ως ένα σύνολο μηνυμάτων σε μορφή αίτησης πρωτοκόλλου μεταφοράς HTTP (Hypertext Transfer Protocol), μαζί με έναν ορισμό της δομής των μηνυμάτων απόκρισης, για παράδειγμα JavaScript Object Notation (JSON), την οποία μορφή χρησιμοποιούμε και εμείς στην εφαρμογή μας. [24]

Στον Server λοιπόν, έχω δημιουργήσει αρχεία PHP τα οποία όταν καλεστούν με αίτηση από την εφαρμογή, εκτελούν κάποια ενέργεια στην βάση δεδομένων. Αυτά τα αρχεία είναι αποθηκευμένα στο μονοπάτι "/var/www/html" του server.



Εικόνα 5.2-1

Πάμε λοιπόν να αναλύσουμε μερικά από τα αρχεία μας. Ένα βασικό αρχείο είναι αυτό επιτυγχάνει την επικοινωνία με την βάση δεδομένων, δηλαδή την σύνδεση με αυτή



(connection) '**init.php**':

Εικόνα 5.2-2

Αυτό το αρχείο συμπεριλαμβάνεται σε όλα τα υπόλοιπα php αρχεία μας για να είναι εφικτή η σύνδεση με την βάση δεδομένων.

login.php

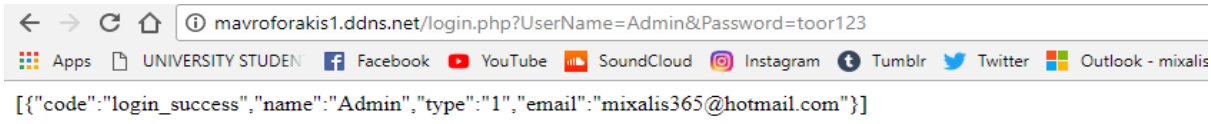
File Edit Search Options Help

```
<?php
session_start();

require "init.php";

date_default_timezone_set('Europe/Athens');
$action="Login";
$description="Success";
$user_name=$_REQUEST["UserName"];
$password=$_REQUEST["Password"];

$sql="select UserName,Email,Type from Users where UserName like '".$user_name."' and Password like '".$passwor
$response=array();
$result=mysqli_query($con,$sql);
if(mysqli_num_rows($result)>0)
{
 $row=mysqli_fetch_row($result);
 $name=$row[0];
 $email=$row[1];
 $type=$row[2];
 $code="login_success";
 array_push($response,array(
 "code"=>$code,
 "name"=>$name,
 "type"=>$type,
 "email"=>$email)
 );
//logfile
$sql10 = "INSERT INTO LogFile (LoginUser,Action,Description) VALUES ('$user_name','$action','$description');";
$result10=mysqli_query($con,$sql10);

print_r(json_encode($response));

}
else{
                $code="login failed";
                $msg="user not found";
                array_push($response,array(
                "code"=>$code,
                "msg"=>$msg)
                );

        print_r(json_encode($response));
}
mysqli_close($con);
?>
```

Εικόνα 5.2-3

Στην εικόνα 5.2-3 βλέπουμε το αρχείο με το οποίο πραγματοποιούμε το Login στην βάση δεδομένων που έχουμε κρατήσει τους χρήστες. Το αρχείο αυτό απαιτεί κατά το κάλεσμα του κάποια δεδομένα. Αυτά τα δεδομένα είναι Το Username και το Password, στην συνέχεια τα βάζουμε σε 2 μεταβλητές τις ‘$user_name’ και ‘$password’. Αυτό μας βοηθάει να πραγματοποιήσουμε το query που μας ανακτά ή όχι από την βάση μας τα στοιχεία και τα επιστρέφει σαν απάντηση σε μορφή JSON array. Για παράδειγμα αν εκτελέσουμε καρφωτά στο URL του browser μας «http://mavroforakis1.ddns.net/login.php?UserName=Admin&Password=toor123» τότε θα

πάρουμε το παρακάτω αποτέλεσμα της εικόνας 5.2-4 :



Εικόνα 5.2-4

Ομοίως για την εγγραφή του χρήστη έχουμε το αρχείο ‘**Register.php’,** όπου δέχεται σε μορφή JSON αίτημα (request), από την εφαρμογή μας τα στοιχεία του χρήστη που θέλουμε να δημιουργήσουμε, ελέγχει αν υπάρχει εγγραφή με ίδιο mail και εφόσον δεν υπάρχει, τα καταχωρεί σε μεταβλητές, τα στοιχεία των οποίων καταχωρεί στην βάση με την βοήθεια του query ‘**INSERT INTO table_name’** ή αλλιώς επιστρέφει ένα μήνυμα ότι υπάρχει εγγραφή**:**

Register.php

File Edit Search Options Help

```
$lname=$_REQUEST["LastName"];
$fname=$_REQUEST["FirstName"];
$user_name=$_REQUEST["UserName"];
$email=$_REQUEST["Email"];
$password=$_REQUEST["Password"];
$type=$_REQUEST["Type"];
$id_storage=$_REQUEST["id_Storage"];

$action="Register";
$description=$user_name;
$user_name2=$_REQUEST["UserName2"];


$sql="select * from Users where email like'".$email."';";


$result=mysqli_query($con,$sql);
$response=array();



if(mysqli_num_rows($result)>0)
{

$code="reg_failed";
$message="User already exist";
array_push($response,array("code"=>$code,"message"=>$message));
print_r(json_encode($response));

}
else
{

$sql="INSERT INTO Users(id_Storage,Type,LastName,FirstName,UserName,Email,Password) VALUES('".$id_storage."','
$result=mysqli_query($con,$sql);
$code="reg_success";
$message="User registered succesfull";
array_push($response,array("code"=>$code,"message"=>$message));

//logfile
$sql = "INSERT INTO LogFile (LoginUser,Action,Description) VALUES ('$user_name2','$action','$description');";
$result=mysqli_query($con,$sql);


print_r(json_encode($response));
```

Εικόνα 5.2-5

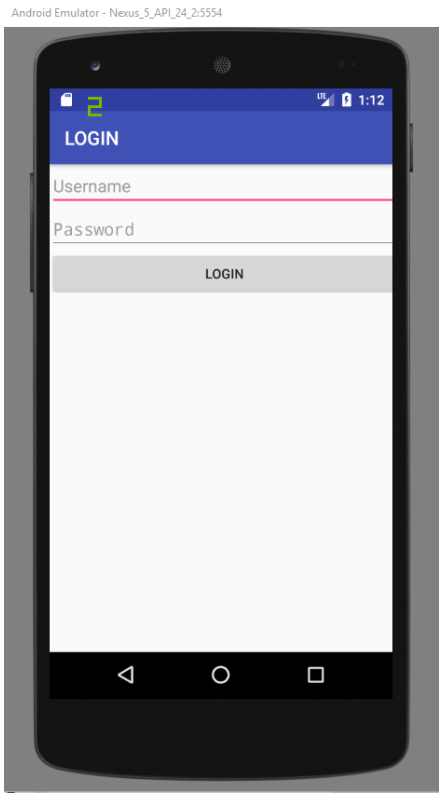
Επιπροσθέτως, κάνει και μία καταχώριση στον πίνακα ‘LogFile’ στον οποίο κρατάμε τις ενέργειες που γίνονται.

Με την βοήθεια λοιπόν των queries που περιγράψαμε σε προηγούμενο κεφάλαιο και με παρόμοιο τρόπο εκτελούμε εργασίες και αλληλοεπιδρούμε με την βάση δεδομένων στον

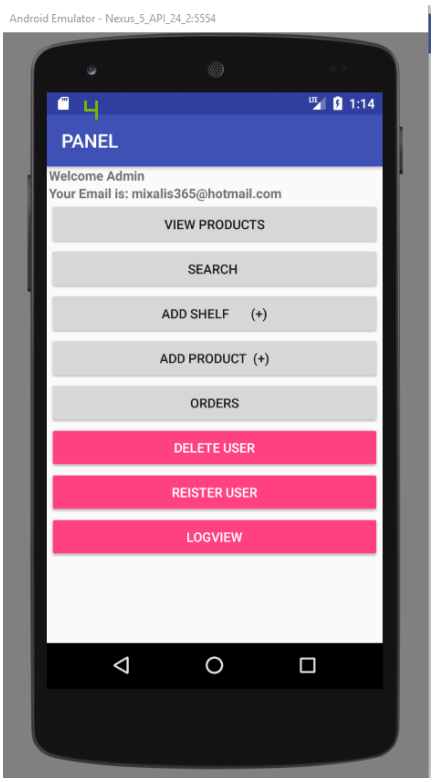
server. Αυτός ο τρόπος επικοινωνίας, πρέπει να αναφέρουμε ότι προσφέρει ένα είδος ασφάλειας των δεδομένων μας, κάτι πολύ σημαντικό για μία επιχείρηση, αφού όλες οι ενέργειες εκτελούνται από τον ίδιο τον server και είναι σε ελεγχόμενο περιβάλλον.

5.3 Περιγραφή Εφαρμογής

Κατά την εκκίνηση της εφαρμογής ο χρήστης καλείτε να κάνει Log in στο σύστημα, χωρίς το δικαίωμα εγγραφής, για τον απλούστατο λόγο ότι εγγραφή και διαγραφή αποθηκάριων μπορεί να κάνει μόνο ο Admin. Η εφαρμογή έχει το αντίστοιχο παρουσιαστικό:



Εικόνα 5.3-1

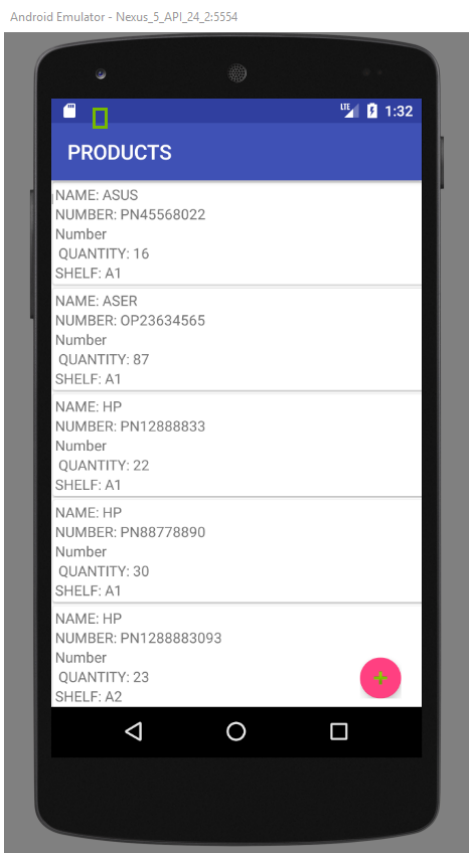


Εικόνα 5.3-2

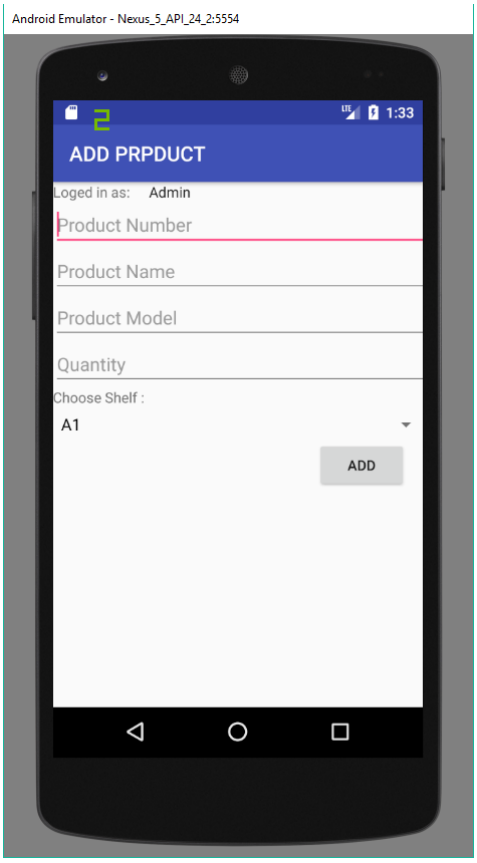
Πραγματοποιώντας σύνδεση παρουσιάζετε το αντίστοιχο Panel, αναγράφοντας στην κορυφή τον συνδεδεμένο χρήστη και το email του το οποίο έχουμε τραβήξει ως αποτέλεσμα του JSON response από το API μας, (Βλέπε εικόνα 5.2-4). Τα χρωματισμένα ροζ κουμπιά, είναι τα κουμπιά εκείνα τα οποία είναι διαθέσιμα αποκλειστικά για χρήση του Admin (π.χ. Manager, Υπεύθυνος αποθήκης). Αντίθετα, κατά την εισαγωγή ενός απλού χρήστη (π.χ. Αποθηκάριος) δεν είναι εμφανή.

Το πρώτο κουμπί της εφαρμογής μας ‘VIEW PRODUCTS’ μας εμφανίζει σε μορφή λίστας (List View) όλα τα προϊόντα που βρίσκονται στον πίνακα products της βάσης

δεδομένων στο server, καθώς και ένα κουμπί (FloatingActionButton), που μας μεταβιβάζει στο αντίστοιχο παράθυρο για εισαγωγή καινούριου προϊόντος στην βάση δεδομένων:



Εικόνα 5.3-1

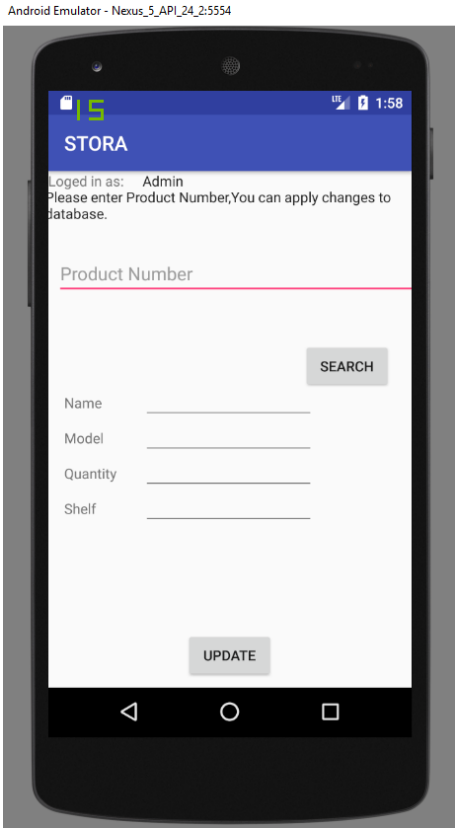


Εικόνα 5.3-2

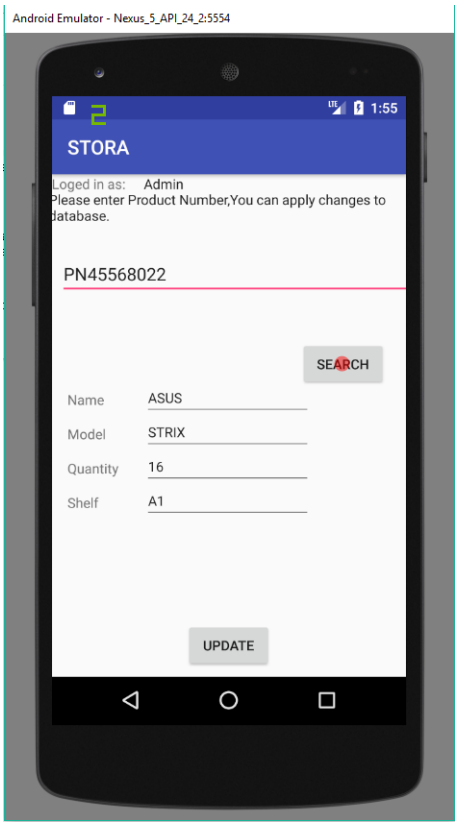
Το παρουσιαστικό προϊόντων (ListView) έχει την ιδιότητα να διαμορφώνει το μέγεθός του, ανάλογα με τις γραμμές του και είναι σε μορφή πάπυρου, δηλαδή έχει δυνατότητα ‘Scroll’.

Το panel εισαγωγής καινούριου προϊόντος, μας ζητάει να εισάγουμε αριθμό προϊόντος , όνομα προϊόντος, μοντέλο, ποσότητα και ράφι τοποθέτησης (φυσική θέση). Το ράφι τοποθέτησης είναι σε μορφή drop down (recycler viewer), όπου αποκτά τις τιμές του από την βάση δεδομένων διαθέσιμων ραφιών.

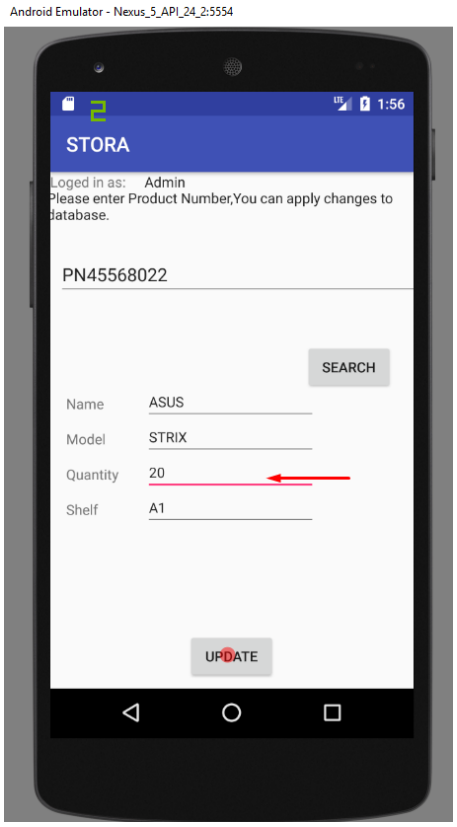
Το δεύτερο κουμπί ‘SEARCH’, μας μεταβιβάζει στο panel εύρεσης ενός προϊόντος, πληκτρολογώντας τον μοναδικό αριθμό προϊόντος και πατώντας το κουμπί ‘SEARCH’. Έπειτα, τα στοιχεία εμφανίζονται στα κουτιά. Επιπροσθέτως, υπάρχει η δυνατότητα αλλαγής των δεδομένων σε κάποιο προϊόν με την χρήση του κουμπιού ‘UPDATE’ και αποθηκεύει στην βάση τις αλλαγές .



Εικόνα 5.3-5

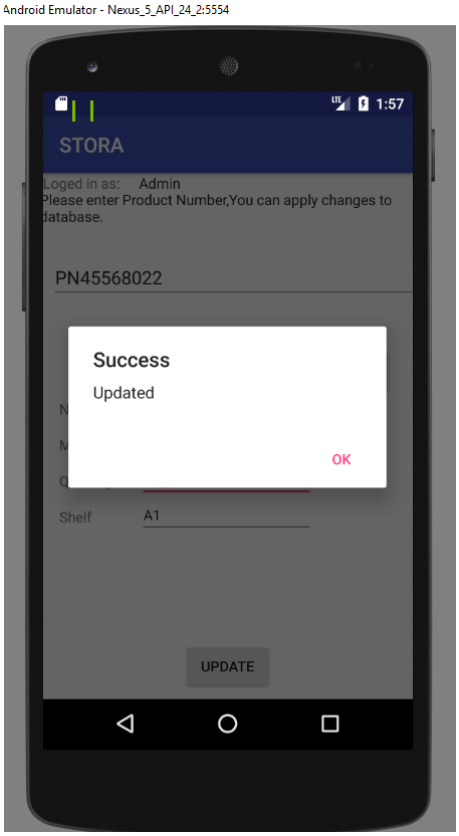


Εικόνα 5.3-6

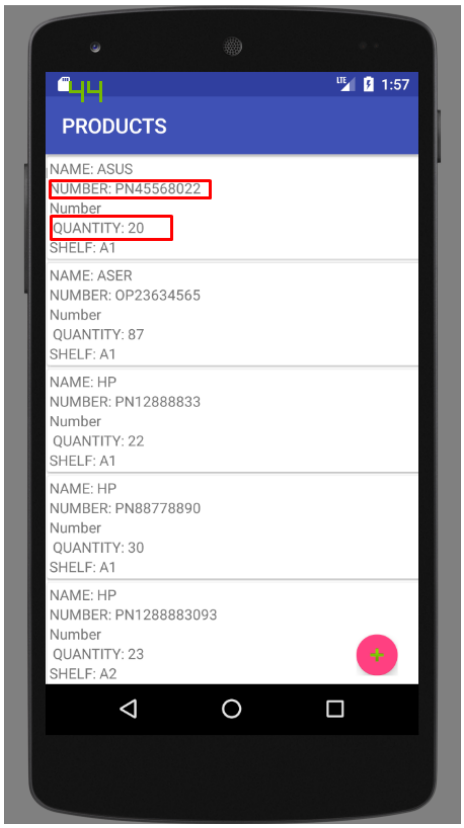


Εικόνα 5.3-7

Στην εικόνα 5.3-8 παρατηρούμε ότι έχουμε ένδειξη επιτυχημένης αλλαγής και ένδειξη αυτής παρουσιάζετε με την ανάδυση ενός ‘Alert Dialog’ όπου μας ενημερώνει ανάλογα. Στην εικόνα 5.3-9 βλέπουμε το αποτέλεσμα της ενημέρωσης προϊόντος που πραγματοποιήθηκε, πηγαίνοντας στο panel ‘PRODUCTS’.

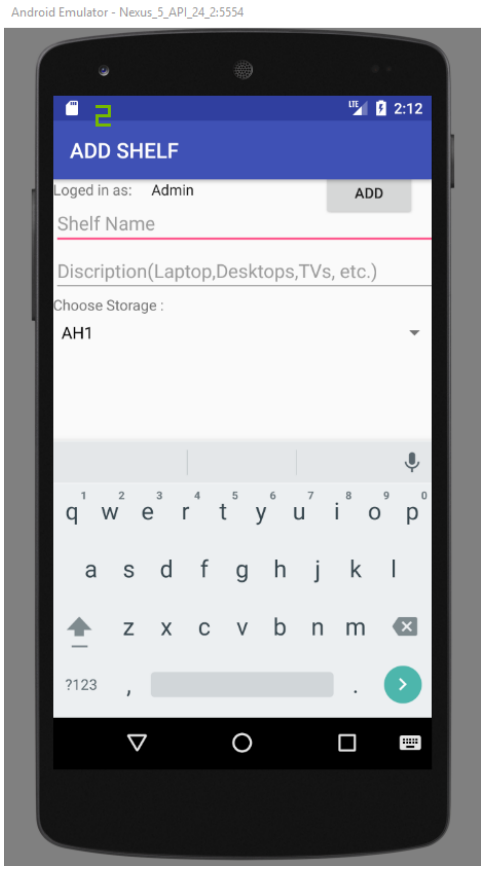


Εικόνα 5.3-4



Εικόνα 5.3-3

Το τρίτο κουμπί ‘ADD SHELF’ επιτρέπει την εισαγωγή καινούριου ραφιού στην βάση δεδομένων μας, κάνοντας εισαγωγή ενός ονόματος και μίας περιγραφής (Βλέπε hint εφαρμογής). Στην συνέχεια επιλέγουμε σε ποια αποθήκη θα γίνει η εισαγωγή (π.χ. Ηράκλειο, Αθήνα).

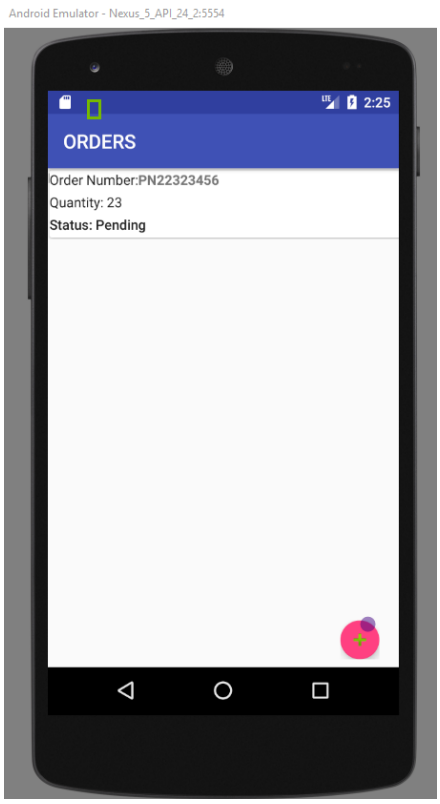


Εικόνα 5.3-5

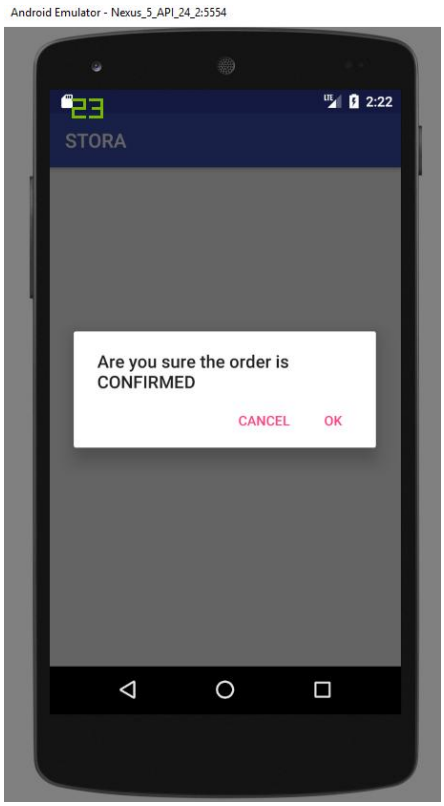
Το τέταρτο κουμπί ‘ADD PRODUCT’ μας μεταβιβάζει στο panel της εικόνας 5.3-4 ώστε να γίνει εισαγωγή προϊόντος στην βάση.

Το πέμπτο κουμπί είναι αυτό που μας δείχνει μία λίστα (List View) με όλες τις μη ολοκληρωμένες παραγγελίες που έχουν γίνει. Ακόμη, βλέπουμε ένα κουμπί (Floating Action Button), με το οποίο πραγματοποιούμε μια νέα παραγγελία. Τα στοιχεία της λίστας είναι επιλέξιμα και κατά την επιλογή τους, θέτετε το ερώτημα επί ολοκλήρωσης της παραγγελίας,

σε ένα Alert Dialog για το κατά πόσο είμαστε σίγουροι για την ολοκλήρωση της παραγγελίας:



Εικόνα 5.3-11

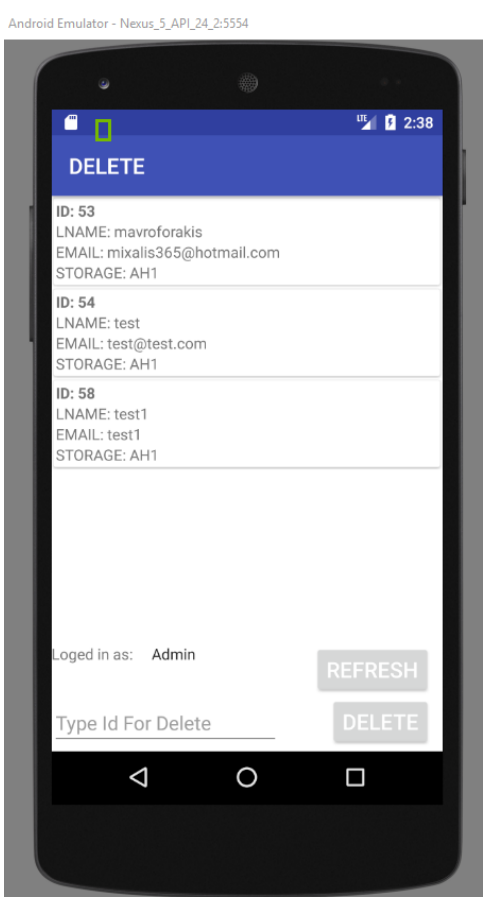


Εικόνα 5.3-12

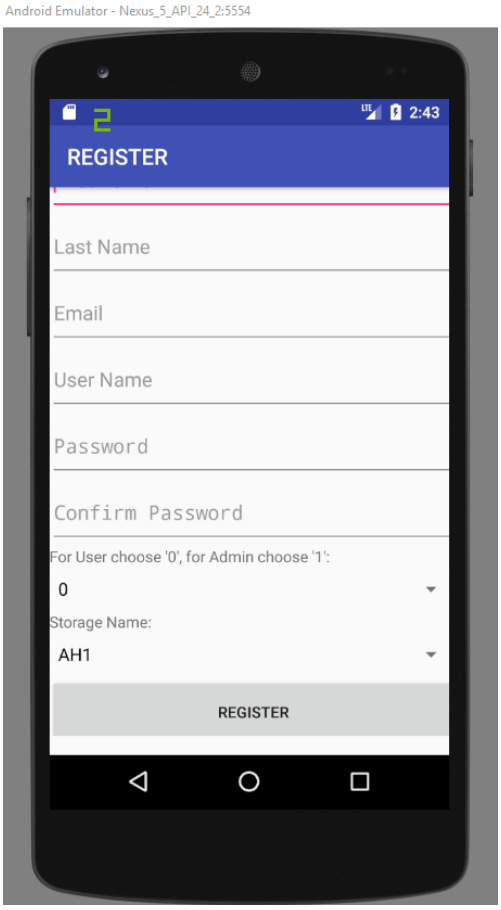
Το έκτο κουμπί του Admin, είναι εκείνο για την διαγραφή χρήστη (π.χ. Απόλυση αποθηκάριου) ‘DELETE USER’, που μας παρουσιάζει όλους τους εγγεγραμμένους χρήστες, με μία έμφαση στο ID τους, καθώς και τα βασικά στοιχεία τους. Η διαγραφή ενός χρήστη επιτυγχάνετε γράφοντας στο κουτάκι (που βρίσκετε στο τέλος του panel) το ID του χρήστη και πατώντας το κουμπί ‘DELETE’. Ακόμα υπάρχει ένα κουμπί ‘REFRESH’ το οποίο κάνει ανανέωση της λίστας.

Το επόμενο κουμπί για αποκλειστική χρήση του Admin, είναι εκείνο της εγγραφής νέου χρήστη, ‘REGISTER USER’ που παρουσιάζει κουτάκια για εισαγωγή στοιχείων καθώς και δυο drop down (recycler view) για την επιλογή του είδους χρήστη (π.χ. Διαχειριστή,

Αποθηκάριου) και σε ποια αποθήκη εργάζεται. Αυτό το στοιχείο για ακόμη μια φορά ανακτάτε από την βάση δεδομένων:

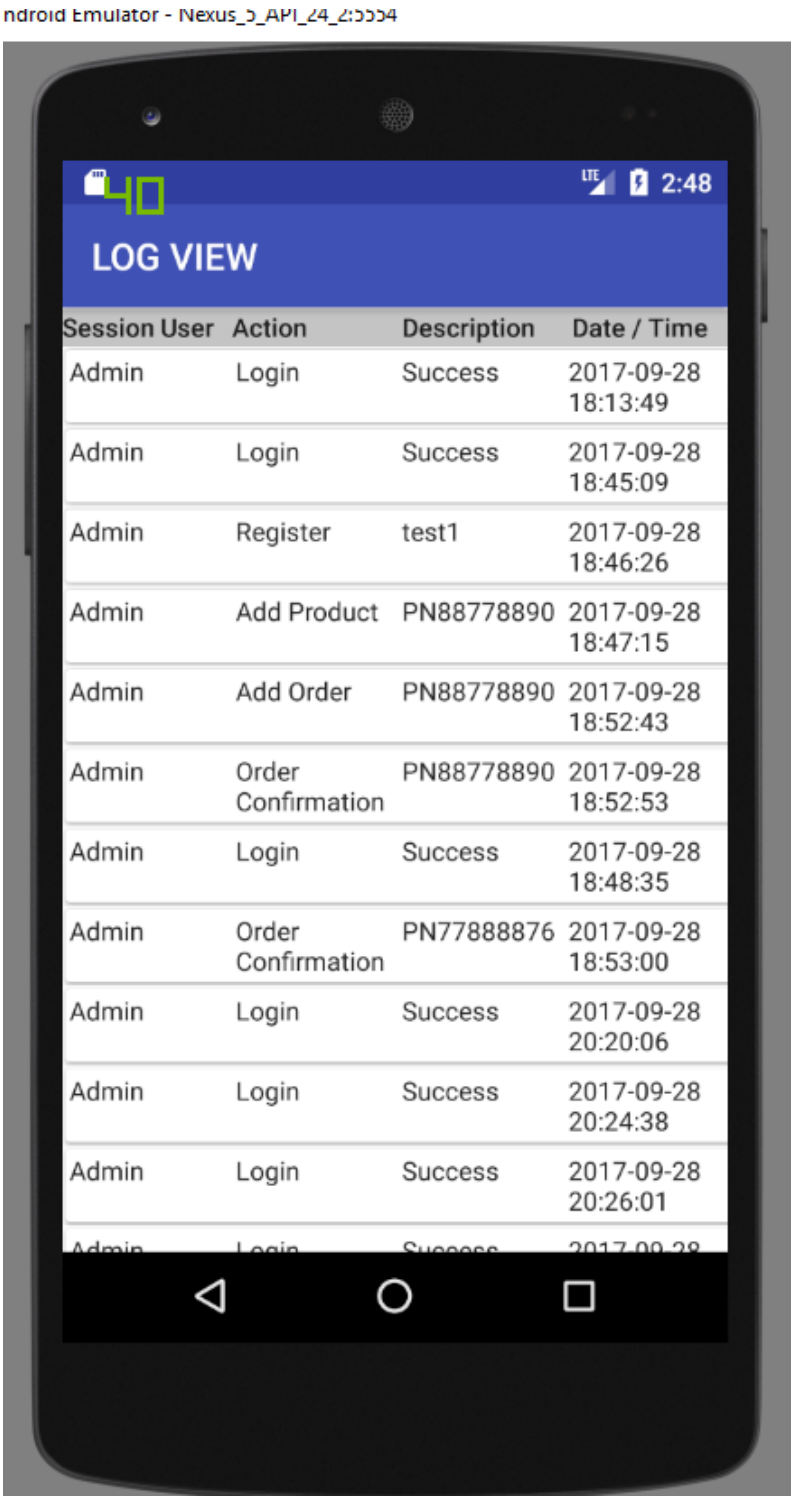


Εικόνα 5.3-13



Εικόνα 5.3-14

Τέλος, έχουμε το κουμπί ‘LOG VIEW’, που έχει την λειτουργία εμφάνισης των κινήσεων και ενεργειών σε όλες τις διαδικασίες που εκτελούνται και για όλους τους χρήστες, σε μορφή πίνακα. Το πεδίο ‘Session User’, δείχνει ποιον αφορά η συγκεκριμένη λειτουργία, το ‘Action’ πεδίο αναδεικνύει την ενεργεία που εκτελέστηκε και το ‘Description’ περιγράφει την δράση. Τέλος, το πεδίο ‘Date/Time’ δείχνει την ημερομηνία και την ώρα εκτέλεσης της, κάτι που παράγετε από τον ίδιο τον server κατά την εισαγωγή.


ndroid Emulator - Nexus_5_API_24_2:5554
2:48
LOG VIEW
Session User
Action
Description
Date / Time
Admin
Login
Success
2017-09-28 18:13:49
Admin
Login
Success
2017-09-28 18:45:09
Admin
Register
test1
2017-09-28 18:46:26
Admin
Add Product
PN88778890
2017-09-28 18:47:15
Admin
Add Order
PN88778890
2017-09-28 18:52:43
Admin
Order Confirmation
PN88778890
2017-09-28 18:52:53
Admin
Login
Success
2017-09-28 18:48:35
Admin
Order Confirmation
PN77888876
2017-09-28 18:53:00
Admin
Login
Success
2017-09-28 20:20:06
Admin
Login
Success
2017-09-28 20:24:38
Admin
Login
Success
2017-09-28 20:26:01

Εικόνα 5.3-6

5.4 Περιγραφή σημαντικών μερών του κώδικα και του Android Studio

Αρχικά, πρέπει να πούμε ότι η εφαρμογή μας δεν επικοινωνεί άμεσα με την βάση δεδομένων. Ο τρόπος επικοινωνίας όπως έχουμε προαναφέρει, γίνετε μέσω του API που κατασκευάσαμε στον server. Η επικοινωνία λοιπόν με το API γίνετε με την χρήση JSON request. Αυτή την διαδικασία, την χρησιμοποιούμε συχνά στην εφαρμογή, αφού κάθε μας ενέργεια πρέπει να γίνετε σε μορφή αίτησης. Παρακάτω, αναδεικνύεται η βασική μορφή του κώδικα που πραγματοποιεί την αίτηση- απάντηση.

```
StringRequest stringRequest = new StringRequest(Request.Method.POST, add_url,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                JSONObject j = null;

                try {

                    JSONArray jsonArray = new JSONArray(response);
                    JSONObject jsonObject = jsonArray.getJSONObject(0);
                    String code = jsonObject.getString("code");

                    if(code.equals("fail"))
                    {
                        builder.setTitle("Error");
                        displayAlert(jsonObject.getString("message"));
                    }
                    else
                    {
                        builder.setTitle("Success");
                        displayAlert(jsonObject.getString("message"));
                    }


                } catch (JSONException e) {
                    e.printStackTrace();
                }

            }
        }, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        Toast.makeText(AddShelf.this,"Error",Toast.LENGTH_LONG).show();
        error.printStackTrace();

    }
```

Εικόνα 5.4-1

```
            }
        }) {
            @Override
            protected Map<String, String> getParams() throws AuthFailureError {
                Map<String, String> params = new HashMap<~>();
                params.put("sname", shname);
                params.put("sdisc", sdisc);
                params.put("storname", sname2);
                params.put("UserName", java.TAG_NAME);
                return params;
            }
        };
        MySingleton.getmInstance(AddShelf.this).addToRequestque(stringRequest);


    }


});
```

Εικόνα 5.4-2

Στην παραπάνω εικόνα 5.4-2 βλέπουμε την αποστολή των στοιχείων μας στο API (file.php). Στη συνέχεια σύμφωνα με την απάντηση που λάβαμε από το API (server), (βλ.5.4-1) αν δηλαδή εκτελέστηκε η ενέργεια μας ή όχι, διαβάζουμε την μεταβλητή ‘code’ που περιέχει, από το αρχείο php, ένα string. (π.χ. Fail, Success).

Στο Map που φτιάχνουμε, το πρώτο μέρος αφορά το όνομα της μεταβλητής όπως το έχουμε ορίσει στο php, ενώ το δεύτερο μέρος αφορά την μεταβλητή που στέλνουμε για να γίνει το query στο php.

Όσο αφορά την μέθοδο του Request συνήθως γίνετε με POST/GET και ακολουθεί η μεταβλητή που έχει διεύθυνση στην οποία κάνουμε το Request, (π.χ.`String add_url = "http://mavroforakis1.ddns.net/add_shelfs.php";`). (βλ.5.4-1)

Τα αποτελέσματα από ένα Request συνήθως γυρνάνε σε μορφή JSON ARRAY. Αυτό το array συνήθως, το ορίζουμε με ένα όνομα. Στο παρακάτω παράδειγμα το όνομα της λίστας μας είναι ‘result’ και περιέχει τα δεδομένα των μεταβλητών μας . (βλ. 5.2-4)

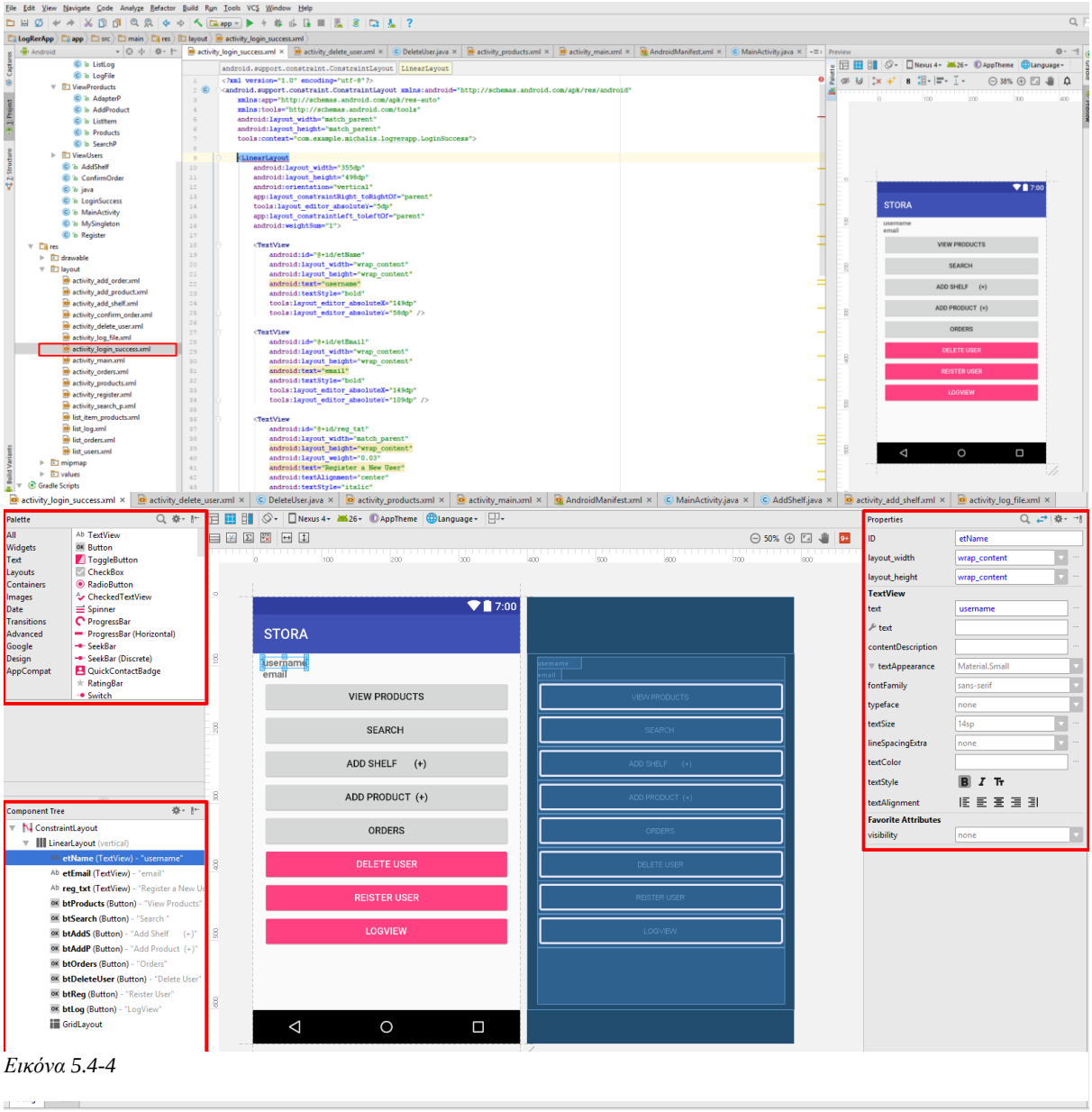
```
        StringRequest stringRequest =new StringRequest(Request.Method.GET,
                URL_PRODUCTS,
                (response) → {
                        progressDialog.dismiss();
                        try {
                            JSONObject jsonObject=new JSONObject(response);
                            JSONArray array=jsonObject.getJSONArray("result")

                            for(int i=0;i<array.length();i++){
                                JSONObject jo=array.getJSONObject(i);
                                ListItem item = new ListItem(
                                        jo.getString("pname"),
                                        jo.getString("pmodel"),
                                        jo.getString("pnumber"),
                                        jo.getString("pqnt"),
                                        jo.getString("sname")
                                );
                                listItems.add(item);
                            }
                            adapter = new AdapterP(listItems,getApplicationContext());
                            recyclerView.setAdapter(adapter);
                        } catch (JSONException e) {
                            e.printStackTrace();
                        }

                }, (error) → {
                progressDialog.dismiss();
                Toast.makeText(Products.this,"Error",Toast.LENGTH_LONG).show();

        });

        RequestQueue requestQueue = Volley.newRequestQueue(this);
        requestQueue.add(stringRequest);
    }
}
```

Εικόνα 5.4-3

Το Android Studio έχει πολλές λειτουργίες· μία από αυτές είναι το σχεδιαστικό κομμάτι. Στην παρακάτω εικόνα, βλέπουμε στην αριστερή μεριά την παλέτα σχεδιασμού μας και το δέντρο με τα συστατικά στοιχεία μας (π.χ. κουμπιά κ.α.).

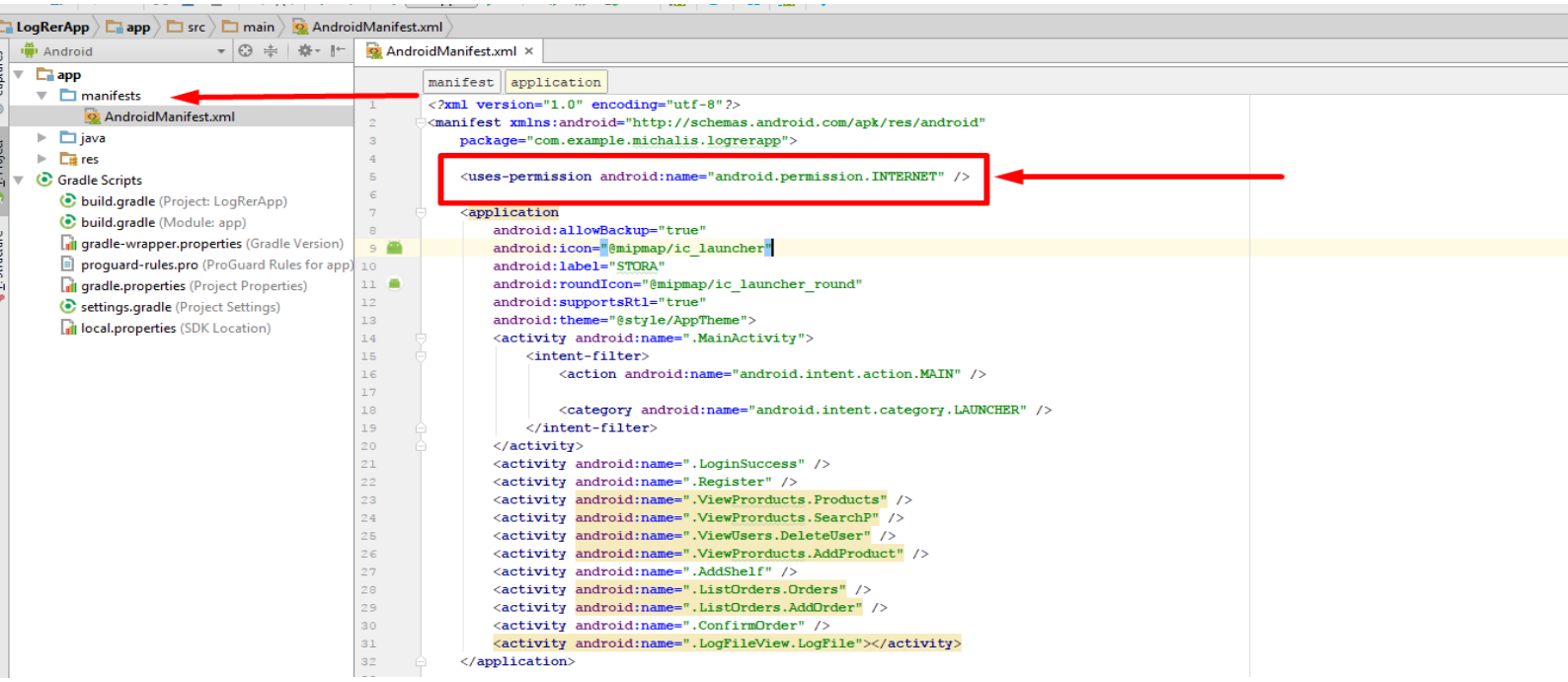
Στην δεξιά μεριά, βλέπουμε το αποτέλεσμα της επιλογής κάποιου συστατικού στην σχεδίαση μας, όπου εκεί μπορούμε να δώσουμε ονόματα στα αντικείμενα μας και διάφορα δεδομένα. Στο κέντρο, βλέπουμε το αποτέλεσμα του compile για μια επιτυχή εκτέλεση όταν τρέξει η εφαρμογή (Εικόνα 5.4-4).

Όλα τα στοιχεία του σχεδιαστικού κομματιού είναι αποτέλεσμα XML κώδικα.



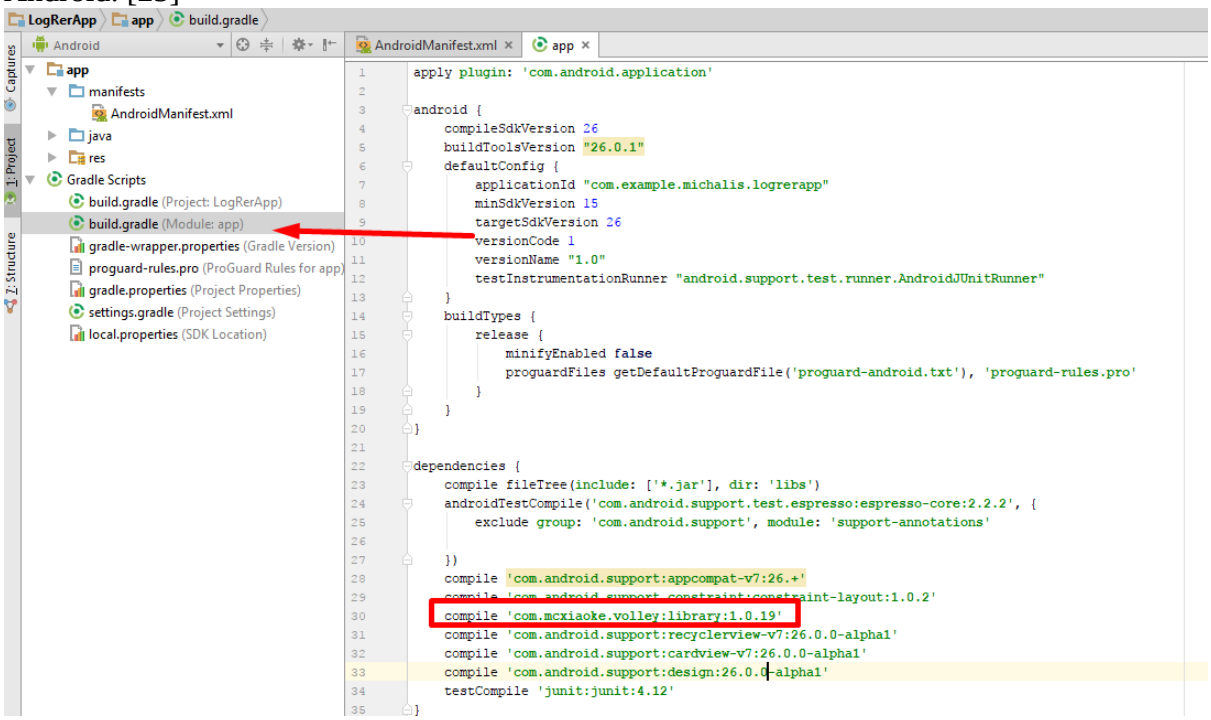
Εικόνα 5.4-4

Για να μπορούμε να έχουμε μια επιτυχή σύνδεση στο διαδίκτυο πρέπει να επιτρέψουμε στο project μας την λειτουργία αυτή. Κάτι τέτοιο επιτυγχάνετε με την εισαγωγή βιβλιοθήκης στο AndroidManifest.xml :



Εικόνα 5.4-5

Όλα μας τα request γίνονται με την βοήθεια μιας βιβλιοθήκης, την volley. Η Volley είναι μια βιβλιοθήκη HTTP η οποία διευκολύνει την διαδικτυακή λειτουργία για εφαρμογές Android. [25]



Εικόνα 5.4-6

Βιβλιογραφία

Αναφορές

1. Εγκυκλοπαίδεια – Wikipedia, Smartphone
   https://el.wikipedia.org/wiki/Smartphone October 2017
2. Article «Μερίδιο Αγοράς»
   http://dimkar.wixsite.com/smartphones/blank-mg6te October 2017
3. «Android Login and Registration with PHP», MySQL and SQLite
   Ravi Tamada January 2012
4. Article «What is PHP?», PHP
   http://php.net/manual/en/intro-whatis.php October 2017
5. Εγκυκλοπαίδεια – Wikipedia, Τι είναι PHP?
   https://el.wikipedia.org/wiki/PHP October 2017
6. Article «What is MySQL? » SiteGround
   https://www.siteground.com/tutorials/php-mysql/mysql.htm October 2017
7. Εγκυκλοπαίδεια – Wikipedia, SQL
   https://el.wikipedia.org/wiki/SQL October 2017
8. Εγκυκλοπαίδεια – Wikipedia, Android
   https://el.wikipedia.org/wiki/%CE%99%CF%83%CF%84%CE%BF%CF%81%CE%AF%CE%B1_%CE%B5%CE%BA%CE%B4%CF%8C%CF%83%CE%B5%CF%89%CE%BD_%CF%84%CE%BF%CF%85_Android October 2017
9. «The history of Android Studio», Ron Amadeo October 2016
10. Εγκυκλοπαίδεια – Wikipedia, Android Studio
    https://en.wikipedia.org/wiki/Android_Studio October 2017
    «Android Studio Release Notes», Android Developers Official Website June 2017
11. Εγκυκλοπαίδεια – Wikipedia, Raspberry Pi
    https://en.wikipedia.org/wiki/Raspberry_Pi October 2017
12. «Raspberry Pi 3 Model B», SPECIFICATIONS, Raspberry Pi
    https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ October 2017
13. «NOOBS», Raspberry Pi
    https://www.raspberrypi.org/downloads/noobs/ October 2017
14. «RASPBIAN», Raspberry Pi
    https://www.raspberrypi.org/downloads/raspbian/ October 2017
    «Introducing Pixel», Raspberry Pi October 2016
15. Εγκυκλοπαίδεια – Wikipedia, No-IP, TECHNOLOGY & SERVICES
    https://en.wikipedia.org/wiki/No-IP October 2017
16. Article «Dynamic DNS Update», Alejandro Munoz
    http://www.teldat.com/blog/dynamic-dns-update/ June 2017
17. «Java Προγραμματισμός» ($10^{η}$ έκδοση), Deitel Paul & Deiter Harvey (2015)
    ελληνική μετάφραση, Αθήνα (Μ. Γκιούρδας)
18. Article «How to make a Raspberry Pi Web Server», Alok Naushad

19. Official No – IP site
https://www.noip.com/login?ref_url=console#!/dynamic-dns October 2017
20. «The Debian System», Martin F. Krafft (2005)
21. Article «What is Referential Intedrity», Mike Chapple
https://www.thoughtco.com/referential-integrity-definition-1019181 March 2017
22. MySQL Database, w3schools.com
https://www.w3schools.com/php/php_mysql_insert_lastid.asp October 2017
23. Εγκυκλοπαίδεια – Wikipedia, Applications programming interface
https://en.wikipedia.org/wiki/Application_programming_interface#cite_note-19
October 2017
24. «Services Mashups: The New Generation of Web Applications», Djamal Benslimane, Dustdar Schahram, Sheth Amit (2008)
IEEE Internet Computing vol12, no5
Institute of Electrical and Electronics Engineers pp 13-15
25. Transmitting Network Data Using Volley, Developers
https://developer.android.com/training/volley/index.html October 2017
26. Εγκυκλοπαίδεια – Wikipedia, Representational state transfer (REST)
https://en.wikipedia.org/wiki/Representational_state_transfer#cite_note-13
October 2017

Αναφορές για την εφαρμογή

1. YouTube, Android Recycler View Tutorial
https://www.youtube.com/watch?v=USbTcGx1mD0&t=8s
2. YouTube, Android SQLite MySQL Sync
https://www.youtube.com/watch?v=jDMYcjdeXno&index=1&t=15s&list=WL
3. YouTube, Android To MySql Database
https://www.youtube.com/watch?v=cDGc28XQnRo&index=3&t=1s&list=WL
4. YouTube, Android Studio Tutorial - Login and Register
https://www.youtube.com/watch?v=x0I5vJfaRIU&t=658s
5. Android Application Development Tutorial, Course, Kilobolt
http://www.kilobolt.com/android-application-development-tutorial.html
6. Official – stackoverflow
https://stackoverflow.com/