ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Υλοποίηση Android εφαρμογής E-shop με χρήση web services

Του φοιτητή
Καρδαμανίδη Χρήστου
Αρ. Μητρώου: it144263

Επιβλέπων
Ονοματεπώνυμο Ευκλείδης Κεραμόπουλος
Αναπληρωτής Καθηγητής

Ημερομηνία 18-9-2020

Υλοποίηση Android εφαρμογής E-shop με χρήση web services
Κωδικός Δ.Ε. 19056
Χρήστος Καρδαμανίδης
Ευκλείδης Κεραμόπουλος
Ημερομηνία ανάληψης Δ.Ε. 27-11-2019
Ημερομηνία περάτωσης Δ.Ε. 18-9-2020

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

«Για τους αγαπημένους μου ανθρώπους που στήριξαν την προσπάθεια μου»

Πρόλογος

Ο λόγος για τον οποίο επιλέχθηκε αυτή η πτυχιακή είναι οι διάφορες γνώσεις οι οποίες προσφέρθηκαν σε όλα τα επίπεδα της. Για αρχή είναι πολύ σημαντική η διαδικασία ανάπτυξης ενός ολοκληρωμένου συστήματος. Η ανάπτυξη του Project αυτού απαιτούσε γνώσεις από διάφορους τομείς της πληροφορικής. Για αρχή, η διαδικασία της ανάλυσης απαιτήσεων και ο σχεδιασμός της διεπαφής και στην πορεία η μοντελοποίηση της βάσης των δεδομένων και ο σχεδιασμός της γενικότερης αρχιτεκτονικής της εφαρμογής. Επίσης για την σωστή σχεδίαση της αρχιτεκτονικής χρειάστηκαν γνώσεις όπως της αρχιτεκτονικής του Android, των δικτύων, των Web Services, αλλά και της ασφάλειας. Ο βασικός λόγος για την ανάπτυξη ήταν η εξειδίκευση πάνω στην ανάπτυξη και τον προγραμματισμό των εφαρμογών Android, αλλά παράλληλα με αυτόν τον σκοπό ήταν γνωστό πως θα υπάρχει ανάγκη ανάπτυξη και άλλων δεξιοτήτων στους τομείς που αναφέρθηκαν οι οποίες είναι κρίσιμες για έναν προγραμματιστή.

Περίληψη

Το θέμα της πτυχιακής είναι “Υλοποίηση Android εφαρμογής e-shop με χρήση Web Services”. Στα πλαίσια της πτυχιακής εργασίας αναπτύχθηκε μια εφαρμογή κινητού για την ολοκληρωμένη διαχείριση ενός ηλεκτρονικού καταστήματος. Η ανάπτυξη αυτής της εφαρμογής περιλαμβάνει μια Android εφαρμογή γραμμένη σε JAVA η οποία θα μπορεί να εξυπηρετεί managers του καταστήματος, αλλά και τους διαχειριστές αυτού οι οποίοι θα έχουν ως ρόλο την τεχνική υποστήριξη αυτού. Η εφαρμογή αυτή θα μπορεί να βρίσκεται σε διάφορα κινητά είτε managers είτε administrators. Για την πλήρη λειτουργικότητα της εφαρμογής αναπτύχθηκαν διάφορα layouts για αυτούς τους ρόλους και τις λειτουργικές τους απαιτήσεις. Οι λειτουργικές απαιτήσεις που υλοποιήθηκαν είναι η σύνδεση δύο διαφορετικών ειδών χρηστών. Για τους administrator χρήστες υπάρχει η δυνατότητα προβολής των διάφορων καταστημάτων, προβολή των διάφορων υπαλλήλων ανά κατάστημα, αλλά και προσθήκη νέων καταστημάτων και υπαλλήλων. Επίσης είναι δυνατή η εποπτεία των διάφορων πληροφοριών των Web Services. Για τους manager χρήστες υπάρχει η δυνατότητα προβολής των στατιστικών πωλήσεων ανά ημέρα σε ραβδόγραμμα. Επίσης είναι η δυνατή η διαχείριση των προϊόντων του μαγαζιού και των κατηγοριών προϊόντων του μαγαζιού. Επιπλέον βασικό σημείο είναι η προβολή και διαχείριση των διάφορων παραγγελιών που υπάρχουν στο ηλεκτρονικό κατάστημα. Για την υποστήριξη όλων αυτών των αντιγράφων της εφαρμογής σε διάφορα κινητά υπάρχει η ανάγκη για την μαζική αποθήκευση όλων των δεδομένων σε μια βάση δεδομένων. Αυτή η βάση δεδομένων περιλαμβάνει τούς πίνακες όπου αποθηκεύονται τα δεδομένα. Για την πρόσβαση στην βάση δεδομένων υπάρχει ανάγκη για ένα ενδιάμεσο στρώμα που να ρυθμίζει την πρόσβαση σε αυτήν και να δίνει ένα WEB API το οποίο θα περιλαμβάνει την επικοινωνία με την βάση δεδομένων. Αυτό είναι το Web Service το οποίο έχει υλοποιηθεί σε PHP κώδικα και χειρίζεται την αποστολή και λήψη δεδομένων σε JSON μορφή.

«Development of an e-shop Android Application using Web Services»

Kardamanidis Christos

Abstract

The subject of the thesis was the development of an Android application using Web Services. This Android Application will be a tool for e-shop owners and their employees in order to accomplish tasks of e-shop management. As a result of that, many related requirements were included during the designing of the Android Application. At first, those requirements were important in order to design the database, the web services and the user Interface of the Application. One simple example of those requirements is the product management that Managers are able to do in the Application. Also, the application helps the manager in order to review the sales profit of the e-shop and to approve new orders from customers. In addition to that it is important that there are tools for administrators of eshop in order to add more physical shops and to add new employees to the shop.

Ευχαριστίες

Η παρούσα πτυχιακή εργασία με θέμα «Υλοποίηση Android εφαρμογής E-shop με χρήση Web Services» πραγματοποιήθηκε για το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς πανεπιστήμιου Ελλάδος. Θα ήθελα να δώσω ιδιαίτερες ευχαριστίες μου στον καθηγητή κύριο Ευκλείδη Κεραμόπουλο που μου έδωσε την δυνατότητα να πραγματοποιήσω την πτυχιακή μου εργασίας υπό την δική του ευθύνη. Οι σημαντικές υποδείξεις και οι συμβουλές του μου ήταν ιδιαίτερα χρήσιμες τόσο και για την ολοκλήρωση της πτυχιακής μου, αλλά μου έδωσε και κατευθύνσεις και σημαντικά εφόδια για την μετέπειτα ζωή μου. Θα ήθελα επίσης να ευχαριστήσω επίσης όλους τους καθηγητές του τμήματος για τις πολύτιμες γνώσεις που μου προσέφεραν όλα αυτά τα χρόνια. Το μεγαλύτερο ευχαριστώ στα αγαπημένα μου πρόσωπα, στους γονείς μου για την καθοδήγηση και την στήριξη τους για όλα αυτά τα χρόνια των σπουδών μου, χωρίς την οποία τίποτα από όσα κατάφερα όλα αυτά τα χρόνια δεν ήταν πραγματικότητα.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
API	Application Programming Interface
DBMS	Database Management System
PHP	HyperText Preprocessor
JDK	Java Development Kit
HTTP	Hypertext Transfer Protocol
TCP	Transmission Control Protocol
JSON	Javascript Object Notation
XML	Extensive Markup Language
SSL	Secured Socket Layer
TLS	Transport Secured Layer
RFC	Request for Comments
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
IP	Internet Protocol
POJO	Plain Old Java Object

Κεφάλαιο 1ο: Εισαγωγή

Το θέμα της πτυχιακής είναι η «Υλοποίηση Android εφαρμογής E-shop με χρήση web services». Για την πλήρη υλοποίηση της πτυχιακής χρησιμοποιήθηκαν διάφορες τεχνολογίες σε διάφορα επίπεδα της πτυχιακής. Η επιλογή όλων αυτών των τεχνολογιών έγινε με κριτήριο την ανάλυση απαιτήσεων που έγινε για την λειτουργικότητα της Android εφαρμογής.

Για αρχή εφόσον μιλάμε για μια εφαρμογή όπου υπάρχει αποθήκευση δεδομένων και χρήση από πολλούς διαφορετικούς χρήστες, σίγουρα θα χρειαζόταν ένα μέρος στο οποίο αποθηκεύονται μαζικά τα δεδομένα. Για την επίλυση αυτού του προβλήματος, υπήρχε η ανάγκη να υπάρχει ένα τέτοιο μέρος το οποίο είναι προσβάσιμο από τον κάθε χρήστη και το κάθε αντίγραφο της εφαρμογής. Για αυτό τον λόγο για αρχή διαλέχτηκε μια βάση δεδομένων η οποία βρίσκεται σε ένα Database Management System. Μέσα σε αυτόν τον Server υπήρχε η σχεδίαση της βάσης, σύμφωνα με την ανάλυση των λειτουργικών απαιτήσεων. Αυτοί οι πίνακες μοντελοποιήθηκαν με σκοπό την εύκολη αποθήκευση των δεδομένων από τον προγραμματιστή, αλλά και για την εύκολη συντήρηση του. Αυτή η βάση περιλαμβάνει τους διάφορους πίνακες για τους χρήστες, τις παραγγελίες, τα προϊόντα, τις κατηγορίες αλλά και διάφορα άλλα ζωτικά σημεία μιας εφαρμογής ηλεκτρονικού καταστήματος. Η βάση δεδομένων η οποία επιλέχθηκε είναι η MariaDB. Επίσης, προκειμένου να υπάρχει και λιγότερος κώδικας αλλά και απόκρυψη κάποιων πεδίων των βάσεων δεδομένων υπήρξε η ανάγκη για την δημιουργία stored procedures.

Πριν αναλυθεί το κομμάτι της Android εφαρμογής υπάρχει η ανάγκη για ένα ενδιάμεσο στάδιο το οποίο να διαχειρίζεται την βάση και να μην υπάρχει απευθείας σύνδεση των Android εφαρμογών στην βάση δεδομένων. Αυτό το ενδιάμεσο στρώμα είναι τα Web Services. Η χρησιμότητα των Web Services είναι ότι με το WEB API το οποίο φανερώνει στις εφαρμογές πελάτες τα δεδομένα της βάσης με ελεγχόμενο τρόπο. Τα web services εμφανίζουν τα διάφορα δεδομένα στους client ως διάφορα Resources. Για να γίνει η υλοποίηση των web services έπρεπε να χρησιμοποιηθεί μια γλώσσα προγραμματισμού η οποία να τρέχει μαζί με έναν Web Server. Η ανάγκη για ύπαρξη ενός Web Server είναι ότι η επικοινωνία των εφαρμογών πελατών γίνεται με το Web Service μέσω του HTTP πρωτοκόλλου. Έτσι, οι εφαρμογές πελάτες όπως η Android εφαρμογή της πτυχιακής εργασίας θα μπορούν μέσω του διαδικτύου να επικοινωνούν με αυτό. Στην επικοινωνία αυτήν, όλα τα δεδομένα είτε λήψης είτε αποστολής γίνονται μέσα από τα αιτήματα του HTTP πρωτοκόλλου. Επίσης και η κρυπτογράφηση και η ασφάλεια ελέγχεται μέσα από το πρωτόκολλο με την χρήση SSL αλλά και διάφορων κεφαλίδων οι οποίες επιτρέπουν ή αποτρέπουν την πρόσβαση στο API. Επίσης σημαντικό σημείο είναι η υποστήριξη του SSL από τον Web Server. Ο Web Server ο οποίος επιλέχθηκε είναι ο Apache. Για την χρήση SSL χρησιμοποιήθηκε ένα SSL πιστοποιητικό το οποίο θα ορίζει την κωδικοποίηση του HTTP μηνύματος. Επίσης σημαντικό είναι ότι ο Apache περιλαμβάνει χρήσιμα αρχεία καταγραφής των διάφορων αιτημάτων, πράγμα το οποίο είναι σημαντική βοήθεια για τον προγραμματιστή για την αποσφαλμάτωση του προγράμματος.

Η γλώσσα η οποία θα χειριστεί την πρόσβαση στην βάση δεδομένων, αλλά και την αποκωδικοποίηση των HTTP μηνυμάτων είναι η PHP. Χάρη στην PHP χρησιμοποιούμε μια σύνδεση στην βάση μέσω του driver MySQLi και στην συνέχεια επιτρέπει την εκτέλεση ερωτημάτων. Το web Service είτε στέλνει είτε λαμβάνει δεδομένα αυτά βρίσκονται με την μορφή JSON το οποίο στέλνεται μέσω του σώματος του HTTP μηνύματος.

Στην συνέχεια υπάρχει ανάγκη για σύνδεση της Android εφαρμογής με το Web Service. Για αυτόν τον λόγο χρησιμοποιήθηκε μια βιβλιοθήκη η οποία να μπορεί να εκτελέσει HTTP αιτήματα. Η βιβλιοθήκη είναι η OkHttp. Χάρη σε αυτήν την βιβλιοθήκη μπορούμε να ορίσουμε μια δικιά μας κλάση στην οποία να ορίζουμε την παραμετροποίηση της σύνδεσης μας και να μην μπλέκονται οι βιβλιοθήκες δικτύου με την λογική του ηλεκτρονικού καταστήματος. Τέτοιες παράμετροι είναι ο χρόνος ο οποίος θα διαβάζει δεδομένα από τον Server ή και πόσα δευτερόλεπτα θα κάνει να συνδεθεί στον Server πριν εγκαταλείψει την προσπάθεια. Η αποκωδικοποίηση των δεδομένων του JSON γίνεται με την μετατροπή του κειμένου του HTTP μηνύματος σε JSONObject αντικείμενα. Αυτό έχει ως αποτέλεσμα την ύπαρξη σύνθετων JSONObject αντικειμένων τα οποία είναι δύσκολα στον χειρισμό τους από τον κώδικα. Για αυτόν τον λόγο υπήρξε ανάγκη για μια ακόμα αλλαγή της μορφής των δεδομένων. Από αντικείμενα JSONObject ή JSONArray θα πρέπει να γίνει η μετατροπή σε POJO. Η μετατροπή αυτή θα χρειαζόταν κάποιον parser για να μπορέσει να γίνει αν δεν βρισκόταν η βιβλιοθήκη Gson. Η βιβλιοθήκη αυτή μπορεί στις model κλάσεις (π.χ. Order, Category) να κάνει serialize και deserialize τα δεδομένα από και προς JSON. Έτσι πλέον τα δεδομένα είναι διαθέσιμα σε αντικείμενα.

Η Android εφαρμογή όσον αφορά την σχεδίαση της διεπιφάνειας περιλαμβάνει διαχείριση προϊόντων, διαχείριση παραγγελιών, διαχείριση αναφορών με ραβδόγραμμα για το επίπεδο των πωλήσεων, αλλά επίσης επιτρέπει και την προσθήκη νέων καταστημάτων και την προσθήκη νέων υπαλλήλων. Σημαντικές βιβλιοθήκες για την ανάπτυξη των διεπιφανειών είναι οι βιβλιοθήκες Picasso και MPAndroidChart. Η πρώτη είναι ιδιαίτερα χρήσιμη για την απευθείας διαχείριση των εικόνων από το διαδίκτυο και η δεύτερη για την δημιουργία ραβδογραμμάτων. Όσον αφορά την πρώτη βιβλιοθήκη έχει σημασία ότι διαχειρίζεται μόνη της την λήψη της εικόνας, καθώς στο Android δεν υπάρχει άμεση πρόσβαση στο διαδίκτυο από το main Thread της εφαρμογής. Με την χρήση αυτής της βιβλιοθήκης γίνεται αυτό το κατέβασμα και απευθείας χρήση της εικόνας. Για τον εμπλουτισμό της διεπιφάνειας του χρήστη χρησιμοποιήθηκαν και άλλα στοιχεία εκτός από τα βασικά στοιχεία που περιέχει η βασική εργαλειοθήκη του Android. Τέτοια είναι τα DatePickerDialogs, AlertDialog αλλά και animations για την καλύτερη εμπειρία του χρήστη. Επίσης σε κάθε ενέργεια υπήρχε άμεση αντίδραση της εφαρμογής με διάφορα SnackBar αλλά και Toast τα οποία ειδοποιούσαν τον χρήστη για το αποτέλεσμα της κάθε διαφορετικής ενέργειας που έκανε στην εφαρμογή Android.

Επίσης σημαντικό σημείο είναι ότι ανάλογα με τον τύπο χρήστη που έκανε σύνδεση στο σύστημα, υπήρχε και διαφορετική σχεδίαση και διαφορετικός τρόπος περιήγησης. Και στην σύνδεση του Manager και στην σύνδεση του Administrator υπήρχε πάντα κάποιο στοιχείο που ήταν ο Container που θα προβάλει το αντίστοιχο Fragment, ανάλογα με την επιλογή που κάνει ο χρήστης στην περιήγηση. Κάθε διαφορετική διεπιφάνεια αποτελούνταν από ένα διαφορετικό Fragment το οποίο είχε και δική του σχεδίαση. Για παράδειγμα, στο Activity που περιλαμβάνει τα Fragments του Administrator υπήρχε ένα BottomNavigationView και στο Activity που περιέχει τα Fragment του Manager υπάρχει ένα Navigation Drawer το οποίο περιλαμβάνει τις διάφορες επιλογές που έχει για να περιηγηθεί ο Manager. Επίσης ένα πολύ σημαντικό σημείο είναι ότι μεταξύ των model κλάσεων και της διεπιφάνειας είναι οι κλάσεις Adapter. Αυτές οι κλάσεις αποτελούν ένα ενδιάμεσο σημείο το οποίο θα εμφανίζει τα διάφορα δεδομένα τα οποία έχουμε στο View των Fragments. Συνήθως με την χρήση Adapter δημιουργούμε δικά μας custom Views. Τέτοια παραδείγματα είναι η δημουργία ενός CustomListView το οποίο να εμφανίζει σε κάθε γραμμή ένα διαφορετικό προϊόν. Το απλό ListView θα εμφάνιζε απλά μια λίστα από συμβολοσειρές, ενώ έτσι μπορούμε για κάθε πεδίο του ListView να

ορίσουμε ένα δικό μας View το οποίο να παραμετροποιούμε και να του ορίζουμε την δική μας σχεδίαση. Εκτός από CustomListView θα εμφανιστεί παράδειγμα και ενός Custom Expandable List το οποίο θα εμφανίζει τις παραγγελίες για διαχείριση.

Στο κεφάλαιο 2 θα φανεί αναλυτικά η διαδικασία με την οποία θα διαλεχτεί η βάση δεδομένων η οποία θα αποτελεί τον αποθηκευτικό χώρο της εφαρμογής μας ενώ στο κεφάλαιο 3 θα υπάρχει λεπτομερής ανάλυση του σχεδιασμού και την υλοποίηση των web service. Το τρίτο κεφάλαιο της πτυχιακής εργασίας θα περιλαμβάνει τα βασικά στοιχεία την Android εφαρμογής το οποίο θα βασίζεται στις προηγούμενες υλοποιήσεις των web services και στην βάση δεδομένων.

Κεφάλαιο 2ο: Βάση Δεδομένων

2.1 Εισαγωγή

Για την διαχείριση ολόκληρου του e-shop υπάρχει η ανάγκη ενός μόνιμου αποθηκευτικού χώρου. Αυτός ο χώρος είναι απαραίτητος για την διαχείριση του e-shop όσον αφορά το κομμάτι λειτουργίας των διαδικασιών του που χρειάζονται σε ένα ηλεκτρονικό κατάστημα καθώς και για την αποθήκευση των δεδομένων των πελατών. Εφόσον μιλάμε για ένα ηλεκτρονικό κατάστημα και η εφαρμογή διαχείρισης Android του καταστήματος θα τρέχει σε πολλές συσκευές παράλληλα, θα πρέπει όλες αυτές οι client εφαρμογές να μπορούν να συγχρονίζονται μεταξύ τους όσον αφορά τα βασικά κομμάτια του e-shop, όπως τα διαθέσιμα προϊόντα, οι εκκρεμείς παραγγελίες και διάφορα άλλα πρέπει να είναι διαθέσιμα ανά πάσα στιγμή. Για το κομμάτι της αποθήκευσης, της μορφοποίησης καθώς και της συσχέτισης των δεδομένων χρησιμοποιήθηκε μια βάση δεδομένων. Αυτή η μορφή αποθήκευσης των δεδομένων βοηθάει τον προγραμματιστή να έχει μαζική προσπέλαση στα δεδομένα της βάσης μέσω κάποιας άλλης εφαρμογής. Στην περίπτωση της πτυχιακής η εφαρμογή που κάνει προσπέλαση στα δεδομένα είναι το back-end project, δηλαδή τα Web Services τα οποία στην πορεία θα κάνουν διαθέσιμα τα δεδομένα στις client εφαρμογές.

2.2 Δομή DBMS

Η MariaDB είναι μια σχεσιακή βάση δεδομένων η οποία μας δίνει την δυνατότητα να αποθηκεύσουμε τα δεδομένα σε ξεχωριστούς πίνακες με λογική μορφοποίηση. Η MariaDB ομαδοποιεί τα δεδομένα σε φυσικά αρχεία στο filesystem βελτιστοποιώντας την ταχύτητα του DBMS (Database Management System) καθώς και περιλαμβάνει διάφορα λογικά αντικείμενα όπως databases, tables, columns τα οποία βοηθάνε τον προγραμματιστή στην οργάνωση. Μια σημαντική επίσης λειτουργία που παρέχει το DBMS είναι η σύνδεση των διάφορων πεδίων με λογικές σχέσεις όπως ένα προς ένα (one-to-one), ένα προς πολλά (one-to-many), μοναδικό (unique), απαραίτητο (required) ή προαιρετικό (optional).

Όσον αφορά την αποθήκευση των δεδομένων στην βάση ιδιαίτερη αναφορά πρέπει να γίνει στην αρχιτεκτονική της MariaDB η οποία υποστηρίζει την επιλογή διαφορετικών storage engines, όπως CSV, MyIsam, Aria καθώς και διάφορα άλλα. Το storage engine το οποίο χρησιμοποιήθηκε για την πτυχιακή εργασία είναι το default storage engine που χρησιμοποιεί η MariaDB, δηλαδή η InnoDB. Η InnoDB είναι ένα storage engine γενικής χρήσης και παρέχει διάφορα χαρακτηριστικά ακολουθώντας το μοντέλο ACID, όπως commit, rollback τα οποία θα χρειαστούν για να προστατεύσουν τα δεδομένα του χρήστη σε περίπτωση που καταρρεύσει το σύστημα. Όλα αυτά τα χαρακτηριστικά υιοθετήθηκαν από την MariaDB που χρησιμοποιείται στην εφαρμογή.

2.3 Λόγοι χρησιμοποίησης MariaDB

Ιδιαίτερη αναφορά πρέπει να γίνει στους λόγους που επιλέχθηκε η MariaDB ως DBMS για να λειτουργεί με το backend project. Για αρχή λόγω του ότι ομαδοποιεί τα δεδομένα σε αρχεία και χωρίς να απασχολεί τον προγραμματιστή. Δηλαδή το DBMS έχει τα δεδομένα οργανωμένα ώστε να υπάρχει μεγαλύτερη ταχύτητα στην πρόσβαση τους αλλά και φιλικά προς τον προγραμματιστή για την διαχείριση τους. Αυτός είναι και ο βασικός λόγος που χρησιμοποιούνται τα DBMS, διότι με την χρήση της SQL (Stuctured Query Language) ο προγραμματιστής μπορεί να κάνει διάφορες διεργασίες στα δεδομένα του.

Δεύτερος αλλά πολύ σημαντικός λόγος είναι πως ο προγραμματιστής μπορεί πολύ εύκολα να κρατήσει αντίγραφο ασφαλείας και να κάνει ανάκτηση των δεδομένων της εφαρμογής. Γενικά είναι πολύ σύνηθες να υπάρχει ένα πρόβλημα υλικού ή κάποια κατάρρευση του συστήματος για αυτό και σε τέτοιες περιπτώσεις ο διαχειριστής πρέπει να μπορεί να κρατήσει διάφορα ήδη αντιγράφων. Για

την υποστήριξη μιας εμπορικής εφαρμογής είναι ιδιαίτερα κρίσιμο σημείο η ακεραιότητα των δεδομένων. Σε περίπτωση απώλειας δεδομένων όπως μιας παραγγελίας υπάρχει ζήτημα οικονομικής απώλειας. Εκτός από την απώλεια τον δεδομένων υπάρχει και το ενδεχόμενο της κλοπής αυτών των δεδομένων, για αυτό και ο διαχειριστής του DBMS πρέπει να φροντίζει να υπάρχει περιορισμένη πρόσβαση στην βάση. Αυτή η προστασία μπορεί να επιτευχθεί με την χρήση κωδικών για τους διάφορους χρήστες που έχουν πρόσβαση στην βάση δεδομένων καθώς και με την κρυπτογράφηση διάφορων ευαίσθητων δεδομένων της εφαρμογής. Τέτοια δεδομένα είναι οι αριθμοί πιστωτικών καρτών, καθώς και τα προσωπικά στοιχεία των χρηστών. Τέτοιες απώλειες θα πρέπει να αποφευχθούν, καθώς εγείρουν και οικονομικά και νομικά προβλήματα για την επιχείρηση που χρησιμοποιεί την εφαρμογή. Για αυτό τον λόγο πρέπει να χρησιμοποιείται ασφάλεια και σε επίπεδο DBMS.

2.4 Ανάλυση απαιτήσεων εφαρμογής και σχεδιασμός Βάσης δεδομένων

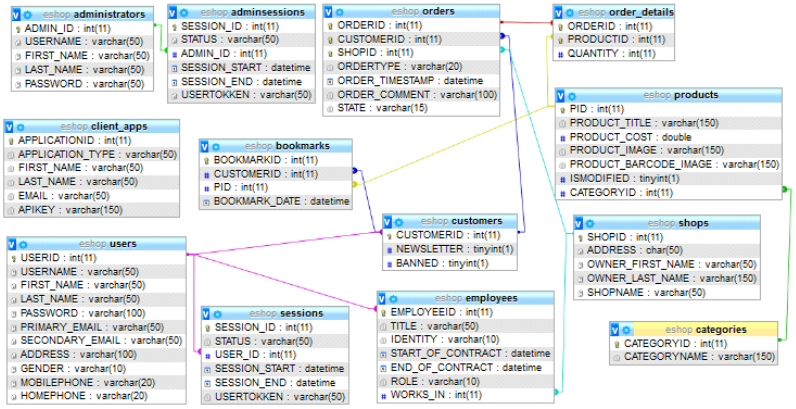
Αρχικά πολύ σημαντικό κομμάτι για την κατάλληλη σχεδίαση της βάσης δεδομένων είναι η ανάλυση των απαιτήσεων της εφαρμογής. Αυτό είναι ένα πολύ σημαντικό κομμάτι στην ανάπτυξη της εφαρμογής, γιατί ο προγραμματιστής κάνει ένα πλάνο για τις διαθέσιμες λειτουργίες που θα έχει η εφαρμογή και προχωράει για την υλοποίηση χρονοδιαγράμματος του έργου. Όσον αφορά την πτυχιακή για την λειτουργικότητα του e-shop πρέπει να υπάρχουν τα δεδομένα ολόκληρου του καταστήματος, είτε παρέχεται από τους υπαλλήλους είτε παρέχεται από τους πελάτες του ηλεκτρονικού καταστήματος είτε από οποιαδήποτε άλλον ο οποίος έχει πρόσβαση στα δεδομένα με έμμεσο ή άμεσο τρόπο. Δηλαδή πάνω στην σχεδίαση της βάσης δεδομένων και κατ’ επέκταση του Web API μπορούν να στηθούν και άλλα είδη εφαρμογών όπως π.χ. Ενός customer app το οποίο θα χρησιμοποιούν οι πελάτες για την πραγματοποίηση των αγορών τους. Στην προκειμένη περίπτωση της πτυχιακής θα υλοποιηθεί μια client side εφαρμογή η οποία θα την χρησιμοποιούν οι υπάλληλοι του μαγαζιού. Στον παρακάτω πίνακα (Πίνακας 1 Λειτουργικές απαιτήσεις) αναφέρονται οι λειτουργικές απαιτήσεις της εφαρμογής. Κάθε μια λειτουργική απαίτηση έχει υλοποιηθεί και είναι απαραίτητη για την εξαγωγή των πινάκων της βάσης δεδομένων.

Πίνακας 1 Λειτουργικές απαιτήσεις

Πακέτο Εργασίας	Περιγραφή απαίτησης
Γενικά	Σύνδεση χρήστη στην εφαρμογή με χρήση username password
Γενικά	Εμφάνιση ειδοποίησης νέας παραγγελίας on boot
Manager	Πληροφόρηση χρήστη με την χρήση manager intro page
Manager	Περιήγηση στις λειτουργίες του manager με χρήση Navigation Drawer
Manager	Έλεγχος επιπέδων πωλήσεων με την προβολή ραβδογράμματος
Manager	Διαχείριση παραγγελιών (προβολή και επικύρωση)
Manager	Διαχείριση προϊόντων (προβολή)
Manager	Αποσύνδεση χρήστη
Administrator	Διαχείριση υπαλλήλων (προσθήκη και προβολή)
Administrator	Διαχείριση καταστημάτων (προσθήκη και προβολή)
Administrator	Προβολή στοιχείων server

Administrator	Δημιουργία φόρμας submit bug
Administrator	Αποσύνδεση χρήστη

Αμέσως μετά το επόμενο στάδιο είναι η σχεδίαση του διαγράμματος ER (Entity Relationship) όπως φαίνεται παρακάτω (Σχήμα 1. Διάγραμμα ER)το οποίο θα περιγράφει την μορφή με την οποία θα αποθηκεύονται τα στοιχεία στην βάση δεδομένων, αλλά και με ποιον τρόπο θα συνδέονται μεταξύ τους. Η βάση δεδομένων περιλαμβάνει 13 πίνακες και 5 stored procedures. Οι πίνακες είναι οι administrators, adminsessions, orders, order_details, client_apps, bookmarks, customers, products, users, sessions, employees και categories.



Σχήμα 1. Διάγραμμα ER

2.5 Πίνακες Βάσης Δεδομένων

2.5.1 Shops

Μπαίνοντας στην λογική οργάνωση και ομαδοποίηση των δεδομένων, πρώτος πίνακας είναι ο SHOPS. Η λογική είναι πως παρόλο που υπάρχει ένα ενιαίο ηλεκτρονικό κατάστημα, μπορεί να υπάρχουν παραπάνω από ένα φυσικά καταστήματα. Για αυτό και κάθε μια εγγραφή σε αυτόν τον πίνακα σημαίνει και ένα πραγματικό υπαρκτό κατάστημα. Τα πεδία τα οποία περιλαμβάνει ο πίνακας είναι τα εξής: ένα κύριο κλειδί SHOPID, το αποτελεί τον αύξων αριθμό του καταστήματος, στην συνέχεια ακολουθούν τα βασικά στοιχεία του καταστήματος, τα οποία είναι η διεύθυνση (ADDRESS), το μικρό όνομα του ιδιοκτήτη (OWNER_FIRST_NAME), το επώνυμο του ιδιοκτήτη (OWNER_LAST_NAME), καθώς και το όνομα του καταστήματος (SHOPNAME).

2.5.2 Categories

Προσπαθώντας να μοντελοποιήσουμε όλο το κατάστημα, δεν γινόταν να μην γίνει αναφορά στις διάφορες κατηγορίες των προϊόντων που μπορεί να έχει το κατάστημα. Χάρη στην χρήση αυτού του πίνακα, ο υπάλληλος μπορεί να προσθέτει κατηγορίες προϊόντων και στην συνέχεια να μπορεί να κάνει περιήγηση στα διάφορα προϊόντα μέσω της android εφαρμογής. Τα πεδία τα οποία τα οποία περιλαμβάνει ο πίνακας είναι ένα κύριο κλειδί (CATEGORYID) το οποίο θα προσδιορίζει μοναδικά κάθε διαφορετική κατηγορία. Το δεύτερο πεδίο του πίνακα είναι το CATEGORYNAME που θα αποθηκεύει το όνομα της κάθε κατηγορίας.

2.5.3 Products

Οι κατηγορίες δημιουργήθηκαν με την λογική της προσπάθειας ομαδοποίησης των διάφορων προϊόντων που έχουν τα καταστήματα. Για αυτό τον λόγο υπάρχει άμεση σχέση μεταξύ των δυο αυτών entities. Η σχέση μεταξύ αυτών των πινάκων ορίζεται με την ύπαρξη ενός ξένου κλειδιού στον πίνακα PRODUCTS που δείχνει στο πεδίο CATEGORYID του πίνακα CATEGORY. Με αυτό τον τρόπο μπορούμε να διασφαλίσουμε πως κάθε προϊόν βρίσκεται κάτω από κάποιο CATEGORY. Αμέσως μετά είναι σημαντικό να έχουμε ένα κύριο κλειδί (PID) το οποίο θα προσδιορίζει μοναδικά κάθε διαφορετικό προϊόν. Αμέσως μετά θέλουμε να έχουμε ένα όνομα για το προϊόν (PRODUCT_TITLE), καθώς και ένα πεδίο κόστους για το προϊόν PRODUCT_COST. Ιδιαίτερο ενδιαφέρον παρουσιάζει ο τρόπος αποθήκευσης της φωτογραφίας του προϊόντος στην βάση δεδομένων. Η ύπαρξη της φωτογραφίας είναι πολύ σημαντική για το προϊόν καθώς μπορεί ένας πελάτης να αξιολογήσει ένα προϊόν όχι μόνο από τον τίτλο του, αλλά και από την εικόνα του. Γενικά υπάρχει η επιλογή της αποθήκευσης στην βάση δεδομένων, αλλά εφόσον μία εικόνα είναι αρκετά μεγάλου μεγέθους, μεγαλύτερη από 1,2 MB προσθέτουμε φόρτο εργασίας στο DBMS και αυτό να ενδέχεται να ρίξει την απόδοση του. Για αυτό τον λόγο η εικόνα μόλις λαμβάνεται από το WEB API αποθηκεύεται στο file storage του web server. Στην βάση δεδομένων θα υπάρχει απλά ένα πεδίο PRODUCT_IMAGE το οποίο θα αποθηκεύει την διαδρομή της εικόνας στο file system.

2.5.4 Bookmarks

Ένα άλλο σημαντικό κομμάτι στην διαχείριση ενός μαγαζιού είναι να μπορεί να διαχειρίζεται τα προϊόντα του έχοντας γνώση του πια είναι πιο συχνά αποθηκευμένα από τους χρήστες του ηλεκτρονικού καταστήματος. Αυτή η γνώση επίσης αποθηκεύεται στην βάση δεδομένων στον πίνακα BOOKMARKS. Αυτός ο πίνακας μπορεί να είναι χρήσιμος για την δημιουργία διαγραμμάτων για τους managers του καταστήματος. Επίσης είναι χρήσιμος για την μελλοντική χρήση και εφαρμογή του ενός recommendation system, όπου με χρήση αλγορίθμων να μπορεί να προτείνει προϊόντα του μαγαζιού τα οποία να ενδιαφέρουν τον πελάτη.

Ο πίνακας αυτός κρατάει 3 βασικά πεδία τα οποία απαιτεί ένας σελιδοδείκτης, ένα κύριο κλειδί BOOKMARKID το οποίο να προσδιορίζει μοναδικά τον σελιδοδείκτη μεταξύ των σελιδοδεικτών όλων των χρηστών. Ένα ξένο κλειδί CUSTOMERID το οποίο αναφέρεται στο κύριο κλειδί CUSTOMERID του πίνακα customers. Αυτό συμβαίνει για να προσδιορίσουμε ποιος πελάτης έχει αποθηκεύσει τον συγκεκριμένο σελιδοδείκτη. Στην συνέχεια το τρίτο πεδίο είναι το πεδίο PID το οποίο είναι ξένο κλειδί στον πίνακα PRODUCTS στην στήλη PID για να μπορέσουμε να ξέρουμε ποιο προϊόν είναι αποθηκευμένο σαν σελιδοδείκτης. Τέλος, το τελευταίο πεδίο είναι το BOOKMARK_DATE μια ημερομηνία, η οποία θα υποδηλώνει το πότε αποθηκεύτηκε το συγκεκριμένο bookmark από τον χρήστη. Συνοψίζοντας, σε αυτόν τον πίνακα αποθηκεύεται ποιο προϊόν αποθηκεύτηκε σαν σελιδοδείκτης, από ποιον πελάτη και πότε.

2.5.5 Administrators

Σημαντικό κομμάτι της διαχείρισης ενός e-shop είναι και η τεχνική υποστήριξη του. Συνεπώς θα πρέπει να υπάρχει και σχετική πρόσβαση από τεχνικούς για καλή ρύθμιση των τεχνικών λεπτομερειών του ηλεκτρονικού καταστήματος. Συνεπώς, θα πρέπει να αποθηκευτεί στην σχεσιακή βάση δεδομένων η πληροφορία για την ύπαρξη σχετικών χρηστών του συστήματος. Ο πίνακας αρχικά έχει ένα κύριο κλειδί ADMIN_ID για τον μοναδικό προσδιορισμό του κάθε διαχειριστή. Αμέσως μετά, σημαντικό είναι να υπάρχει ένα πεδίο για το όνομα του χρήστη του διαχειριστή, το οποίο έχει το

όνομα USERNAME το οποίο θα μας χρειαστεί σε συνδυασμό με το άλλο πεδίο PASSWORD για τον προσδιορισμό της ταυτότητας του χρήστη. Αμέσως μετά υπάρχουν και κάποιες επιπλέον πληροφορίες σαν πεδία στον πίνακα. Το πρώτο είναι η στήλη FIRST_NAME που αποτελεί το όνομα του χρήστη καθώς και η στήλη LAST_NAME που δηλώνει το επώνυμο του διαχειριστή.

2.5.6 Admin_Sessions

Ένα άλλο κομβικό σημείο για την λειτουργία της ταυτοποίησης του χρήστη είναι ένας πίνακας ADMIN_SESSIONS. Αυτός ο πίνακας είναι κρίσιμος για την συντήρηση του ηλεκτρονικού καταστήματος, αλλά και για την ασφάλεια του. Γενικά είναι καλό να μην υπάρχει συνεχής μεταφορά των κωδικών χρηστών και των ονομάτων τους. Για αυτό όπως θα αναλυθεί παρακάτω η ταυτοποίηση των χρηστών να γίνεται με την χρήση ενός ειδικού token το οποίο θα μεταφέρεται μέσω του API για κάθε call για να αντιλαμβάνεται το e-shop ότι κάθε API call είναι από χρήστη συνδεδεμένο στο σύστημα. Συγκεκριμένα αυτός ο πίνακας κρατάει τα sessions των διαχειριστών.

Αρχικά το πρώτο πεδίο και κύριο κλειδί είναι το SESSION_ID το οποίο είναι auto increment και προσδιορίζει μοναδικά κάθε session που έχει ανοίξει ένας διαχειριστής, ότι δηλαδή είναι συνδεδεμένος στο σύστημα και μπορεί να εκτελεί calls στο web web service που έχει αναπτυχθεί. Αμέσως επόμενο πεδίο είναι το πεδίο STATUS, το οποίο δείχνει σε τι κατάσταση είναι το συγκεκριμένο session entry. Αν είναι CLOSED σημαίνει ότι ο συγκεκριμένος διαχειριστής έχει κάνει logout από το σύστημα και πλέον δεν μπορεί να εκτελεί calls με την χρήση του συγκεκριμένου token που έλαβε όταν ξεκίνησε το session. Στην περίπτωση που είναι CONNECTED το συγκεκριμένο token είναι ακόμα ενεργό και έχει πρόσβαση στο API. Αυτό μας οδηγεί στο επόμενο πεδίο το οποίο είναι το πεδίο USERTOKKEN το οποίο θα αποθηκεύει το generated token που υπάρχει με το που ξεκινήσει μια σύνδεση από έναν διαχειριστή. Η αποθήκευση αυτών των token είναι απαραίτητα σαν log, καθώς και για troubleshooting στο μέλλον του web service για την επίλυση ζητημάτων ασφαλείας. Δεδομένου ότι κάθε token συσχετίζεται άμεσα με ένα username μπορεί πολύ εύκολα να διαπιστωθεί από ποιον χρήστη έγινε κάθε ενέργεια. Τελευταία πεδία, αλλά επίσης χρήσιμα είναι τα SESSION_START και SESSION_END τα οποία θα δείχνουν την αρχή και το τέλος της κάθε συνεδρίας. Στην περίπτωση που απλά είναι CONNECTED το πεδίο SESSION_END είναι απλά κενό.

2.5.7 Users

Καθώς και έχει γίνει ήδη αναφορά στην ύπαρξη ρόλων στο σύστημα, θα έπρεπε να υπάρξει κάποια μοντελοποίηση των βασικών στοιχείων του χρήστη. Για αυτό το λόγο θα πρέπει στο schema της βάσης δεδομένων να έχει ένα πίνακα με χρήστες. Σε προηγούμενο κομμάτι αναφέρθηκε και περιγράφηκε ένας πίνακας, ο οποίος αποθηκεύει τα στοιχεία των διαχειριστών και είναι καλό να αναφερθεί η διαφοροποίηση των διάφορων ρόλων που περιλαμβάνει το σύστημα. Γενικά, το σύστημα περιλαμβάνει διαχειριστές, υπαλλήλους και πελάτες. Στον πίνακα users είναι αναγκαίο να τονισθεί πως περιλαμβάνονται μόνο στοιχεία των υπαλλήλων και των πελατών και όχι των διαχειριστών, για λόγους ασφαλείας. Αυτό γιατί μπορεί πολύ εύκολα με κάποιο λάθος σε κάποιο query να υπάρξει διαρροή δεδομένων των διαχειριστών.

Αρχικά το πρώτο πεδίο είναι το κύριο κλειδί με όνομα USERID και προσδιορίζει μοναδικά κάθε χρήστη. Αμέσως μετά αποθηκεύονται διάφορα άλλα στοιχεία ενός χρήστη. Τα αμέσως επόμενα πεδία τα οποία θα χρησιμοποιηθούν στην σύνδεση ενός χρήστη στην βάση δεδομένων, είναι τα πεδία USERNAME και PASSWORD, αυτά τα δυο πεδία αποθηκεύουν το όνομα χρήστη και τον προσωπικό του κωδικού προκειμένου να εισέλθει στο σύστημα. Συμπληρωματικά πεδία τα οποία χρησιμεύουν

στην καλύτερη μοντελοποίηση ενός χρήστη, είναι το FIRST_NAME και το LAST_NAME τα οποία αποθηκεύουν το όνομα και το επώνυμο του κάθε χρήστη αντίστοιχα. Ιδιαίτερη υπενθύμιση πρέπει να γίνει στο ότι ο πίνακας αυτός είναι αρκετά γενικός όποτε οι διάφοροι τύποι χρηστών που μπορεί να υπάρχουν πρέπει να μοντελοποιηθούν σε άλλους πίνακες που θα συνδέονται με τον πίνακα USER και θα αποτελούν εξειδικεύσεις τους και θα συνδέονται με αυτόν με την σχέση ISA.

2.5.8 Sessions

Όπως και οι χρήστες διαχειριστές και άλλοι χρήστες θα πρέπει να έχουν έναν βασικό πίνακα που να αποθηκεύει στοιχεία των συνδέσεων αυτών. Οπότε αντίστοιχα, ο πίνακας SESSIONS θα έχει ένα κύριο κλειδί SESSION_ID, το οποίο θα προσδιορίζει μοναδικά κάθε περίπτωση σύνδεσης χρήστη στο σύστημα. Το πεδίο είναι auto increment και ουσιαστικά η μέγιστη τιμή που έχει η στήλη αυτή, αποτελεί και το πλήθος των επιτυχών συνδέσεων στο σύστημα. Αμέσως μετά υπάρχει μια στήλη STATUS που προσδιορίζει αν εκείνο το session έχει κλείσει. Αν έχει κλείσει, θα έχει την τιμή CLOSED, αλλιώς θα είναι CONNECTED. Αμέσως μετά υπάρχει ένα ξένο κλειδί από τον πίνακα SESSIONS στον πίνακα USERS, με το όνομα USER_ID που προσδιορίζει ποιος χρήστης συνδέθηκε. Σε αυτό το σημείο πρέπει να τονιστεί πως σαν USERS αναφέρονται και οι υπάλληλοι και οι πελάτες. Για το κομμάτι της σύνδεσης υπάρχει ένα USERTOKKEN, το οποίο θα κρατάει κάθε client εφαρμογή, όταν ο κάθε χρήστης κάνει σύνδεση από την συσκευή του. Τα δύο τελευταία πεδία τα οποία χρειάζονται είναι ένα SESSION_START και ένα SESSION_END τα οποία δείχνουν την χρονική στιγμή που έκανε σύνδεση και αποσύνδεση ο χρήστης.

2.5.9 Customers

Όπως αναφέρθηκε πιο πάνω, για την διαχείριση των χρηστών υπάρχει ο πίνακας USERS. Για αυτό τον λόγο ήταν αναγκαίο σε ότι διεργασίες χρειάζεται να κάνει το σύστημα επάνω σε πελάτες, δεν θα πρέπει να έχει απευθείας σύνδεση πάνω σε όλα τα δεδομένα όλων των χρηστών, αλλά θα πρέπει να έχει πρόσβαση μόνο στα δεδομένα πελατών, διευκολύνοντας τον προγραμματιστή από το να κάνει απόκρυψη των προσωπικών στοιχείων των χρηστών. Τα πεδία τα οποία έχει ένας πελάτης είναι καταρχάς ένα κύριο κλειδί CUSTOMERID το οποίο είναι ξένο κλειδί στο πεδίο USER_ID του πίνακα USERS. Ουσιαστικά έτσι διασφαλίζουμε πως κάθε πελάτης είναι χρήστης στο σύστημα. Αμέσως μετά ο πελάτης έχει άλλα 2 πεδία τα οποία έχει ένας πελάτης που τον διαφοροποιούν από τους άλλους χρήστες, το ένα είναι το πεδίο NEWSLETTER το οποίο αποθηκεύει αν ο πελάτης είναι εγγεγραμμένος στο newsletter του καταστήματος και να λαμβάνει ενημερωτικά δελτία για τα περιεχόμενα του καταστήματος. Το τρίτο και τελευταίο πεδίο είναι το πεδίο banned, το οποίο υποδηλώνει εάν ο πελάτης έχει γίνει ban από το κατάστημα για τον οποιονδήποτε λόγο.

2.5.10 Employees

Μια άλλη εξειδίκευση του πίνακα USERS και αποτελεί μια περίπτωση χρήστη που θα συνδεθεί στο κατάστημα είναι ο πίνακας EMPLOYEES. Και αυτός ο πίνακας, όπως και ο CUSTOMERS συνδέεται με τον πίνακα USERS με την σχέση ISA. Η σχέση αυτή επιτυγχάνεται με την χρήση ενός κύριου κλειδιού EMPLOYEEID ο οποίος αποτελεί και ξένο κλειδί στο κύριο κλειδί του πίνακα USERS, ώστε να επιτύχουμε πως κάθε υπάλληλος είναι και χρήστης στο σύστημα. Αμέσως μετά είναι πολύ σημαντικό να αναφερθεί σε ποιο κατάστημα δουλεύει ο υπάλληλος. Το κατάστημα προσδιορίζεται με την χρήση ενός πεδίου WORKS_IN στον πίνακα το οποίο είναι ξένο κλειδί στον πίνακα SHOPS και στο κύριο κλειδί του το SHOPID. Αυτό διασφαλίζει πως κάθε υπάλληλος δουλεύει σε κάποιο κατάστημα. Στην πορεία, ανάλογα με τον υπάλληλο, είναι ανάγκη να υπάρχει μια διάκριση στον ρόλο

που έχει ο κάθε υπάλληλος. Αυτόν τον ρόλο τον καθορίζουμε σε ένα πεδίο ROLE, το οποίο θα περιλαμβάνει τους ρόλους Manager, Salesman και οτιδήποτε άλλο custom role επιθυμεί ο καταστηματάρχης. Βασικά άλλα σημεία στην μοντελοποίηση ενός υπαλλήλου είναι η ύπαρξη του αριθμού ταυτότητα του με την χρήση ενός πεδίου IDENTITY. Τέλος άλλα δυο πεδία είναι απαραίτητα για αποθήκευση για έναν υπάλληλο, τα οποία και τα δυο είναι πεδία ημερομηνίας START_OF_CONTRACT και END_OF_CONTRACT τα οποία θα αποθηκεύουν την έναρξη και την λήξη του συμβολαίου αντίστοιχα.

2.5.11 Orders

Το ποιο κομβικό σημείο στην δημιουργία ενός e-shop είναι η ικανότητα δημιουργίας παραγγελιών. Αυτό έχει να κάνει με τον θεσμικό ρόλο του καταστήματος που είναι το κέρδος. Συνεπώς κάθε απώλεια παραγγελίας, κάθε δυσκολία στην διεκπεραίωση παραγγελιών μεταφράζεται σε απώλεια χρημάτων. Για αυτόν τον λόγο η σωστή μοντελοποίηση οδηγεί στην καλύτερη αποσφαλμάτωση στο μέλλον, στην αποφυγή λαθών καθώς και στην ασφάλεια των δεδομένων.

Για αρχή υπάρχει ένα κύριο κλειδί ORDERID, το οποίο προσδιορίζει μοναδικά κάθε διαφορετική παραγγελία που υπάρχει στο σύστημα. Αμέσως μετά χρειάζονται κάποιες βασικές πληροφορίες όπως το ποιος πελάτης έχει κάνει την παραγγελία καθώς και σε ποιο κατάστημα έγινε η παραγγελία καθορίζονται με 2 πεδία CUSTOMERID και SHOPID τα οποία είναι ξένα κλειδιά στα κύρια κλειδιά των CUSTOMERS και SHOPS. Αμέσως μετά είναι ανάγκη να προσδιορίσουμε το είδος της παραγγελίας για να καθοριστεί ο τρόπος παραλαβή, από τον πελάτη. Στην προκειμένη περίπτωση μιλάμε για ένα πεδίο με όνομα ORDERTYPE το οποίο θα έχει delivery σαν είδος παραγγελίας. Επίσης ένα άλλο πεδίο που χρειάζεται είναι το πεδίο comment το οποίο μπορεί να παραθέσει ο πελάτης σαν οποιαδήποτε πληροφορία σχετικά με την παραγγελία.

Ένα ίσως κρίσιμο σημείο το οποίο έχει να κάνει με την διαχείριση των παραγγελιών είναι η κατάσταση της παραγγελίας, το οποίο προσδιορίζεται από το πεδίο STATE. Η κατάσταση της παραγγελίας μπορεί να είναι είτε Pending, είτε New είτε Send. Ως New είναι οι παραγγελίες οι οποίες έχουν έρθει, αλλά δεν έχουν διαβαστεί από υπαλλήλους, Pending είναι οι παραγγελίες οι οποίες έχουν διαβαστεί αλλά δεν έχει γίνει αποστολή του προϊόντος στον πελάτη. Τέλος, όταν μια παραγγελία έχει κατάσταση Sent, σημαίνει πως έχει γίνει αποστολή των προϊόντων στον πελάτη. Τελευταίο πεδίο το οποίο χρειάζεται αναφορά είναι η χρονική στιγμή της παραγγελίας. Αυτό δηλώνεται σε ένα πεδίο με όνομα ORDER_TIMESTAMP.

2.5.12 Order_Details

Επόμενος πίνακας ο οποίος παίζει τον δικό του ρόλο είναι ο Order_Details. Αυτός ο πίνακας έχει να κάνει με τις λεπτομέρειες των προϊόντων που θα έχει μια πιθανή παραγγελία. Αρχικά έχει ένα σύνθετο κύριο κλειδί που αποτελείται από τα πεδία ORDERID και PRODUCTID τα οποίο είναι και ξένα κλειδιά στον πίνακα ORDER στο πεδίο ORDERID και στον πίνακα PRODUCTS στο πεδίο PID. Με αυτόν τον τρόπο ορίζουμε πως κάθε έγγραφη στον πίνακα προσδιορίζει μοναδικά κάθε ποσότητα προϊόντος σε μια παραγγελία ενός προϊόντος.

2.5.13 Client_APPS

Ιδιαίτερη αναφορά πρέπει να γίνει στον πίνακα client apps. Σε αυτόν τον πίνακα θα αποθηκεύονται τα στοιχεία όλων των εφαρμογών που θα κάνουν calls στο web service που έχει σχεδιαστεί. Στην προκειμένη περίπτωση, επειδή η μόνη client εφαρμογή που θα κάνει calls στο web είναι η εφαρμογή

android διαχείρισης του e-shop θα υπάρχει μόνο ένα entry, το οποίο περιλαμβάνει τα στοιχεία του developer, δηλαδή FIRST_NAME, LAST_NAME, EMAIL, στην συνέχεια ένα κύριο κλειδί του πίνακα με όνομα APPLICATIONID, το οποίο θα είναι μοναδικό για κάθε εφαρμογή, ένα πεδίο APPLICATION_TYPE το οποίο θα αποθηκεύει το είδος της client side εφαρμογής. Το τελευταίο και πιο σημαντικό πεδίο αυτού του πίνακα είναι το APIKEY, καθώς με αυτό το κλειδί μπορεί η συγκεκριμένη εφαρμογή να είναι έχει πρόσβαση στο API

2.6 Stored Procedures

Η βάση δεδομένων, εκτός από την μοντελοποίηση των δεδομένων, παρέχει επίσης την δυνατότητα του ορισμού συναρτήσεων και μεθόδων οι οποίες είτε να επιστρέφουν τα δεδομένα τα οποία θέλει ο προγραμματιστής είτε αυτά τα οποία θέλει ο σχεδιαστής της να αποκαλύψει στον προγραμματιστή. Στο συγκεκριμένο DBMS δηλαδή την MariaDB υπάρχει η δυνατότητα αυτών των procedures και εφαρμόστηκε με την δημιουργία 5 procedures. Οι procedures που χρησιμοποιήθηκαν είναι οι εξής GetNewOrders(), GetOrderProducts(), GetOrders(), SetOrdersAsPending(), CheckForNewOrders()

2.6.1 CheckForNewOrders()

Για να μπορεί ένας υπάλληλος να ελέγχει τις καινούργιες παραγγελίες που υπάρχουν στο κατάστημα θα πρέπει να υπάρχει ένα query που να ελέγχει κατά πόσο θα υπάρχουν παραγγελίες με κατάσταση New, δηλαδή παραγγελίες οι οποίες θα έχουν καταχωρηθεί από κάποιον πελάτη αλλά δεν θα έχουν διαβαστεί από τον υπάλληλο. Κάνει προσπέλαση στα ίδια δεδομένα με την GetOrders() με την διαφορά ότι αυτή η procedure δεν επιστρέφει όλα τα αποτελέσματα όλων των παραγγελιών, αλλά θα επιστρέφει έναν αριθμό για το πόσες παραγγελίες είναι καινούργιες. Αυτό σχεδιαστικά βοηθάει διότι μπορεί η εφαρμογή android να κάνει web service calls ανά τακτά χρονικά διαστήματα για να ελέγχει αν θα πρέπει να κάνει ένα περαιτέρω web service call για να πάρει νέες παραγγελίες.

2.6.2 GetOrders()

Για την ενημέρωση του υπαλλήλου σχετικά με όλες τις παραγγελίες που υπάρχουν στην βάση δεδομένων του ηλεκτρονικού καταστήματος. Γενικά υπάρχει η δυνατότητα της εκτέλεσης query, από την PHP για να διαβάσει τα δεδομένα από την βάση δεδομένων, αλλά στην προκειμένη περίπτωση επιλέχθηκε να αποθηκευμένο το query στην MariaDB έτσι ώστε να γίνει αποφυγή του περίπλοκου κώδικα στο προγραμματιστικό κομμάτι. Αυτό συμβαίνει καθώς περιλαμβάνει 2 πίνακες με πολλά πεδία και πολλά entries που περιλαμβάνουν ευαίσθητα δεδομένα όπως κωδικοί χρηστών και στοιχεία χρηστών.

2.6.3 SetOrdersAsPending()

Όπως αναφέρθηκε πιο πάνω υπάρχει η ανάγκη του ελέγχου για νέες παραγγελίες. Για αυτό τον λόγο μόλις ελέγξει ο υπάλληλος για νέες παραγγελίες και στην πορεία πάρει τα δεδομένα για όλες τις νέες παραγγελίες θα πρέπει να ορίσει τις παραγγελίες που προηγουμένως δεν είχαν διαβαστεί από τους υπαλλήλους να αλλάξουν κατάσταση και από New να είναι Pending. Αυτόν τον ρόλο επιτελεί αυτό το procedure, δηλαδή να καλεστεί και κάθε παραγγελία που είναι στην βάση και είναι New να γίνει Pending.

2.6.4 GetNewOrders()

Για την καλύτερη λειτουργία του web service είναι ανάγκη να μην υπάρχει άσκοπη μεταφορά δεδομένων μέσω του δικτύου. Για αυτόν τον λόγο και θα πρέπει η client side android εφαρμογή να μπορεί να πάρει μόνο τις νέες παραγγελίες και να μην παίρνει όλες τις παραγγελίες που υπάρχουν στο κατάστημα. Έτσι ουσιαστικά επιστρέφουμε όλες τις νέες παραγγελίες με την χρήση αυτού stored procedure που απλά την καλεί η PHP για να τα στείλει μέσω του κατάλληλου endpoint στην android εφαρμογή.

2.6.5 GetOrderProducts()

Όπως αναφέρθηκε πιο πάνω για την επιστροφή των δεδομένων των παραγγελιών έγινε χρήση των μεθόδων GetNewOrders() και GetOrders(), οι οποίες και οι δυο επιστρέφουν διάφορα στοιχεία των παραγγελιών, αλλά δεν επιστρέφουν τα προϊόντα τα οποία περιλαμβάνονται σε κάθε παραγγελία. Για αυτόν τον λόγο υπάρχει η μέθοδος GetOrderProducts για να μπορέσει να φέρει ποια ποσότητα υπάρχει από κάθε προϊόν στην παραγγελία. Για αυτόν τον λόγο και η procedure παίρνει σαν παράμετρο το ORDERID με βάσει το οποίο θα επιστρέψει τα στοιχεία που αναφέρθηκαν ανεξάρτητα από το STATE της παραγγελίας.

2.7 Επίλογος

Σε αυτό το κεφάλαιο αναλύθηκε η χρησιμότητα του DBMS ως μόνιμο αποθηκευτικό χώρο για τα δεδομένα του ηλεκτρονικού καταστήματος αλλά και η διαμόρφωση της βάσης δεδομένων που περιλαμβάνει. Στην αρχή του κεφαλαίου κάποια στοιχεία της αρχιτεκτονικής της MariaDB. Επιπροσθέτως, αναφέρθηκαν οι βασικοί πίνακες της βάσης δεδομένων καθώς και των διάφορων πεδίων τους. Επίσης αναφορά έγινε στην συσχέτιση όλων αυτών των πινάκων μεταξύ τους με την χρήση των ξένων κλειδιών. Τέλος ειδική αναφορά έγινε στις stored procedures οι οποίες κατά κύριο λόγο δημιουργήθηκαν για την διαχείριση των παραγγελιών του ηλεκτρονικού καταστήματος.

Κεφάλαιο 3ο: Web Services

3.1 Εισαγωγή

Για την πλήρη λειτουργία του e-Shop application και την εξυπηρέτηση πολλαπλών αντιγράφων της android εφαρμογής θα έπρεπε να υλοποιηθεί ένας μηχανισμός ο οποίος θα μπορεί να εξυπηρετεί όλους αυτούς τους clients. Η λύση η οποία υλοποιήθηκε είναι ένα Web API το οποίο θα μπορεί να είναι συνέχεια προσβάσιμο σε έναν Web Server. Πριν την ανάλυση της λειτουργίας του Web API πρέπει να γίνει αναφορά στο είδος του Web server που θα χειριστεί όλες αυτές τις αιτήσεις. Στην περίπτωση μας η λύση είναι ένας Apache HTTP Server. Η ανάγκη ύπαρξης ενός Web Server είναι απαραίτητη καθώς οι Servers είναι σε υπολογιστές συνδεδεμένους στο διαδίκτυο και έχουν μια συγκεκριμένη IP, πράγμα το οποίο καθιστά το Web API μας διαθέσιμο από παντού στο διαδίκτυο. Επίσης σημαντικός παράγοντας είναι ότι οι Web Servers συντηρούνται από third party providers, πράγμα το οποίο απαλλάσσει τον προγραμματιστή από την διαχείριση είτε του λογισμικού είτε του υλικού του.

3.2 Αρχιτεκτονική και HTTP

Μεταξύ του back end και του front end project, χρησιμοποιείται η αρχιτεκτονική client server. Το μηχάνημα που λειτουργεί σαν Web Server είναι μόνιμα διαθέσιμο. Για αυτό τον λόγο είναι πάντα διαθέσιμο για επικοινωνία από άλλες client εφαρμογές, μέσω του HTTP πρωτοκόλλου. Η επικοινωνία μέσω του HTTP υλοποιείται με την ανταλλαγή μηνυμάτων HTTP που έχουν μορφή όπως ορίζει το πρωτόκολλο. Το πρωτόκολλο HTTP βρίσκεται στο 5ο επίπεδο στο μοντέλο του TCP/IP και τρέχει μόνο πάνω στο TCP πρωτόκολλο πράγμα το οποίο εξασφαλίζει ασφάλεια στην μεταφορά των δεδομένων. Για την επίτευξη της επικοινωνίας σε επίπεδο HTTP υπάρχουν 2 είδη μηνυμάτων, το request message και το response message. Κάτι το οποίο χρειάζεται ιδιαίτερη αναφορά είναι ότι το πρωτόκολλο είναι ακαταστατικό, δηλαδή ο HTTP Server δεν κρατάει πληροφορίες για τα όποια requests έχει κάνει ο client στο παρελθόν. Για αυτό τον λόγο πρέπει σε προγραμματιστικό επίπεδο να γίνει η ταυτοποίηση του χρήστη ή της εφαρμογής σε κάθε επόμενο request.

3.3 Χρήση SSL/TLS

Ως HTTPS εννοούμε την χρήση του HTTP με χρήση SSL. Σε αυτή την περίπτωση, αντί για HTTP σε ένα URI χρησιμοποιούμε HTTPS. Στην περίπτωση που χρησιμοποιούμε HTTPS το προκαθορισμένο port που χρησιμοποιείται είναι το 443 αντί για το 80. Για την ασφαλέστερη μεταφορά των δεδομένων από τον client στον server χρησιμοποιείται η αρχιτεκτονική TLS. Το TLS (Transport Secured Layer) αποτελείται από 2 επίπεδα πρωτοκόλλων, το πρωτόκολλο εγγραφής (Record Protocol) και ένα ανώτερο επίπεδο που περιλαμβάνει πρωτόκολλα, χειραψίας, καθορισμού αλλαγής κρυπτογραφήματος, προειδοποίησης και παλμού. Ιδιαίτερη αναφορά πρέπει να γίνει στο πρωτόκολλο εγγραφής SSL το οποίο παρέχει εμπιστευτικότητα και ακεραιότητα στα μηνύματα. Το SSL χρησιμοποιεί συμμετρική κρυπτογράφηση, δηλαδή χρησιμοποιείται ένα κοινό μυστικό κλειδί για κρυπτογράφηση των δεδομένων, πετυχαίνοντας εμπιστευτικότητα. Επίσης χρησιμοποιεί και ένα άλλο κοινόχρηστο μυστικό κλειδί (Message Authentication Code) για την ταυτοποίηση των μηνυμάτων.

Για την ανάπτυξη του Web API χρησιμοποιήθηκε ένα self signed certificate TLS. Η δημιουργία του πιστοποιητικού έγινε με την χρήση του makecert.exe utility των windows. Για το πιστοποιητικό χρησιμοποιήθηκε η σουίτα κρυπτογράφησης ECDHE-RSA-AES256-GCM-SHA384 με

κρυπτογράφηση AESGCM και μήκος κλειδιού 256 bits. Κατά την διάρκεια της λειτουργίας του SSL γίνονται τα εξής βήματα. Για αρχή γίνεται κατάτμηση του μηνύματος διαιρώντας κάθε μήνυμα σε τμήματα των 214 bytes, στην συνέχεια εφαρμόζεται συμπίεση (compression) . Αμέσως μετά υπολογίζεται το mac για την πιστοποίηση του μηνύματος. Τέλος το συμπιεσμένο μήνυμα και ο mac κρυπτογραφούνται με χρήση συμμετρικής κρυπτογράφησης και προστίθενται μια κεφαλίδα μήκους και έκδοσης πριν το στείλει σε ένα TCP segment. Μόλις φτάσει το segment στον παραλήπτη επαναλαμβάνεται η προαναφερθείσα διαδικασία από το τέλος προς την αρχή.

3.4 Υποστήριξη SSL

Για την υποστήριξη του SSL και για να γίνει δημόσιο το Web service χρειάζεται μία συγκεκριμένη παραμετροποίηση. Το αρχείο στον Web Server στο οποίο χρειάζεται να γίνει η προσθήκη είναι το httpd-vhosts.conf. Αυτό γίνεται με την προσθήκη μιας directive όπως φαίνεται στο Σχήμα 2 Directive for SSL support.Σε αυτό το σημείο προσθέτουμε ένα Virtual Host το οποίο να ακούει στην πόρτα 443 του HTTPS. Αμέσως μετά ορίζω και ένα ServerName έτσι ώστε να είναι διαθέσιμο όχι μόνο με IP αλλά και με όνομα το συγκεκριμένο host. Όπως αναφέρθηκε πιο πάνω χρειάστηκε η δημιουργία ενός self signed πιστοποιητικού για να μπορεί να επικοινωνήσει με ασφάλεια ο server με τον client. Το πιστοποιητικό δημιουργήθηκε με το makecert εργαλείο και στην συνέχεια τα ορίζω στο virtual host. Αυτό το πιστοποιητικό θα μας χρειαστεί για να δοθεί και στον client προκειμένου να αναγνωρίζεται και από τον server ότι είναι έμπιστη συσκευή. Επίσης, έχοντας το πιστοποιητικό μπορεί ο client να κάνει κωδικοποίηση και αποκωδικοποίηση τα δεδομένα τα οποία λαμβάνει και στέλνει στον server. Έτσι πλέον θα είναι δυνατή η πρόσβαση στα Web services με την χρήση SSL. Επίσης, αυτό επιφέρει και αλλαγές και στα logs τα οποία αποθηκεύει ο Web Server. Πλέον υπάρχει η δυνατότητα της καταγραφής των διάφορων HTTP requests τα οποία γίνονται με χρήση SSL στο αρχείο ssl_request.log. Αυτό το αρχείο έρχεται για να συμπληρώσει τα αιτήματα του αρχείου access.log αρχείου που περιλαμβάνει μόνο τα HTTP αιτήματα.

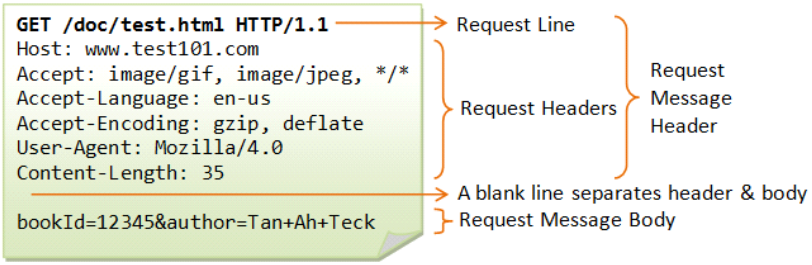
```
#<VirtualHost *:443>
#    DocumentRoot C:/xampp/htdocs/ptyxiaki
#    ServerName kardamanidischristos.com
#    SSLEngine on
#    SSLCertificateFile "conf/ssl.crt/server.crt"
#    SSLCertificateKeyFile "conf/ssl.key/server.key"
#</VirtualHost>
```

Σχήμα 2 Directive for SSL support

3.5 Δομή HTTP μηνύματος

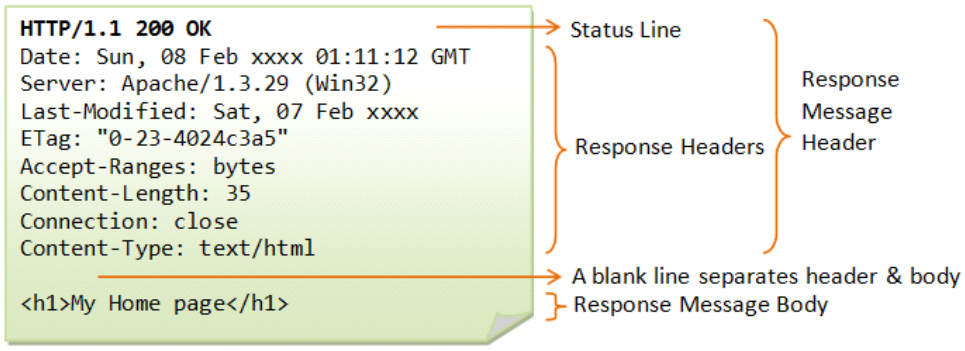
Κατά το RFC το HTTP έχει συγκεκριμένη δομή για τα μηνύματα του. Τα μηνύματα του μπορεί να είναι είτε response είτε request. Για αρχή θα αναλυθεί ένα παράδειγμα ενός HTTP request message. Ένα παράδειγμα HTTP αιτήματος φαίνεται στο Σχήμα 3 Δομή HTTP αιτήματος. Τα μηνύματα του HTTP είναι γραμμένα σε ASCII ώστε να μπορούν να διαβαστούν και από ανθρώπους. Η πρώτη γραμμή είναι η γραμμή του αιτήματος (request line). Επίσης σημαντικό σημείο είναι το ποια μέθοδος θα χρησιμοποιηθεί. Οι μέθοδοι που χρησιμοποιεί το HTTP είναι οι GET, POST,PUT, DELETE, PATCH, CONNECT, HEAD, OPTIONS, TRACE. Κάθε μια από αυτές έχει διαφορετική χρήση. Για παράδειγμα η GET χρησιμοποιείται για την λήψη ενός αντικειμένου από τον server, αντίθετα από την POST που χρησιμοποιείται για την αποστολή ενός αντικειμένου στον server. Αμέσως μετά προσδιορίζεται το URI στο οποίο θα γίνει το αίτημα. Το URI μπορεί να είναι κάποιο endpoint, είτε κάποιο URL Μετά από το URI συμπεριλαμβάνεται και η έκδοση του HTTP. Οι αμέσως επόμενες γραμμές είναι οι γραμμές κεφαλίδας (header lines). Σε αυτές τις γραμμές γίνεται ο προσδιορισμός του

server σαν κεφαλίδα του HTTP όπου προσθέτουμε σαν "host" τον server ο οποίος θα αποκριθεί στο αίτημα μας.



Σχήμα 3 Δομή HTTP αιτήματος

Μία άλλη κεφαλίδα η οποία έχει μεγάλη σημασία είναι το "x-api-key" header, μέσα στο οποίο μπορούμε να τοποθετήσουμε το API key για να προστατέψουμε το API από αιτήματα που δεν είναι από κάποια γνώριμη εφαρμογή όπως η Android που αναπτύσσεται στην πτυχιακή. Μαζί με αυτό σαν κεφαλίδα μπορούμε να χρησιμοποιήσουμε διάφορα άλλες κεφαλίδες για την επικοινωνία με τον server. Σημαντικό ρόλο παίζει η κεφαλίδα "content-type" καθώς προσδιορίζει το τι δεδομένα θα κουβαλάει το HTTP αίτημα στο σώμα του μηνύματος. Τελευταίο και πολύ σημαντικό κομμάτι του HTTP μηνύματος είναι το σώμα του μηνύματος του αιτήματος με το οποίο μπορούμε να στείλουμε δεδομένα στον server χωρίς να τα έχουμε ορατά όπως οι παράμετροι που κωδικοποιούνται στο URL. Σημαντικό είναι πως πολλές κεφαλίδες είναι ίδιες και στην απόκριση του HTTP όπως φαίνεται στο Σχήμα 4 Δομή HTTP απόκρισης.



Σχήμα 4 Δομή HTTP απόκρισης

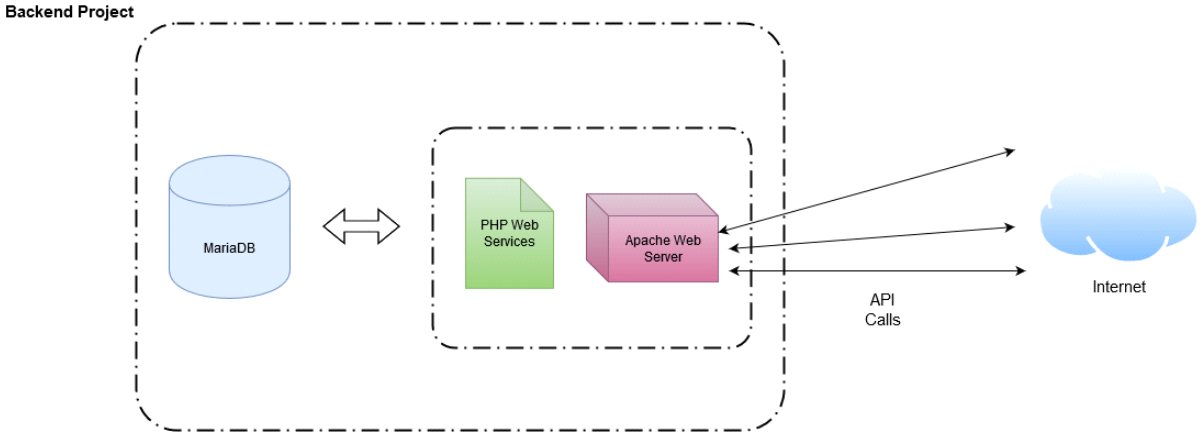
3.6 Web Services και PHP

Για την ανάπτυξη των Web Services, χρησιμοποιήθηκε η γλώσσα PHP. Η γλώσσα PHP είναι μια server side scripting γλώσσα η οποία κυρίως χρησιμοποιείται για την ανάπτυξη Web εφαρμογών, αλλά και σαν γλώσσα γενικού σκοπού. Γενικά για την διαχείριση των δεδομένων του ηλεκτρονικού καταστήματος, είναι αναγκαίο να μην υπάρχει απευθείας σύνδεση όλων των εφαρμογών client στην βάση δεδομένων όλου του καταστήματος. Για αυτόν τον λόγο η επικοινωνία των client side εφαρμογών γίνεται μέσω ενός API το οποίο κάνει διαχείριση όλων αυτών των συνδέσεων από την Android εφαρμογή από όλα τα κινητά στα οποία υπάρχει η εφαρμογή της πτυχιακής. Η ανάπτυξη του Web Service είναι ένα ενδιάμεσο στάδιο στο οποίο η PHP μπορεί να κάνει προσπέλαση στην βάση δεδομένων και να μπορεί να κάνει προσθήκες τροποποιήσεις και να παίρνει και να δίνει δεδομένα με

την MariaDB. Η χρησιμότητα του Web Service είναι ότι με την δημιουργία αυτού και σχηματίζοντας ολόκληρο το back end που αποτελείται από την MariaDB, το PHP Web service και τον Apache Web Server, μπορεί οποιουδήποτε τύπου client εφαρμογή να συνδεθεί με την χρήση του Web Service αυτού, ανεξάρτητα από τον τύπο της client side εφαρμογής. Στην περίπτωση της πτυχιακής μιλάμε για μια Android εφαρμογή, ενώ θα μπορούσε στο μέλλον να συνδεθεί μια άλλη εφαρμογή ανεξάρτητα από τον τύπο του λειτουργικού και της συσκευής.

3.7 Web Services και Αρχιτεκτονική

Για την επικοινωνία με την βάση δεδομένων και την διαχείριση από την Android εφαρμογή υπάρχει ανάγκη από ένα ενδιάμεσο στάδιο. Αυτό το ενδιάμεσο στάδιο είναι ένα Web Service δηλαδή μια υπηρεσία η οποία βασίζεται στο HTTP. Αυτή η υπηρεσία είναι διαθέσιμη όσο είναι διαθέσιμος και ο Web server, έτσι ώστε με ένα HTTP αίτημα να μπορεί να στείλει δεδομένα στο service αλλά και να λάβει. Το Web Service που είναι υλοποιημένο σε PHP θα μπορεί να επικοινωνεί στην συνέχεια με την MariaDB. Η αρχιτεκτονική του backend φαίνεται στο Σχήμα 5 Αρχιτεκτονική backend. Για να γίνει η επικοινωνία με την Android εφαρμογή όμως πρέπει να υπάρχουν και κάποια standards. Σε αυτό το σημείο υπάρχει ανάγκη δημιουργίας ενός API.



Σχήμα 5 Αρχιτεκτονική backend

Γενικά ως API είναι κάποιοι μέθοδοι οι οποίες είναι από πριν καθορισμένες ώστε να μπορεί ένα software component να επικοινωνεί με ένα άλλο software component. Στην προκειμένη περίπτωση το ένα software component είναι το Web service και το άλλο είναι η Android εφαρμογή. Μια πολύ κοινή αρχιτεκτονική σχεδίασης είναι το REST API. Στην περίπτωση χρησιμοποιήθηκαν πολλά από τα standards του REST API. Αν δεν ακολουθείται κάποιο από αυτά τα standards τότε θεωρείται ως ένα Web API το οποίο όμως παραμένει λειτουργικό και χρήσιμο για την επικοινωνία του client με τον server. Τα αρχικά του REST σημαίνουν Representational State Transfer είναι ένας τρόπος να πετυχαίνουμε επικοινωνία διάφορων συστημάτων μέσω του διαδικτύου.

Γενικά στα Rest API υπάρχει η έννοια του πόρου (resource).Κάθε πόρος αντιστοιχίζεται με ένα URI δηλαδή ο client μέσω αυτού του URI έχει πρόσβαση σε αυτόν. Το κάθε URI έχει μια συγκεκριμένη μορφή για να δείχνει πληροφορίες για το resource με το οποίο αντιστοιχίζεται. Για αυτό και υπάρχουν συγκεκριμένοι κανόνες μέσα από τους οποίους αυτά τα URIs αποκτούν μια συγκεκριμένη μορφή. Η γενική μορφή ενός URI είναι η εξής protocol://serviceName/Resource/. Σε αυτό το URI μπορούμε να περάσουμε και διάφορες πληροφορίες σαν query parameter και το URI πλέον είναι της μορφής “protocol://serviceName/Resource?parameter=value”. Όσον αφορά τις όποια ενέργειες μπορούν να γίνουν πάνω σε αυτό το URI, το ίδιο το URI δεν θα πρέπει να περιλαμβάνει πληροφορίες για την

ενέργεια την οποία θα εφαρμοστεί σε αυτό. Το τι μπορεί να γίνει σε ένα Resource, θα πρέπει να περιγράφεται από ένα από τα HTTP methods. Για παράδειγμα αν σε ένα Resource εφαρμόζεται η μέθοδος GET, θα πρέπει το web service να επιστρέφει ότι μπορεί να διαβάσει για αυτό το resource. Για παράδειγμα, αν στο endpoint "/ptyxiaki/index.php/api/v1/Categories" εφαρμοστεί η μέθοδος "GET", τότε το Web service θα πάει να διαβάσει το "Categories" πόρο και να μου επιστρέψει στον client όλο αυτό το resource το οποίο περιλαμβάνει όλες τις κατηγορίες των προϊόντων του e-shop.

3.8 Υλοποίηση κώδικα

3.8.1 Σύνδεση PHP με βάση δεδομένων και MySQLi connector

Για να γίνει σύνδεση του PHP κώδικα με την βάση δεδομένων, υπάρχει ανάγκη για την ύπαρξη μιας βιβλιοθήκης για να μπορούμε μέσα από τον κώδικα να εκτελούμε ερωτήματα για να διαβάζουμε ή να τροποποιούμε την βάση δεδομένων. Σε αυτό το σημείο είναι ανάγκη να αναφερθεί πως η MySQL έχει μια μεγάλη χρήση στην ανάπτυξη των Web εφαρμογών και πως πλήθος εφαρμογών ήδη χρησιμοποιούσαν αυτήν την βάση δεδομένων. Από την έκδοση MySQL 5.5 και μετά δημιουργήθηκε ένας κλάδος της, η αντίστοιχη έκδοση της MariaDB. Για αυτόν τον λόγο και παρουσιάζουν πολύ λίγες διαφορές. Σημαντικό είναι πως τα αρχεία δεδομένων(data files) και τα table definition files της MariaDB είναι συμβατά με την αντίστοιχη έκδοση της MySQL. Για αυτόν τον λόγο και το να γίνει upgrade από μια MySQL σε μια MariaDB είναι εφικτό τηρώντας την διαφορά των εκδόσεων του κάθε DBMS. Ως επακόλουθο αυτών των ομοιοτήτων και της ήδη υπάρχουσας χρήσης της MySQL, η MariaDB μπορεί να χρησιμοποιεί τα ίδια πρωτόκολλα με την MySQL. Αυτό περιλαμβάνει τα διάφορα ports και sockets που χρησιμοποιούν.

Τελευταίο αλλά και πολύ σημαντικό είναι πως όλοι οι connectors της MySQL μπορούν να λειτουργήσουν κανονικά και με την MariaDB. Σύμφωνα με την τεκμηρίωση της MariaDB ώς "connector" λογίζεται κάθε λογισμικό το οποίο μπορεί να συνδεθεί με τον MySQL DBMS. Ο connector ο οποίος χρησιμοποιείται στην πτυχιακή είναι η βιβλιοθήκη mysqli και με αυτήν ο κώδικάς θα μπορεί να συνδεθεί και να μέσω αυτής ο χρήστης να εκτελεί διάφορες διαδικασίες στην βάση δεδομένων. Η βιβλιοθήκη αυτή έχει διάφορα πλεονεκτήματα, για να χρησιμοποιήσει ο προγραμματιστής. Αρχικά είναι αντικειμενοστραφής, οπότε μπορεί να έχει ενθυλάκωση των διάφορων δεδομένων. Επίσης σημαντικό είναι ότι μπορεί να εκτελεί πολλαπλές δηλώσεις στην βάση καθώς και να κάνει και χρήση των prepared statements. Επίσης άξιο αναφοράς είναι πως υπάρχει και η δυνατότητα χρήσης συναλλαγών (transactions) πράγμα το οποίο μπορεί να χρησιμεύσει στην περίπτωση που πολλαπλοί χρήστες θέλουν να κάνουν επεξεργασία στα ίδια δεδομένα

3.8.2 Αρχεία index.php και dbconnect.php

Αυτά τα οποία πρέπει το Web service να αντιληφθεί για να κάνει τις όποιες ενέργειες στο back end είναι κάποια βασικά συστατικά. Το πρώτο είναι το URI το οποίο αντιστοιχεί σε ένα Resource (π.χ. Products) . Το δεύτερο είναι η HTTP μέθοδος που θα εφαρμοστεί σε αυτό. Τρίτο και είναι οι διάφορες κεφαλίδες οι οποίες στέλνονται με το HTTP μήνυμα. Τέταρτο και τελευταίο είναι το σώμα του μηνύματος HTTP που περιλαμβάνει τον κύριο όγκο των δεδομένων. Αυτά τα σημεία θα αναλυθούν παρακάτω. Βασικό σημείο για την έναρξη της εκτέλεσης του PHP κώδικα είναι να γίνει ένα HTTP αίτημα στο αντίστοιχο αρχείο PHP.

Στον Apache Web server το αρχείο το οποίο είναι προκαθορισμένο για εκτέλεση είναι το αρχείο index.php. Για αυτόν τον λόγο το σημείο αρχής της εκτέλεσης είναι αυτό το αρχείο στα Web services. Αμέσως μετά για να μπορέσει να γίνει η οποιαδήποτε πρόσβαση στην βάση δεδομένων πρέπει να υπάρχει ένας τρόπος να συνδέεται το Web service στην MariaDB. Αυτό συμβαίνει με την χρήση του αρχείου dbconnect.php όπως φαίνεται στο Σχήμα 6 Κώδικας σύνδεσης στην βάση δεδομένων. Το αρχείο αυτό γίνεται require από το αρχείο index.php και ουσιαστικά στο αρχείο index.php καλείται ο κώδικας του dbconnect.php. Το αρχείο αυτό με χρήση της βιβλιοθήκης mysqli επιτυγχάνει μια σύνδεση στο DBMS ώστε να μπορεί να στην συνέχεια της εκτέλεσης του PHP script να κάνει επεμβάσεις στην βάση δεδομένων. Στον PHP κώδικα όπως φαίνεται είναι καθορισμένα στον κώδικα τα στοιχεία της σύνδεσης στην βάση δεδομένων ώστε να μπορέσει η mysqli να εκτελέσει ερωτήματα στην συνέχεια. Όπως φαίνεται φτιάχνουμε ένα mysqli αντικείμενο το οποίο έχει 4 παραμέτρους για την σύνδεση. Το πρώτο είναι το “host” του server στο οποίο βρίσκεται η MariaDB. Αμέσως μετά χρειάζεται άλλες τρεις παραμέτρους, το “username” με το οποίο θα συνδεθείς στην βάση δεδομένων, το “password” με το οποίο θα συνδεθεί η εφαρμογή, καθώς και την βάση δεδομένων στην οποία θα συνδεθεί.

dbconnect.php

```php
<?php

$user='root';
$pass='';
$host='localhost';
$db = 'eshop';

$mysqli = new mysqli($host, $user, $pass, $db);
if ($mysqli->connect_errno) {
    echo "Failed to connect to MySQL: (" .
    $mysqli->connect_errno . ") " . $mysqli->connect_error;
}

?>
```

Σχήμα 6 Κώδικας σύνδεσης στην βάση δεδομένων

Αμέσως μετά στο αρχείο php εφόσον έχει πετύχει η σύνδεση στην βάση δεδομένων πρέπει να κάνει αποκωδικοποίηση το Web service ποιο είναι το URI το οποίο αιτήθηκε η εφαρμογή πελάτη, με ποια μέθοδο του HTTP έγινε αυτή η κλήση, καθώς και να γίνει λήψη όλων των κεφαλίδων που έχουν σταλεί μέσω του HTTP και των δεδομένων που έχει η στείλει η εφαρμογή πελάτη στο Web service στο σώμα του HTTP μηνύματος. Για αρχή όπως φαίνεται στον κώδικα Σχήμα 7. με την “file_get_contents(‘php://input’)” μπορώ να διαβάσω το περιεχόμενο του αρχείου “php://input”. Το “php://input” είναι ένα read only stream το οποίο μας επιτρέπει να διαβάσουμε τα περιεχόμενα ενός σώματος ενός αιτήματος HTTP σε μορφή raw data. Αυτό είναι το κομβικό σημείο μεταφοράς όλων των δεδομένων μέσα από το σώμα του HTTP μηνύματος. Μέσα από το read only stream μπορούμε να διαβάσουμε ότι στέλνει η εφαρμογή πελάτη στον server. Στην περίπτωση της πτυχιακής, το περιεχόμενο που λαμβάνει ο server είναι πάντα σε μορφή JSON οπότε όπως φαίνεται στην φωτογραφία το περιεχόμενο το κάνουμε αποκωδικοποίηση. Η αποκωδικοποίηση γίνεται με την μετατροπή της μορφής JSON σε ένα PHP associative array. Associative array είναι ένας πίνακας ο οποίος σαν δείκτες των κελιών του έχει strings σαν δείκτη για να έχει πρόσβαση στα στοιχεία των κελιών του. Αυτό βοηθάει γιατί ουσιαστικά όπως και σε ένα JSON format έχουμε ζεύγη κλειδιών και τιμών, μπορούμε να το έχουμε σε ένα associative array για να έχουμε πρόσβαση σε κάθε τιμή του JSON.

Το δεύτερο το οποίο θα πρέπει να γίνει αντιληπτό είναι η μέθοδος, την οποία παίρνουμε από το κελί “REQUEST_METHOD” του global πίνακα $_SERVER και το αποθηκεύουμε σε μια μεταβλητή

“method” για να έχουν πρόσβαση όλα τα resources σε αυτό. Τρίτο σημείο το οποίο αναφέρθηκε παραπάνω είναι οι κεφαλίδες οι οποίες στέλνει η εφαρμογή πελάτη. Η πιο σημαντική κεφαλίδα είναι το “x-api-key” του HTTP πρωτοκόλλου. Αυτή η κεφαλίδα πρέπει να έχει μια συγκεκριμένη τιμή για να μπορεί να αντιληφθεί το Web service ότι η εφαρμογή πελάτη είναι έμπιστη. Αυτό έχει να κάνει με την ασφάλεια του API, καθώς δεν πρέπει να γίνονται κλήσεις από μη εξουσιοδοτημένες εφαρμογές. Με την μέθοδο “getallheaders()[‘x-api-key’]” μπορώ να έχω πρόσβαση στο κλειδί API το οποίο έστειλε η εφαρμογή πελάτη και να το ελέγξω αν είναι σωστό. Στην περίπτωση μας υπάρχει ένα προκαθορισμένο κλειδί API “1234567890”. Η δεύτερη κεφαλίδα η οποίο θα αναφερθεί παρακάτω είναι η “Authorization” μέσα από την οποία θα είναι το δεύτερο επίπεδο ασφάλειας το οποίο θα ελέγχει το token του χρήστη το οποίο είναι θα δείχνει και πως ποιος συνδεδεμένος χρήστης του ηλεκτρονικού καταστήματος έκανε αυτήν την ενέργεια. Σε αυτό το σημείο σημαντικό είναι να ελέγξουμε τι στέλνεται στο API μέσω του σώματος του HTTP μηνύματος. Για αυτόν τον λόγο και υπάρχει η μέθοδος validateInput() η οποία έχει οριστεί στο toolbox.php και θα κάνει επαλήθευση τον JSON πίνακα που έχει λάβει το Web service από το HTTP μήνυμα.

```php
require_once 'dbconnect.php';
require_once 'toolbox.php';

$request_body = file_get_contents('php://input'); //php raw stream from http request body
$json = json_decode($request_body,true);
$json = validateInput($json);
$method = $_SERVER['REQUEST_METHOD'];

if (!isset(getallheaders()['x-api-key']) || getallheaders()['x-api-key'] != "1234567890") {
    header("HTTP/1.1 401 Unauthorized");
    exit;
}

if (!isset($_SERVER['PATH_INFO'])) {
    header('HTTP/1.1 500 Internal Server Error');
    exit;
}
```

Σχήμα 7 Εξαγωγή headers, request body, http method

Αμέσως μετά πρέπει να γίνει αποκωδικοποίηση το URI για το οποίο έγινε το αίτημα. Για παράδειγμα έστω υπάρχει ένα αίτημα για το endpoint “/index.php/api/v1/Products”, θα πρέπει το service να το ερμηνεύσει κατάλληλα για να εξάγει τις διάφορες πληροφορίες που περιλαμβάνει. Στο αναφερθέν παράδειγμα, μπορούμε να δούμε πως δούμε πως ο client θέλει να κάνει αίτημα στο API, στην συνέχεια αναφέρει πως το αίτημα θέλει να γίνει στο API με έκδοση βάσης δεδομένων “v1” και θέλει να κάνει εφαρμόσει μια μέθοδο στο Resource “Products”. Η πληροφορία για όλα αυτά τα πεδία του URI βρίσκεται στο κελί ‘PATH_INFO’ του πίνακα $_SERVER. Για αυτόν τον λόγο αν δεν υπάρχει σημαίνει πως υπάρχει κάποιο πρόβλημα. Για αυτόν τον λόγο και θα πρέπει το αίτημα να τερματιστεί και να επιστρέψει το ανάλογο μήνυμα. Με την χρήση της μεθόδου header() και της exit μπορούμε να διακόψουμε την εκτέλεση του PHP script κατά την διάρκεια της επεξεργασίας του API αιτήματος για να επιστρέψουμε στον client το όποιο μήνυμα σφάλματος. Για παράδειγμα κατά τον έλεγχο του ‘x-api-key’ αν διαπιστωθεί λάθος, σημαίνει ότι δεν είναι διαπιστευμένη εφαρμογή και πρέπει να επιστρέψουμε “401 Unauthorized”. Στην περίπτωση που δεν υπάρχει PATH για URI ο server επιστρέφει “500 Internal Server Error”. Αυτή η πληροφορία του URI στο $_SERVER[‘PATH_INFO’] όμως πρέπει να ελεγχθεί και να είναι έτοιμη για περαιτέρω ελέγχους για την διευθυνσιοδότηση του resource. Για αρχή το σπάμε σε κομμάτια με διαχωριστή τον χαρακτήρα “/” και το ελέγχουμε το μέγεθος του URI. Τα υποχρεωτικά πεδία στο URI είναι τα service/dbversion/resource τα οποία είναι 3. Αν είναι μικρότερος το URI δεν είναι έγκυρο. Από εκεί και κάτω ότι υπάρχει περνάει στο ανάλογο resource σαν κωδικοποιημένα δεδομένα του URL. Για αυτό ελέγχουμε αν “sizeof(request)”, δηλαδή το μέγεθος του URI είναι μεγαλύτερο απο 3, κόβουμε ένα URI που είναι της μορφής “service/dbversion/resource/data1/data2/data3” από το resource, τα τοποθετούμε σε έναν πίνακα “$urlEncodedData” και μετά για να είναι προσβάσιμα στο resource. Συνοψίζοντας, μετά από όλην αυτήν την επεξεργασία έχουμε στα χέρια μας έναν JSON πίνακα με όλο το σώμα του μηνύματος HTTP, έχουμε στην μεταβλητή method την μέθοδο του HTTP αιτήματος.

Επίσης το Web service έχει το URI σπασμένο σε 2 κομμάτια, το request που περιλαμβάνει {service,dbversion,resource} και urlEncodedData{data1,data2,data3}. Πλέον σε αυτό το σημείο πρέπει να γίνει η διευθυνσιοδότηση του URI και λήψη των δεδομένων από το ανάλογο Resource. Το πρώτο στάδιο που ελέγχεται είναι αν το service που θέλει να γίνει το αίτημα είναι το API. Αν είναι το API θα κάνει require το αρχείο DBVersions.php το οποίο θα συνεχίσει τους ελέγχους στο αμέσως επόμενο κομμάτι του URI το οποίο είναι αν το αίτημα γίνεται σε διαθέσιμη έκδοση της βάσης δεδομένων.

3.8.3 DBVersions.php και Resources.php

Μετά το index.php τα αμέσως επόμενα δυο αρχεία τα οποία θα ελέγξουν την δομή του URI για να γίνει η διευθυνσιοδότηση του πόρου. Σε ένα URI της μορφής "/index.php/api/v1/Products" το αμέσως επόμενο κομμάτι του URI είναι η έκδοση της βάσης δεδομένων στην οποία θα γίνει το αίτημα. Ο πίνακας request στο πεδίο "dbversion" υπάρχει η έκδοση της βάσης δεδομένων που θέλει η εφαρμογή πελάτης. Στη περίπτωση της πτυχιακής, υπάρχει μόνο μια έκδοση βάσης με όνομα "v1" όπως περιγράφεται στο Σχήμα 8. Στην περίπτωση που είναι έγκυρη η έκδοση της βάσης δεδομένων, θα κάνει require_once το αρχείο Resources,php. Η εντολή require_once όταν καλεί ένα αρχείο, καλείται στην θέση της και εκτελείται ο κώδικας του αρχείου.

DBVersions.php

```
1   <?php
2
3   if($request['dbversion'] == "v1") {
4       require_once "Resources.php";
5   }
6   else {
7       print "Database version not found!";
8   }
9
10  ?>
```

Σχήμα 8 Αρχείο ελέγχου έκδοσης βάσης

Το αμέσως επόμενο αρχείο για την διευθυνσιοδότηση του Resource, μέσα στο Web service, είναι το αρχείο Resources.php. Κάθε resource το οποίο παρέχει το Web service για λήψη και αποστολή δεδομένων βρίσκεται προκαθορισμένο σε αυτό το αρχείο Resources.php όπως φαίνεται στο Σχήμα 9. Ο πίνακας resources περιλαμβάνει όλα αυτά και σε αυτόν τον πίνακα θα πρέπει να γίνει έλεγχος αν υπάρχει το Resource που ζητάει το API αίτημα. Με την συνάρτηση in_array() μπορούμε να ελέγξουμε αν μια συμβολοσειρά υπάρχει μέσα σε έναν πίνακα, αυτή η συμβολοσειρά η οποία είναι το $request['resource'] δηλαδή το resource για το οποίο έγινε το request μέσα στον πίνακα με διαθέσιμα resources. Αν υπάρχει, θα πάει και θα κάνει require το αντίστοιχο αρχείο php με το όνομα του resource που υπάρχει στο URI. Όλα τα αρχεία με τα resources υπάρχουν στον υποφάκελλο Resources που είναι το προκαθορισμένο μέρος για την δημιουργία μελλοντικών resources.

Resources.php

```php
<?php

$resources = array("Users","Products",
                   "Orders","Carts",
                   "Categories","Administrators",
                   "Customers","CustomSQL",
                   "ServerDetails","Employees");

if (in_array($request['resource'], $resources)) {
    require_once "./Resources/" . $request['resource'] . ".php";
}
else
{
    header('HTTP/1.1 404 Not found');
    exit;
}
?>
```

Σχήμα 9 Αρχείο ελέγχου διαθέσιμου πόρου

3.8.4 CustomSQL Resource

Αυτό το συγκεκριμένο Resource αποτελεί ένα σημείο πρόσβασης για τους διαχειριστές για να έχουνε πρόσβαση στην βάση δεδομένων μέσω του Web API. Με ένα http αίτημα στο συγκεκριμένο endpoint, ο διαχειριστής στέλνοντας το κατάλληλο ερώτημα στο endpoint, να πάρει σε μορφή JSON ότι δεδομένα θέλει από τον server. Αυτό το resource απαιτεί μέσω του HTTP αίτημα να στείλει η εφαρμογή πελάτης ένα ερώτημα στον server. Το ερώτημα θα πρέπει να έρχεται στον server σε JSON της μορφής {“query” : “SELECT...”}.

Για αρχή όπως φαίνεται στο Σχήμα 10 ελέγχεται αν η μέθοδος που χρησιμοποιεί το HTTP αίτημα είναι η POST και γίνεται ο έλεγχος αν πέρασε σαν παράμετρος το ερώτημα στο JSON. Στην πορεία με την χρήση του $mysqli→query() δηλαδή την μέθοδο query() εκτελώ το ερώτημα περνώντας της την συμβολοσειρά που το περιέχει. Αν έχει εκτελεστεί σωστά αποθηκεύεται το αποτέλεσμα στην μεταβλητή result όπου διαβάζουμε με την μέθοδο fetch_assoc() κάθε γραμμή του αποτελέσματος και την τοποθετούμε στην μεταβλητή $r. Το αποτέλεσμα είναι ο πίνακας $rows να περιέχει όλες τις γραμμές. Τέλος κάνουμε json_encode το αποτέλεσμα και το κάνουμε print για να επιστραφεί στην εφαρμογή πελάτη. Ιδιαίτερη αναφορά πρέπει να γίνει στην παράμετρο “JSON_PRETTY_PRINT" η οποία επιστρέφει το JSON τυπωμένο με πολύ καλύτερη μορφή στον χρήστη. Αυτό μπορεί να μην κάνει διαφορά στην απόδοση του προγράμματος, αλλά έχει διαφορά στον προγραμματιστή όταν κάνει έλεγχο της λειτουργικότητας του API.

```php
if ($method == "POST") {
    if (isset($json['query'])) {

        $query = $json['query'];
        $rows = array();

        if ($result = $mysqli->query($query)) {
            while ($r = $result->fetch_assoc()) {
                $rows[] = $r;
            }
        }
        print json_encode($rows, JSON_PRETTY_PRINT);
    }
}
```

3.8.5 ServerDetails Resource

Το πρώτο Resource το θα αναλυθεί είναι το ServerDetails. Σε αυτό το Resource εμπεριέχονται κάποιες βασικές πληροφορίες για το back end της εφαρμογής. Τέτοιες πληροφορίες είναι η έκδοση της mysql, η IP του server, η έκδοση της PHP αλλά και η έκδοση του API. Αρχικά με την χρήση της μεθόδου phpversion() εξάγουμε την έκδοση της PHP, στην συνέχεια από τον associative πίνακα $_SERVER παίρνουμε την IP του Web Server με το κλειδί 'SERVER_ADDR' ($_SERVER[''SERVER_ADDR' ']) και τέλος από το $mysqli-> server_info δηλαδή το πεδίο server_info της $mysqli μεταβλητής, παίρνουμε τις πληροφορίες για την έκδοση της MariaDB που χρησιμοποιούμε. Όλα αυτά θα τα κάνουμε κωδικοποίηση σε μορφή JSON και θα τα στείλουμε στην εφαρμογή πελάτη. Με την χρήση της εντολής echo κάνουμε εκτύπωση ότι θέλουμε, με την χρήση της json_encode μεθόδου κάνουμε κωδικοποίηση έναν associative πίνακα σε JSON μορφή. Τέλος αυτό το οποίο επιστρέφει η μέθοδος το κάνουνε echo για να σταλεί το JSON από τον Web server στον client. Οπότε, αν γίνει ένα HTTP αίτημα στο endpoint "api/v1/ServerDetails" θα επιστρέψει σε JSON μορφή τις πληροφορίες που αναφέρθηκαν παραπάνω.

3.8.6 Employees Resource

Το αμέσως επόμενο resource το οποίο υπάρχει στο Web service είναι το "Employees". Αυτό το Resource περιλαμβάνει ένα subresource το οποίο είναι το "Sessions". Το "Sessions" subresource είναι αυτό το οποίο απεικονίζει τις συνεδρίες των υπαλλήλων. Πάνω σε αυτό επιτρέπονται δύο είδη HTTP μεθόδων, η POST και η DELETE όπως φαίνεται στο Σχήμα 10. Το συγκεκριμένο subresource επιστρέφει διαφορετικά δεδομένα ανά την μέθοδο που εφαρμόζεται.

Όταν κάνουμε POST πρέπει να περάσουμε σαν παράμετρο στο JSON που στέλνει η εφαρμογή πελάτης μέσω του σώματος του HTTP μηνύματος, δύο κλειδιά-τιμές, το ένα είναι το "username", που περιλαμβάνει το όνομα χρήστη και το "password" το οποίο περιλαμβάνει το συνθηματικό του χρήστη. Αν πετύχει η σύνδεση, θα επιστρέψει ένα JSON με ένα κλειδί "userTokken" και η τιμή του θα περιλαμβάνει το token το οποίο θα συμπεριλαμβάνεται στο αίτημα της Android εφαρμογής, αντί να μεταφέρεται κάθε φορά το όνομα χρήστη και ο κωδικός. Αλλιώς επιστρέφει JSON της μορφής { "error" : "Invalid credentials"}. Για αρχή ελέγχει αν υπάρχει ένα subresource περασμένο στο URI που έγινε αίτηση και στην πορεία ελέγχεται για το αν είναι μέσα στην λίστα των διαθέσιμων subresources αλλά και αν υπάρχει η μέθοδος στις διαθέσιμες μεθόδους. Στην συνέχεια από τον πίνακα JSON παίρνουμε τα username και password για να φτιάξουμε το SQL ερώτημα. Αμέσως μετά κάνουμε prepare το SQL statement καλώντας την $prepare() μέθοδο του $mysqli αντικειμένου για να φτιάξουμε το statement το οποίο θα κάνει ένα SELECT για να ελέγξει αν υπάρχουν χρήστες με αυτό το username και το password της σύνδεσης. Στην συνέχεια με την μέθοδο και ορίσματα bind param("ss", $username, $password) κάνουμε bind τα ορίσματα στο SQL statement και τέλος με την μέθοδο execute() του stmt (statement) εκτελούμε το query.

Τέλος με την μέθοδο get_result() παίρνουμε το αποτέλεσμα από stmt αντικείμενο και από το result αντικείμενο παίρνουμε το num_rows που δείχνει πόσες γραμμές υπάρχουν. Σε αυτήν την φάση το αντικείμενο περιλαμβάνει μια γραμμή με τα στοιχεία του χρήστη που έκανε ταυτοποίηση. Τέλος αν όντως έχει βρεθεί μόνο ένας χρήστης θα πάμε και θα κάνουμε insert στον πίνακα Sessions τα πεδία "STATUS", "USER_ID," "SESSION_START", "USERTOKKEN", όπου το πρώτο θα είναι "CONNECTED", το "USER_ID" του χρήστη θα είναι του χρήστη που βρήκε παραπάνω, η αρχή του session θα είναι η τωρινή ημερομηνία που μας δίνεται από την συνάρτηση date() και το token το οποίο θα επιστραφεί θα είναι μια md5 κωδικοποιημένη συμβολοσειρά η οποίο θα δίνεται από το

πέρασμα της ημερομηνίας σαν παράμετρο από την md5() συνάρτηση. Τέλος, με την χρήση της json_encode() συνάρτησης κάνουμε encode το token και το επιστρέφουμε στην εφαρμογή Android.

Η αντίστροφη διαδικασία γίνεται στην HTTP μέθοδο DELETE κατά την οποία γίνεται UPDATE ο πίνακας SESSION στην η γραμμή που προστέθηκε στην POST. Σε αυτήν την περίπτωση περνάμε στο Web service σαν παράμετρο το usertokken και εκεί που ταυτίζεται το tokken θα πάει και θα κλείσει το session και δεν θα μπορεί το token να χρησιμοποιείται για HTTP αίτημα. Αν βρει αυτήν την γραμμή επιστρέφει HTTP 204 αλλιώς επιστρέφει σε JSON ένα error μήνυμα “Session could not be closed”.

```php
<?php
$subresource = "undefined";
if (isset($urlEncodedData[0])) {
    $subresource = $urlEncodedData[0];
}

if ($subresource == "Sessions") {
    if ($method == "POST") {
        $username = $json['username'];
        $password = $json['password'];
        $sql = "SELECT * FROM Users WHERE USERNAME = ? AND PASSWORD = ?";
        if (!($stmt = $mysqli->prepare($sql))) {
            print "Prepared failed:(" . $mysqli->errno . ") " . $mysqli->error;
        }
        if (!$stmt->bind_param("ss", $username, $password)) {
            print "Binding parameters failed:(" . $stmt->errno . ")" . $stmt->error;
        }
        if (!$stmt->execute()) {
            print "Execute failed:(" . $stmt->errno . ")" . $stmt->error;
        }
        $result = $stmt->get_result();
        $matchedUsers = $result->num_rows;
        $row = $result->fetch_assoc();
        $userid = $row['USERID'];
        $stmt->close();
        if ($matchedUsers == 1) {
            $sql = "INSERT INTO Sessions(STATUS, USER_ID, SESSION_START, USERT

            if (!($stmt = $mysqli->prepare($sql))) {
                print "Prepared failed:(" . $mysqli->errno . ") " . $mysqli->error;
            }
            $status="CONNECTED";
            $date = date("Y-m-d H:i:s");
            $tokken = md5(date("Y-m-d H:i:s"));

            if (!($stmt->bind_param("siss", $status, $userid, $date, $tokken))) {
                print "Binding parameters failed:(" . $stmt->errno . ")" . $stmt->error;
            }
            if (!$stmt->execute()) {
                print "Execute failed:(" . $stmt->errno . ")" . $stmt->error;
            }
            echo json_encode(array('userTokken' => "$tokken"));
            $stmt->close();
        }
        else if ($matchedUsers == 0) {
            echo json_encode(array('error' => "Invalid credentials"));
        }
    }
```

Σχήμα 10 Sessions Subresource POST

3.8.7 Categories Resource

Το επόμενο Resource το οποίο θα αναλυθεί είναι το Categories Resource. Αυτό το Resource απεικονίζει όλες τις κατηγορίες των προϊόντων που υπάρχουν στο ηλεκτρονικό κατάστημα. Σε αυτό το Resource επιτρέπονται οι μέθοδοι του HTTP GET,POST, DELETE οι οποίες χρησιμεύουν για όταν θέλουμε να κάνουμε POST δηλαδή να προσθέσουμε μια νέα κατηγορία και GET για όταν θέλουμε να εξάγουμε πληροφορίες για κάποιες κατηγορίες. Για αρχή ελέγχουμε την τιμή της μεταβλητής method για να δούμε ποιά μέθοδο χρησιμοποιεί ο client το request του.

Αν είναι GET θα επιστρέψουμε όλες τις κατηγορίες που υπάρχουν και βρίσκονται στον πίνακα της βάσης “categories” από τον οποίο θα κάνουμε SELECT όλες τις γραμμές όπως φαίνεται στο Σχήμα 11. Για την εκτέλεση του ερωτήματος χρησιμοποιείται η μέθοδος mysqli_query(), όπου της περνάμε σαν παράμετρο το $mysqli αντικείμενο το οποίο αρχικοποιήσαμε στο αρχείο dbconnect. Η μεταβλητή result είναι false αν δεν εκτελεστεί το ερώτημα στην βάση δεδομένων. Στην περίπτωση που εκτελεστεί με επιτυχία με την μέθοδο fetch_assoc() του αντικειμένου result παίρνουμε την κάθε επόμενη γραμμή του αποτελέσματος. Κάθε γραμμή φτιάχνει έναν πίνακα και προστίθεται στο return_data το οποίο θα είναι ένας πίνακας από πίνακες. Με την μεταβλητή $row[“collumnname”] μπορούμε να πάρουμε από την κάθε γραμμή την τιμή για την στήλη που προσδιορίζουμε. Τέλος αυτόν τον πίνακα από πίνακες τον κάνουμε json_encode και το επιστρέφουμε στον client ενώ ταυτόχρονα του στέλνουμε και κωδικό επιτυχίας “200 OK” που κατά τις συμβάσεις του HTTP σημαίνει πως όλα πήγαν καλά. Αυτό συμβαίνει με την μέθοδο header(‘HTTP/1.1 200 OK’) με την οποία προσθέτουμε αυτό το header στο HTTP response.

Η άλλη μέθοδος που εφαρμόζεται στο συγκεκριμένο Resource είναι η POST. Η μέθοδος αυτή περιμένει από τον client να στείλει στο web service το όνομα της κατηγορίας. Οπότε θα πρέπει ο πίνακας json να περιέχει τιμή στο κλειδί “categoryname”. Οπότε με την μέθοδο isset() ελέγχουμε αν έχει η τιμή το $json[‘categoryname’], αλλιώς θα επιστρέψει το web service μήνυμα σφάλματος. Το μήνυμα σφάλματος είναι το http code “400 Bad Request” το οποίο δείχνει ότι το request του client δεν είναι σωστό και θα κάνει exit το script. Αμέσως μετά εφόσον έχει ελεγχθεί το input θα σχηματίσουμε το insert ερώτημα “INSERT INTO CATEGORIES(CATEGORYNAME) VALUES(?);”, όπου θα κάνουμε prepare αυτό το ερώτημα και θα κάνουμε bind στην θέση του “?” το όνομα κατηγορίας που έχουμε σαν input. Τέλος το κάνουμε execute αυτό το ερώτημα και με την χρήση της json_encode επιστρέφουμε ένα json με κλειδί ‘categoryid’ και σαν τιμή το κλειδί της γραμμής που προστέθηκε. Αυτό χρησιμεύει γιατί έτσι μπορεί ο client να διαχειριστεί στο μέλλον αυτήν την κατηγορία. Αν η πρόσθεση στην βάση δεδομένων γίνει με επιτυχία τότε επιστρέφουμε με την μέθοδο header “HTTP/1.1 201 Created”, αλλιώς αν δεν γίνει σωστά η εκτέλεση επιστρέφουμε “HTTP/1.1 500 Internal Server Error” το οποίο δείχνει πως το web service είχε σφάλμα εκτελώντας το request.

Στην περίπτωση που η HTTP μέθοδος που χρησιμοποιήθηκε είναι η PUT απλά επιστρέφουμε κωδικό “HTTP/1.1 405 Method Not allowed”. Η τελευταία μέθοδος είναι η DELETE που ελέγχει για αρχή πόσα πεδία έχει στείλει ο client σαν JSON στο web service. Ελέγχει αν έχει δοθεί ένα “categoryname” και στην πορεία θα γίνει εκτέλεση το query “DELETE FROM CATEGORIES = ?” το οποίο θα γίνει prepare, θα γίνει bind το “categoryname” που έδωσε ο client και θα γίνει και εκτέλεση. Αν εκτελεστεί σωστά επιστρέφεται το “HTTP/1.1 204 No Content” το οποίο δείχνει ότι έχει εκτελεστεί εφόσον ο κωδικός είναι της μορφής “2XX” αλλά και ότι δεν επιστρέφει κάποιο περιεχόμενο. Στην περίπτωση που δεν εκτελεστεί σωστά το ερώτημα απλά το web service θα επιστρέψει “HTTP/1.1 500 Internal Server Error”. Αν δεν έχει προσδιοριστεί σαν παράμετρος το όνομα της κατηγορίας θα εκτελέσει ένα DELETE που διαγράφει όλες τις κατηγορίες. Τέλος, αν η μέθοδος που έχει δοθεί στο client request δεν είναι μια από τις μεθόδους που αναλύθηκαν παραπάνω, θα επιστρέψει το web service μήνυμα “HTTP/1.1 405 Method Not allowed” καθώς δεν επιτρέπεται άλλη μέθοδος για αυτό το Resource.

```php
<?php
if ($method == "GET") {
    $result = mysqli_query($mysqli, "SELECT * FROM  categories");
    if($result) {
        $return_data = array();
        while ($row = $result->fetch_assoc()) {
            $return_data[] = array(
                "categoryid" => $row['CATEGORYID'],
                "categoryname" => $row["CATEGORYNAME"]);
        }
        echo json_encode($return_data, JSON_PRETTY_PRINT);
        header('HTTP/1.1 200 OK');
    }
}
else if ($method == "POST") {
else if ($method == "PUT") {
    header('HTTP/1.1 405 Method Not allowed');
}
else if ($method == "DELETE") {
else {
    header('HTTP/1.1 405 Method Not allowed');
}
?>
```

Σχήμα 11 Επιστροφή κατηγοριών σε JSON

3.8.8 Products Resource

Το Resource το οποίο θα αναλυθεί παρακάτω είναι το Products resource το οποίο θα επιτρέπει την προσθήκη, την διαγραφή, αλλά και την επιστροφή των προϊόντων που έχει ο το ηλεκτρονικό κατάστημα. Οι διαθέσιμες μέθοδοι οι οποίες επιτρέπονται για αυτό το resource είναι οι POST, DELETE και GET.

Για αρχή θα αναφερθεί η HTTP μέθοδος DELETE. Στην περίπτωση αυτής της μεθόδου με την μέθοδο query του $mysqli αντικειμένου εκτελούμε το ερώτημα "DELETE FROM Products" το οποίο θα διαγράψει όλα τα προϊόντα του καταστήματος. Τέλος θα επιστραφεί στον client μήνυμα "HTTP/1.1 204 No Content" με την μέθοδο header.

Η επόμενη μέθοδος η οποία εφαρμόζεται είναι η μέθοδος POST η οποία έχει ιδιαίτερο ενδιαφέρον, καθώς όταν θέλουμε να κάνουμε POST ένα καινούργιο προϊόν στην βάση δεδομένων, δεν κάνουμε απλά μεταφορά text, αλλά θα πρέπει να γίνει και μεταφορά εικόνας. Σε αυτό το σημείο πρέπει να αναφερθεί ότι στην βάση δεδομένων δεν γίνεται αποθήκευση της εικόνας, αλλά η αποθήκευση της εικόνας γίνεται στο file storage του μηχανήματος που βρίσκεται ο web server. Η αποθήκευση γίνεται στο directory "./ProductImages". Η μεταφορά από τον client στο server γίνεται με encoding της εικόνας σε μορφή Base64 και βρίσκεται στο json στο κλειδί "product_image". Αυτός ο τρόπος έχει το πλεονέκτημα ότι είναι εύκολο να πάρεις την εικόνα, αλλά αργεί πιο πολύ το request καθώς γίνεται μεγάλο το request body του http request. Το web service θα λάβει τα πεδία "product_title", "product_cost", "product_image" και "categoryid". Στην συνέχεια καλείται η μέθοδος base64ToJpeg που υλοποιήθηκε στο toolbox.php το οποίο δέχεται 2 παραμέτρους, η πρώτη είναι το όνομα της εικόνας, δηλαδή το όνομα του προϊόντος και το δεύτερο είναι η διαδρομή στην οποία θα γίνει η αποθήκευση. Στην συνέχεια κάνουμε insert όλα αυτά τα πεδία τα οποία αναφέρθηκαν παραπάνω και ορίζουμε σαν "modified" στον πίνακα false. Στην συνέχεια κάνουμε prepare το sql ερώτημα, κάνουμε bind τις παραμέτρους οι οποίες αναφέρθηκαν παραπάνω. Τέλος με την μέθοδο execute () κάνουμε εκτέλεση του ερωτήματος και επιστρέφουμε το κατάλληλο μήνυμα. Αν η εκτέλεση γίνει σωστά επιστρέφουμε "HTTP/1.1 201 Created", αλλιώς το web service επιστρέφει "HTTP/1.1 500 Internal Server Error".

Η τρίτη μέθοδος η οποία εφαρμόζεται είναι η GET η οποία θα επιστρέψει όλα στοιχεία από όλα τα προϊόντα όπως φαίνεται στο Σχήμα 12. Για αρχή αφού βεβαιωθούμε ότι έχει δοθεί αυτή η μέθοδος και στην συνέχεια με χρήση της mysqli_query() εκτελούμε το ερώτημα "SELECT * FROM Products". Αν έχει εκτελεστεί σωστά, θα αρχίσουμε να διαβάζουμε γραμμή γραμμή το αποτέλεσμα του ερωτήματος αυτού με την κλήση της μεθόδου fetch_assoc() του αντικειμένου $result η οποία γραμμή φορτώνεται στην μεταβλητή $row σαν πίνακας. Στην συνέχεια για όσο υπάρχουν γραμμές στο αποτέλεσμα κάθε γραμμή παίρνουμε και τα κωδικοποιούμε σε έναν πίνακα όλα τα στοιχεία της δηλαδή 'PID', "PRODUCT_TITLE", "PRODUCT_COST", "ISMODIFIED", "CATEGORYID" και τέλος το "PRODUCT_IMAGE".

Το ενδιαφέρον με το τελευταίο πεδίο είναι ότι έχει περιλαμβάνει το σχετικό μονοπάτι για την εικόνα στο file storage. Για να μπορεί να γίνει αυτή η εικόνα διαθέσιμη στον client θα πρέπει να δημιουργηθεί ένα link για αυτήν την εικόνα ώστε να είναι διαθέσιμη μέσω του διαδικτύου. Για αυτήν την περίπτωση από $_SERVER['SERVER_ADDR'] παίρνουμε την IP του web server στην συνέχεια προσθέτουμε στο link το path μέχρι τον χώρο του web service δηλαδή το directory "/ptyxiaki" και μετά θα ενώσουμε στην συμβολοσειρά το όνομα του σχετικού μονοπατιού για την εικόνα στο file storage της πτυχιακής. Όλα αυτά μπαίνουν σε ένα JSON με όνομα κλειδιών τα ονόματα των στηλών

και σαν τιμές, τα περιεχόμενα αυτής της γραμμής που διαβάστηκε. Αυτή η διαδικασία επαναλαμβάνεται για όλες τις γραμμές. Στην συνέχεια όλα αυτά θα γίνουν json_encode με παραμέτρους JSON_UNESCAPED_SLASHES, JSON_PRETTY_PRINT. Η χρησιμότητα της δεύτερης παραμέτρου έχει εξηγηθεί ενώ η πρώτη έχει σχέση με την μετάφραση του χαρακτήρα "/" οποίος γίνεται escape η ειδική του σημασία προσθέτοντας τον χαρακτήρα αναίρεσης της ειδικής τους σημασίας, δηλαδή το "\". Έτσι μπορεί το link για την φωτογραφία του προϊόντος να εμφανιστεί μόνο με τους χαρακτήρες "/". Τέλος για την σωστή επεξεργασία του request στο web service, επιστρέφουμε όλο το json το οποίο έχει γίνει encode και επιστρέφουμε σαν header HTTP/1.1 200 OK.

```php
<?php
if ($method == "GET") {
    $result = mysqli_query($mysqli, "SELECT * FROM  Products");
    if($result) {
        $return_data = array();
        while ($row = $result->fetch_assoc()) {
            $return_data[] = array(
            "productid" => $row['PID'],
            "product_title" => $row['PRODUCT_TITLE'],
            "productCost" => $row['PRODUCT_COST'],
            "productImage" => $_SERVER['SERVER_ADDR'] . "/ptyxiaki" . substr($row['PRODUCT_IMAGE'], 1),
            "ismodified" => $row['ISMODIFIED'],
            "categoryid" => $row['CATEGORYID']
                                    );
        }
        echo json_encode($return_data, JSON_UNESCAPED_SLASHES | JSON_PRETTY_PRINT);
        header('HTTP/1.1 200 OK');
    }
}
else if ($method == "POST") {
else if ($method == "DELETE") {
    $mysqli->query("DELETE FROM Products");
    header('HTTP/1.1 204 No Content');
}
?>
```

Σχήμα 12 Επιστροφή προϊόντων σε JSON

3.8.9 Customers Resource

Το Resource το οποίο θα αναλυθεί σε αυτήν παράγραφό είναι το Customers το οποίο βρίσκεται στο αντίστοιχο αρχείο. Το συγκεκριμένο Resource περιλαμβάνει διάφορα sub Resources το οποία γίνονται ομαδοποίηση κάτω από τους Customers. Τα subresources τα οποία υπάρχουν είναι τα εξής: Registration, Sessions, Orders, Carts, Bookmarks όπως φαίνεται στο Σχήμα 13. Κάθε ένα subresource δέχεται διαφορετικές παραμέτρους από το json του request body και σε κάθε ένα εφαρμόζονται διαφορετικές μέθοδοί του HTTP. Το πρώτο Resource το οποίο θα μπορούσε να χρησιμεύσει στο μέλλον σε μελλοντική έκδοση του ηλεκτρονικού καταστήματος είναι το Carts. Για την ώρα αν κάποιος client κάνει ένα http call σε αυτό το sub Resource θα του επιστραφεί μήνυμα "HTTP/1.1 501 Not Implented".

Το επόμενο subresource το οποίο υπάρχει είναι το Bookmarks. Αυτό επιτρέπει έναν πελάτη να κάνει POST έναν σελιδοδείκτη για ένα προϊόν. Η μόνη μέθοδος που εφαρμόζεται είναι η μέθοδος POST. Αυτό το Resource λαμβάνει από το json request body 2 βασικά πράγματα. Το πρώτο είναι το "customerid", το δεύτερο είναι το "pid". Εκτός από αυτά τα 2 αποθηκεύεται και η τωρινή ημερομηνία που υπολογίζεται από την μέθοδο date(). Στην συνέχεια, κάνουμε prepare το ερώτημα "INSERT INTO BOOKMARKS (CUSTOMERID, PID, BOOKMARK_DATE) VALUES (?, ?, ?)" όπου θα κάνουμε bind τα πεδία που ήδη αναφέραμε παραπάνω. Αμέσως μετά θα κάνουμε execute το ερώτημα και αν πετύχει σωστά θα επιστρέψουμε στον client μήνυμα επιτυχίας "HTTP/1.1 201 Created".

Το αμέσως επόμενο subresource το οποίο χρειάζεται ανάλυση είναι το Registration. Αυτό βρίσκεται κάτω από Customers Resource και δείχνει ότι το Registration έχει να κάνει με τον Customer και την εγγραφή του στο ηλεκτρονικό κατάστημα. Η μόνη μέθοδος η οποία υποστηρίζεται για αυτό το sub Resource είναι η POST. Για να μπορέσει να λειτουργήσει η εφαρμογή της POST μεθόδου σε αυτό το Resource, πρέπει ο client να στείλει μέσα από το request body τα παρακάτω στοιχεία σε μορφή json. Τα πεδία τα οποία χρειάζονται είναι τα εξής: το "username", "firstname", "lastname", "password",

“primaryemail”, “secondaryemail”, “address”, “gender”, “mobilephone”, “homephone”. Όλα αυτά τα πεδία είναι σαν κλειδιά στο json. Η PHP έχει πρόσβαση από τον πίνακα json σε όλα αυτά τα πεδία και τα τοποθετεί σε αντίστοιχες μεταβλητές. Στην πορεία θα γίνει στην βάση δεδομένων το ερώτημα “INSERT INTO Users (USERNAME, FIRST_NAME, LAST_NAME, PASSWORD, PRIMARY_EMAIL, SECONDARY_EMAIL, ADDRES, GENDER, MOBILEPHONE, HOMEPHONE) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)” το οποίο στην συνέχεια θα γίνει prepare, bind_param για να περαστούν οι παράμετροι στο query τα στοιχεία τα οποία αναφέρθηκαν ποιο πάνω ότι έστειλε ο client στον server. Επειδή το κομμάτι του Registration περιλαμβάνει INSERT σε δύο πίνακες, τον User και τον Customer θα πρέπει να ελεγχθεί κατά πόσο έχει γίνει σωστά το insert και στου δύο πίνακες. Για αυτόν τον λόγο έχουμε μια μεταβλητή $errorOccured η οποία ελέγχει αν υπήρξε πρόβλημα στα διαδοχικά στάδια των 2 ερωτημάτων. Εφόσον οι 2 πίνακες έχουν σχέση ISA τότε το κύριο κλειδί του ενός πίνακα θα βρίσκεται και στον άλλο πίνακα. Για αυτό το ερώτημα διαμορφώνεται στο εξής “INSERT INTO Customers (CUSTOMERID, NEWSLETTER, BANNED) VALUES (?, ?, false)” το οποίο θα το κάνουμε πρώτα prepare με την μέθοδο prepare της $mysqli μεταβλητής, στην συνέχεια θα καλεστεί η bind_param για να περάσει τις τιμές των μεταβλητών στο παραμετροποιήσιμο ερώτημα και τέλος θα εκτελεστεί η μέθοδος execute() για την εκτέλεση του. Αμέσως μετά ελέγχουμε αν έχει υπάρξει σφάλμα οπουδήποτε σε όλη αυτήν την διαδικασία. Στην περίπτωση που έχει υπάρξει σφάλμα, θα επιστρέψει μήνυμα σφάλματος. Αυτό το μήνυμα σφάλματος είναι το “HTTP/1.1 500 Internal Server Error” και το προσθέτουμε σαν κεφαλίδα στο response με την μέθοδο header(). Αν έχει υπάρξει προσθήκη μια γραμμής τότε οι affected_rows του stmt είναι ίση με ένα και θα πρέπει να επιστρέψουμε “HTTP/1.1 201 Created” και σε json το id του πελάτη που μόλις προστέθηκε, για μελλοντική χρήση και πρόσβαση του client στον πελάτη που μόλις προστέθηκε.

Το αμέσως επόμενο Subresource το οποίο υλοποιήθηκε είναι το Sessions το οποίο απεικονίζει τα Sessions των πελατών στην βάση δεδομένων. Οι μέθοδοι που υποστηρίζονται για αυτό το Subresource είναι οι POST και η DELETE. Για την POST υπάρχουν 2 βασικές παράμετροι για να λειτουργήσει. Το subresource παίρνει σαν παράμετρο το “username” και το “password” το οποίο στέλνει ο client σε μορφή JSON, στην συνέχεια εκτελείται ακριβώς ο ίδιος κώδικας με την διαχείριση των SESSIONS των employees που αναφέρθηκαν πιο πάνω, θα γίνει το prepare, το bind_param και επιστρέψει “HTTP/1.1 201 Created” και σε μορφή json το userTokken στον client. Αλλιώς αν υπάρχει σφάλμα σε οποιοδήποτε σημείο της διαδικασίας τότε ο server επιστρέφει “HTTP/1.1 500 Internal Server Error” στον client. Στην συνέχεια όπως με την διαχείριση των SESSION των Employees ο server λαμβάνει το token το οποίο είχε εκδώσει πιο πριν και εκτελεί ένα UPDATE ερώτημα για να θέσει σαν CLOSED την γραμμή του SESSION που αντιστοιχεί με το token που έστειλε ο client για να κάνει αποσύνδεση. Στην περίπτωση που πετύχει θα επιστρέψει “HTTP/1.1 204 No Content” και αν δεν πετύχει θα επιστρέψει “HTTP/1.1 500 Internal Server Error” και με μορφή JSON “{ “error” : “Session could not be closed”}”. Το αμέσως επόμενο SubResource το οποίο χρειάζεται εξήγηση και παίζει κομβικό ρόλο είναι το Orders, το οποίο επιτρέπει την πρόσβαση σε διάφορα είδη κατηγοριών.

```php
<?php
$subresource = $urlEncodedData[0];
$error = "undefined";
$subsubresource = "undefined";
if (isset($urlEncodedData[1])) {
    $subsubresource = $urlEncodedData[1];
}
if ($subresource == "Bookmarks") {
    if ($method == "POST") {
}
if ($subresource == "Carts") {
    header('HTTP/1.1 501 Not Implemented');
}
if ($subresource == "Registration") {
    if ($method == "POST") {
}
if ($subresource == "Sessions") {
    if ($method == "POST") {
    if ($method == "DELETE") {
}
if ($subresource == "Orders") {
    $orderState = "undefined";
    if ($subsubresource == "NewOrdersCheck") {
    if ($method == "GET") {
        if ($subsubresource == "New") {
        if ($subsubresource == "All") {
    }
    if ($method == "POST") {
    if ($method == "PUT") {
}
?>
```

Σχήμα 13 Customer Resource Structure

3.8.10 Administrators Resource

Το αμέσως επόμενο Resource το οποίο χρειάζεται αναφορά είναι το Administrators το οποίο βρίσκεται στο αντίστοιχο αρχείο. Για αρχή πρέπει να αναφερθούν τα διαθέσιμα subresources τα οποία υπάρχουν, τα οποία είναι τα Sessions, Shops και Employees όπως εμφανίζονται στο Σχήμα 14. Σε αυτό το σημείο βλέπουμε πως οι διαχειριστές δεν έχουν κάποιο είδος resource το οποίο να τους κάνει Register. Αυτό συμβαίνει γιατί κάθε διαχειριστή είναι τοποθετημένος στην βάση με SQL ερώτημα κατευθείαν στο DBMS.

Το πρώτο subresource το οποίο θα αναλυθεί είναι το Shops το οποίο δείχνει τα διαθέσιμα μαγαζιά τα οποία υπάρχουν στο ηλεκτρονικό κατάστημα. Σε αυτό το subresource οι μέθοδοι που εφαρμόζονται είναι η POST και η GET. Για αρχή ελέγχεται ποια μέθοδος εφαρμόστηκε και στην περίπτωση της GET, εκτελούμε το ερώτημα “SELECT * FROM SHOPS”. Το ερώτημα θα εκτελεστεί με την χρήση της μεθόδου mysqli_query() περνώντας σαν παράμετρο το mysqli αντικείμενο και το παραπάνω ερώτημα. Στην πορεία αποθηκεύουμε το αποτέλεσμα σε μια μεταβλητή result και με την χρήση της μεθόδου fetch_assoc() του result διαβάζουμε γραμμή το αποτέλεσμα και τοποθετούμε όλα τα στοιχεία των γραμμών σε json με κλειδιά τον τίτλο της στήλης. Κάθε τέτοια γραμμή την τοποθετούμε σε ένα array το οποίο θα κάνουμε json_encode και θα το επιστρέψουμε στην Android εφαρμογή. Η αμέσως επόμενη μέθοδος που εφαρμόζεται στο subresource SHOP είναι η μέθοδος POST. Για την σωστή προσθήκη ενός καινούργιου καταστήματος το web service πρέπει να λάβει τις παρακάτω πληροφορίες, την διεύθυνση (“address”), το όνομα (“firstName”), το επώνυμο (“lastName”) και το όνομα κατήστήματος “shopname”). Όλα αυτά τα πεδία θα γίνουν INSERT στον πίνακα SHOPS. Για αρχή θα γίνει prepare το ερώτημα, στην συνέχεια θα γίνει bind_param όλες οι μεταβλητές στο query και τέλος θα γίνει execute το ερώτημα. Στην περίπτωση που δεν γίνει σωστά η προσθήκη τότε θα εμφανιστεί “HTTP/1.1 500 Internal Server Error” στην εφαρμογή πελάτη, ενώ αν εκτελεστεί σωστά το ερώτημα θα επιστραφεί σε json το id του καταστήματος με την μορφή {“shopid” : id}.

Το αμέσως επόμενο subresource το οποίο υπάρχει στο Administrators είναι το Employees. Αυτό το Resource έχει πρόσβαση στα στοιχεία όλων των υπαλλήλων. Οι μέθοδοι οι οποίοι εφαρμόζονται σε αυτό το Resource είναι οι HTTP μέθοδοι GET, POST και DELETE. Η δομή των υπαλλήλων στην βάση δεδομένων είναι ήδη γνωστή από τον σχεδιασμό του schema της βάσης δεδομένων. Μιλάμε για 2 πίνακες που διαχωρίζουν τα στοιχεία του υπαλλήλου, ο πίνακας Users και ο πίνακας Employees. Για αυτόν τον λόγο στην μέθοδο GET θα εκτελεστεί ένα ερώτημα "SELECT * FROM employees INNER JOIN users ON employees.EMPLOYEEID = users.USERID" το οποίο θα φέρει τα στοιχεία του υπαλλήλου και από τους 2 πίνακες.

Σε αυτό το subresource υπάρχει και μια παράμετρος η οποία έχει κωδικοποιηθεί στο URL. Η παράμετρος αυτή είναι η το τύπος του υπαλλήλου για τον οποίο θα προσθέσει στο ηλεκτρονικό κατάστημα. Τέτοια είδη υπαλλήλων είναι οι Managers, οι Salesmen, οι Warehousemen αλλά και οποιοδήποτε άλλος ρόλος. Στην πορεία υπάρχει και μια επιπλέον παράμετρος η οποία θα παίρνει σαν παράμετρο αυτό το συγκεκριμένο subresource η οποία είναι το "shopid" το οποίο περνιέται σαν παράμετρος στο query string για να μας ορίσει για ποιο κατάστημα να επιστρέψουμε τους υπαλλήλους. Δηλαδή ένα παράδειγμα endpoint είναι το "/api/v1/Administrators/Employees/Managers?shopid=1". Σε αντίθεση με όλα τα υπόλοιπα που έχουν αναφερθεί πιο πάνω οι παράμετροι του URL που βρίσκονται στο query string, δηλαδή μετά το "?" τις εξάγουμε από τον πίνακα $_GET της PHP. Αν έχει προσδιοριστεί το κατάστημα το οποίο θέλουμε τους υπαλλήλους, τότε προσθέτουμε στο SQL ερώτημα το "AND employees.WORKS_IN = SHOPID" το οποίο θα μας περιορίσει τα αποτελέσματα του ερωτήματος. Στην συνέχεια με την χρήση της συνάρτησης mysqli_query() εκτελούμε το ερώτημα περνώντας σαν παράμετρο στην συνάρτηση το $mysqli αντικείμενο και το $sql το οποίο έχουμε φτιάξει πιο πάνω. Στην συνέχεια, αφότου έχει γίνει αποθήκευση του αποτελέσματος στην μεταβλητή $result με την χρήση της μεθόδου fetch_assoc διαβάζουμε κάθε επόμενη γραμμή του αποτελέσματος και τα βάζουμε σε έναν πίνακα employeeDetails ο οποίος περιέχει όλες τις γραμμές του αποτελέσματος με κλειδιά για κάθε τιμή τον τίτλο της στήλης του πίνακα. Στην συνέχεια με την χρήση της μεθόδου json_encode δημιουργούμε το json από την μορφή του πίνακα που είχε πριν και τέλος με την χρήση της echo εντολής το εκτυπώνουμε και το επιστρέφουμε στον client μέσα από response body.

Η αμέσως επόμενη μέθοδος του HTTP που εφαρμόζεται στο subresource Employee είναι η POST, με την οποία μπορούμε να προσθέσουμε όποιον υπάλληλο θέλουμε. Για αρχή ορίζουμε μια μεταβλητή role η οποία θα έχει αρχικά την συμβολοσειρά "role" και στην συνέχεια θα ελεγχθεί τι είδος role έχει δώσει ο client στο web service. Υπάρχουν 3 προκαθορισμένα roles για προσθήκη τα Manager, Salesmen, Warehousemen, αν είναι κάποιο άλλο role τότε προστίθεται σαν custom role στον πίνακα. Στην συνέχεια από τον πίνακα json παίρνουμε τα πεδία τα οποία χρειάζονται για την προσθήκη ενός υπαλλήλου "INSERT INTO Users(USERNAME, FIRST_NAME, LAST_NAME, PASSWORD, PRIMARY_EMAIL, SECONDARY_EMAIL, ADDRESS, GENDER, MOBILEPHONE, HOMEPHONE) VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?)" όλα αυτά τα στοιχεία που χρειάζονται για την για να γίνουν bind στο query θα τα εξάγουμε από τον πίνακα json. Για αρχή το query θα γίνει prepare με την μέθοδο του $mysqli αντικειμένου. Στην συνέχεια θα γίνει bind_param από το αντικείμενο της $mysqli για να περαστούν τα περιεχόμενα των μεταβλητών στο SQL ερώτημα. Τέλος θα γίνει execute αυτό το ερώτημα για να γίνει η προσθήκη του κομματιού του User στον πίνακα.

Στην συνέχεια, εφόσον ο πίνακας Users έχει σχέση ISA με τον πίνακα Employees, θα πρέπει το id της γραμμής του Employee να είναι το ίδιο και για αυτό από το αντικείμενο της mysqli παίρνουμε το κλειδί της τελευταίας εισαγόμενης γραμμής, το πεδίο insert_id. Στην συνέχεια καλούμε την μέθοδο

close του stmt αντικειμένου για να μπορούμε να κάνουμε δημιουργία του επόμενου ερωτήματος. Το επόμενο κομμάτι της διαδικασίας είναι να πάρουμε από τον πίνακα json τα επόμενα πεδία τα οποία θα γίνουν insert στον πίνακα Employees. Τα πεδία είναι τα $json[‘title’], $json[‘identity’], $json[‘startOfContract’], $json[‘endOfContract’] και $json[‘worksIn’], αυτά θα γίνουν insert στον πίνακα Employees. Για αρχή θα γίνει prepare το SQL , στην πορεία θα κάνει bind_param όλες τις μεταβλητές και με την πρώτη παράμετρο “issssi” δηλώνουμε με σειρά των μεταβλητών τι τύπου περιεχόμενο περιέχουν, για παράδειγμα “i” είναι για integer και “s” είναι για String. Αμέσως μετά από αυτό γίνεται η εκτέλεση με την χρήση της execute() μεθόδου του ελέγχουμε αν οι affected_rows, δηλαδή οι γραμμές οι οποίες προστέθηκαν είναι 1 και επιστρέφουμε “HTTP/1.1 201 Created” το οποίο δείχνει ότι η προσθήκη έγινε επιτυχώς. Αλλιώς επιστρέφουμε σε JSON {“error” : “Employee Could not be posted”} και HTTP κωδικό “HTTP/1.1 500 Internal Server Error”.

```php
<?php
$subresource = "undefined";
if (isset($urlEncodedData[0])) {
    $subresource = $urlEncodedData[0];
}
$param = "undefined";
if (isset($urlEncodedData[1])) {
    $param = $urlEncodedData[1];
}
if ($subresource == "Sessions") {
    if (isset($json['administratorTokken'])) {
        $tokken = $json['administratorTokken'];
    }
    if (isset($json['username'])) {
        $username = $json['username'];
    }
    if (isset($json['password'])) {
        $password = $json['password'];
    }
    if ( $method == "POST") {
    if ( $method == "DELETE") {
    if ( $method == "GET") {
}
if ($subresource == "Shops") {
    if ( $method == "GET") {
    if ( $method == "POST") {
}
if ($subresource == "Employees") {
    if ( $method == "GET") {
    if ( $method == "POST") {
    if ( $method == "DELETE") {
    }
}
?>
```

Σχήμα 14 Administrator Resource structure

3.9 Επίλογος

Η ανάπτυξη των Web Services ήταν μια διαδικασία η οποία χρειαζόταν πολύ προσεκτικό σχεδιασμό. Για την ανάπτυξη τους χρειάστηκε συστηματική μελέτη και των λειτουργικών απαιτήσεων που υπάρχουν στην Android εφαρμογή αλλά και των πόρων του DBMS και του Web Server. Για αρχή χρειάστηκε να δοθεί ιδιαίτερη βάση στο πρωτόκολλο HTTP αλλά και στην τεχνολογία του SSL. Στην συνέχεια αναλύθηκε ο τρόπος με τον οποίο συνδέθηκε η PHP των Web Services στην MariaDB και το πώς γίνεται η εκτέλεση των διάφορων ερωτημάτων. Επίσης σημαντική ανάλυση έγινε στην αποκωδικοποίηση των HTTP αιτημάτων στο Web Service αλλά και το πώς ελέγχονται τα δεδομένα που γίνονται αποστολή και λήψη από το Web Service. Σε διάφορες ενότητες του κεφαλαίου βλέπουμε την ανάλυση πολλών διαφορετικών Resources μέσα από τα οποία υπάρχει η πρόσβαση στα στοιχεία της βάσης δεδομένων. Αυτά τα Resources αμέσως μετά την αποκωδικοποίηση του HTTP αιτήματος αναλύονται ανά ενότητα και δείχνουν ανάλογα την μέθοδο του HTTP πρωτοκόλλου με ποιον τρόπο επεξεργάζεται την βάση δεδομένων.

Κεφάλαιο 4ο: Android εφαρμογή

4.1 Εισαγωγή

Το λειτουργικό στο οποίο θα δημιουργηθεί η εφαρμογή είναι η πλατφόρμα του Android.Συγκεκριμένα η εφαρμογή Android θα τρέχει σε Android κινητά παρόλο που η πλατφόρμα χρησιμοποιείται σε πολλά διαφορετικά είδη συσκευών. Η εφαρμογή Android της πτυχιακής στο Web service, το οποίο αναλύθηκε η κατασκευή, η αρχιτεκτονική και η λειτουργία του σε προηγούμενα κεφάλαια. Η εφαρμογή, θα είναι διαθέσιμη στο κινητό τηλέφωνο του υπαλλήλου του ηλεκτρονικού καταστήματος, παρέχοντας όλες τις λειτουργικότητες ενός ολοκληρωμένου eshop. Ο υπάλληλος, θα μπορεί να διαχειρίζεται τα προϊόντα, δηλαδή τα ηλεκτρονικά είδη, να τα ομαδοποιεί σε κατηγορίες, να διαχειρίζεται παραγγελίες και διάφορες άλλες λειτουργίες.

Με την εξέλιξη των κινητών συσκευών, υπάρχει δυνατότητα ανάπτυξης εργονομικών εφαρμογών που βοηθάνε τον υπάλληλο να κάνει την δουλειά του. Η εξέλιξη των κινητών, μας έχουν οδηγήσει από τα παλιά τηλέφωνα με ενσωματωμένα λειτουργικά συστήματα με πολύ βασικές εφαρμογές, που δεν έδιναν ευκαιρίες σε προγραμματιστές να ασχοληθούν με δικές τους εμπορικές εφαρμογές καθώς οι εφαρμογές εξαρτιόταν από το υλικό του κινητού. Τα τελευταία χρόνια υπάρχουν δυο μεγάλοι τύποι λειτουργικών συστημάτων που επικρατούν στην αγορά. Το ένα είναι το IOS και το δεύτερο είναι το Android. Πλέον οι κινητές συσκευές, παρέχουν πολλές διαφορετικές λειτουργικότητες, αισθητήρες και επεξεργαστική ισχύ, πράγμα το οποίο δίνει την δυνατότητα για την ανάπτυξη νέων εφαρμογών που εξυπηρετούν διαφορετικές ανάγκες. Πλέον, τα υπολογιστικά συστήματα των κινητών και τα λειτουργικά τους συστήματα μπορούν να εξυπηρετήσουν ψυχαγωγικές, επικοινωνίας αλλά και εμπορικές εφαρμογές η οποίες διευκολύνουν τους πελάτες για την λήψη προϊόντων ή υπηρεσιών ή τους υπαλλήλους του εκάστοτε καταστήματος για την διαχείριση των εμπορικών τους δραστηριοτήτων.

Η εφαρμογή η οποία αναπτύσσεται είναι καθαρά εμπορικού χαρακτήρα καθώς μιλάμε για την διαχείριση ενός ηλεκτρονικού καταστήματος από ένα android κινητό τηλέφωνο. Το λειτουργικό σύστημα Android είναι ένα σύστημα το οποίο στην αρχή του σχεδιάστηκε για κινητές συσκευές όπως τηλέφωνα ή tablet. Στην πορεία του όμως έχει ευρεία χρήση και σε διάφορες συσκευές όπως android TV, android Auto. Το λειτουργικό αυτό έχει την πιο ευρεία χρήση ξεπερνώντας κατά πολύ το IOS της Apple. Επίσης έχει επεκταθεί και σε άλλους τομείς καθώς είναι το λειτουργικό το οποίο χρησιμοποιείται από το Διαδίκτυο των πραγμάτων, καθώς όλο και περισσότερο υπάρχει μια μεγαλύτερη συνδεσιμότητα των έξυπνων συσκευών διάφορων ειδών όπως οικιακές συσκευές ή ακόμα και χρήση διάφορων αισθητήρων.

4.2 Android Λειτουργικό

Το λειτουργικό σύστημα Android έχει βάση στο Linux. Το Android έχει μια συγκεκριμένη αρχιτεκτονική η οποία περιλαμβάνει πολλά διαφορετικά στρώματα με βάση έναν ενσωματωμένο πυρήνα Linux. Η αρχιτεκτονική αποτελείται από μια στοίβα λογισμικού, με τον πυρήνα που αναφέρθηκε, το ενδιάμεσο λογισμικό middleware και κάποιες βασικές εφαρμογές. Γενικά οι διάφορες εφαρμογές που υπάρχουν σε αυτό το λειτουργικό, επικοινωνούν με ένα application layer το οποίο το οποίο περιλαμβάνει διάφορες εφαρμογές. Αυτές οι εφαρμογές είναι γενικού σκοπού και οι οποίες χρησιμοποιούνται από άλλες εφαρμογές. Τέτοιες εφαρμογές είναι οι επαφές, οι χάρτες και η κάμερα. Η βασική λογική της πλατφόρμας του Android είναι η διευκόλυνση ανάπτυξης νέων εφαρμογών οι οποίες θα είναι για πιο συγκεκριμένες ανάγκες του τελικού χρήστη. Ένα άλλο σημαντικό σημείο με

την εξάπλωση και την ευρεία χρήση των Android συσκευών είναι η δημιουργία cross platform εφαρμογών, δηλαδή να μπορεί να τρέχουν σε πολλές διαφορετικές συσκευές με διαφορετικό υλικό η καθεμία. Σε αυτό το σημείο υπάρχει ανάγκη να αναφερθεί ότι η γλώσσα στην οποία συντάχθηκαν αυτές οι εφαρμογές είναι η JAVA η οποία μπορεί να εκτελείται σε πολλές συσκευές. Αυτές οι εφαρμογές είναι στο πιο υψηλό layer του android stack.

4.2.1 Application Framework

Το αμέσως επόμενο layer στο android stack είναι το πλαίσιο εφαρμογών (Application Framework). Αυτό το στρώμα αποτελείται από κάποια συγκεκριμένα API τα οποία είναι διαθέσιμα σε προγραμματιστές για την δημιουργία νέων εφαρμογών αλλά και για την επαναχρησιμοποίηση του κώδικα. Τα κομμάτια τα οποία αποτελούν αυτό Layer είναι ο Activity Manager, ο Window Manager, ο Package Manager, ο Telephony Manager, ο Content Provider, ο Resource Manager, το View System, ο Location Manager, ο Notification Manager και το XMPP

4.2.2 Activity Manager

Ο συγκεκριμένος manager είναι υπεύθυνος για τον κύκλο ζωής των διάφορων activities. Τα activities στο android σαν κάθε ξεχωριστό παράθυρο σε μια desktop εφαρμογή. Το experience από το desktop pc διαφέρει πολύ με το android καθώς κάθε activity μπορεί να βρίσκεται στο παρασκήνιο και να εκτελείται ή να είναι σταματημένο και ο χρήστης να το επαναφέρει όποτε θέλει. Μια εφαρμογή μπορεί να έχει ένα ή πολλά περισσότερα activities. Για κάθε ένα activity υπάρχει ένα κύκλος ζωής και κάποια στάδια από τα οποία περνάει. Ο activity manager περιλαμβάνει διάφορες μεθόδους οι οποίες ορίζονται στα activities για να αντιλαμβάνονται αυτές τις αλλαγές στα στάδια της ζωής ενός activity. Με αυτόν τον τρόπο όταν ένας χρήστης μπορεί ένας χρήστης να εναλλάσσεται μεταξύ των διάφορων εφαρμογών και να μπορεί το λειτουργικό να το αντιλαμβάνεται βοηθώντας σε κρίσιμα ζητήματα τις εφαρμογές. Τέτοια ζητήματα είναι το να μην κάνει crash μια εφαρμογή αν σε μια οθόνη κινητού, αλλάζει το orientation, από portrait σε landscape και το αντίστροφο. Να βελτιστοποιεί την απόδοση του κινητού μη χρησιμοποιώντας τα διάφορα resources του κινητού, όταν ο χρήστης δεν χρησιμοποιεί την εφαρμογή. Επίσης στο να μην παθαίνουν crash οι όποιες εφαρμογές, με το που γίνεται η εναλλαγή, από την μια στην άλλη.

4.2.3 Package Manager και Telephony Manager

Ο Package manager είναι υπεύθυνος για την εγκατάσταση και την απεγκατάσταση εφαρμογών στο λειτουργικό του κινητού. Επίσης ο διαχειριστής τηλεφωνίας (Telephony Manager). επιτρέπει την στον προγραμματιστή να κάνει χρήση των υπηρεσιών SMS και MMS. Επίσης δίνει πληροφορίες όπως ο σειριακός αριθμός της κάρτας SIM, το είδος του δικτύου. Αυτές μπορεί ο developer μπορεί να έχει πρόσβαση, αλλά και να αλλάξει και την κατάσταση του κινητού. Μπορεί να ελέγξει αν είναι σε κατάσταση Roaming, καθώς και άλλες πληροφορίες όπως το Network Country ISO, τον αριθμό του Voice Mail και το IMEI του κινητού.

4.2.4 Android Kernel

Όπως αναφέρθηκε πριν ο πυρήνας του android έχει μεγάλες ομοιότητες με του Linux, αν και έχουν κάποιες διαφορές. Μια μεγάλη του διαφορά είναι ότι στο Android ο πυρήνας είναι πιο μικρός καθώς δεν περιλαμβάνει drivers. Επίσης σημαντικό είναι ότι το Android έχει ενισχυμένο πυρήνα με προσαρμοσμένα στο κινητό χαρακτηριστικά τα οποία χρησιμεύουν σε κινητά περιβάλλοντα, σε

αντίθεση με άλλα περιβάλλοντα όπως επιτραπέζιους υπολογιστές στους οποίους δεν θα είναι χρήσιμα. Οι ομοιότητες του Android με το Linux έχουν κυρίως να κάνουν με τις υπηρεσίες του συστήματος. Υπηρεσίες τέτοιες μπορεί να είναι η διαχείριση μνήμης, για διαχείριση των διεργασιών αλλά και η στοίβα δικτύου. Ο πυρήνας χρησιμοποιείται επίσης και ως ένα επίπεδο αφαίρεσης, δηλαδή ένα στρώμα μεταξύ του υλικού και την στοίβα του λογισμικού. Αυτό έχει ως σκοπό την υποστήριξη πολλών drivers, δηλαδή οδηγούς για την διαχείριση του υλικού όπως υποστηρίζεται και στο Linux. Ο πυρήνας του Linux, είναι ένα από τα μεγαλύτερα project. Το έτος 2016 δουλέψανε πάνω από 4000 προγραμματιστές από 450 εταιρίες, αναπτύσσοντας ένα project το οποίο αποτελούνταν από 56 χιλιάδες αρχεία και 22 εκατομμύρια γραμμές κώδικα.

4.3 Android Studio

Για την ανάπτυξη της Android εφαρμογής χρειάζεται ένα IDE (Integrated Development Environment). Η λύση η οποία επιλέχθηκε για την επιλογή ολοκληρωμένου περιβάλλοντος είναι το Android Studio το επίσημο IDE για την ανάπτυξη Android εφαρμογών. Το Android Studio βασίζεται στο IntelliJ IDEA και παρέχει διάφορα χαρακτηριστικά για να βοηθήσει τον προγραμματιστή να αυτοματοποιήσει κάποιες διαδικασίες και να μεγιστοποιήσει την απόδοση του όσον αφορά την ταχύτητα και τον χρόνο της ανάπτυξης του λογισμικού. Παρέχει διάφορα εργαλεία όπως τα building tools. Το Android Studio περιλαμβάνει ένα ολοκληρωμένο σύστημα building το οποίο μπορεί να κάνει compile όλο το project και να κάνει εγκατάσταση την εφαρμογή σε οποιαδήποτε συσκευή. Το Gradle είναι ένα ολοκληρωμένο αυτοματοποιημένο εργαλείο build για JVM (Java Virtual Machine) γλώσσες προγραμματισμού, όπως η Java, η Groovy και η Scala. Το gradle αυτόματα παίρνει όλα πηγαία αρχεία .java και .xml και χρησιμοποιώντας το κατάλληλο εργαλείο, τα ομαδοποιεί όλα και τα μετατρέπει σε ένα συμπιεσμένο αρχείο το .apk. Αυτό το αρχείο μπορεί ο προγραμματιστής να το ελέγξει είτε σε κανονική συσκευή κάνοντας της εγκατάσταση με χρήση ενός καλωδίου USB και κάνοντας την εγκατάσταση μέσα από το Android Studio, είτε χρησιμοποιώντας τον κατάλληλο emulator που παρέχει το Android studio.

Ο emulator για να μπορέσει να τρέξει θέλει κάποιες απαιτήσεις συστήματος. Για αρχή θέλει SDK Tools 26.1.1, 64 bit επεξεργαστή και HAXM 6.2.1 ή πιο καινούργιο. Όσον αφορά το υλικό έχει κάποιες επιπλέον απαιτήσεις ανάλογα αν ο υπολογιστής είναι Windows ή Linux. Είτε σε Linux είτε σε Windows χρειάζεται Intel επεξεργαστής με υποστήριξη Intel VT-x Intel EM64T και λειτουργία Execute Disable (XD) Bit. Για Linux χρειάζεται επεξεργαστή με υποστήριξη AMD-V και SSSE3. Τέλος όσον αφορά τα συστήματα Windows 32 bit από τον Ιούνιο του 2019 σταμάτησε η υποστήριξη μέχρι και τον Ιούνιο του 2019 οπού συνέχισε η υποστήριξη διορθώνοντας κάποια σφάλματα στο λογισμικό του Android Studio, αλλά δεν θα υπάρξουν καινούργια χαρακτηριστικά στο μέλλον.

Επίσης, όσον αφορά τον έλεγχο της εφαρμογής, το Android Studio περιλαμβάνει εργαλεία τόσο για την απόδοση της εφαρμογής, όσο και για την ευχρηστίας της αλλά και για την δημιουργία εκδόσεων του κώδικα και τα θέματα συμβατότητας. Για παράδειγμα μπορεί μέσα από το εργαλείο δημιουργίας προφίλ να ελέγξει ο προγραμματιστής να κάνει αξιολόγηση τους χρόνους απόκρισης της εφαρμογής καθώς και άλλα τεχνικά θέματα όπως το ποσοστό χρησιμοποίησης του επεξεργαστή, αξιολόγηση του Java Heap, αλλά και να μετρήσει την κίνηση δικτύου που υπάρχει κατά την διάρκεια της εκτέλεσης της εφαρμογής. Επιπρόσθετα στοιχεία είναι και το ποσοστό μνήμης που χρησιμοποιεί, αλλά και το ποσοστό της ενέργειας που καταναλώνεται κατά την διάρκεια της εκτέλεσης της εφαρμογής. Επίσης το Android Studio περιέχει διάφορα frameworks και ένα ενιαίο περιβάλλον για τον έλεγχο της εφαρμογής σε παραπάνω από ένα είδος συσκευών.

Επίσης πολύ σημαντικό σημείο ενός IDE είναι η δυνατότητα του προγραμματιστή να ανεβάζει κώδικα σε ένα online repository. Αυτό είναι πολύ χρήσιμο σημείο, καθώς ο προγραμματιστής δεν θα χρειάζεται να χρησιμοποιεί κάποια άλλη εφαρμογή για διαχείριση των εκδόσεων του κώδικα. Έτσι αντί για παράδειγμα να χρησιμοποιεί ένα GitHub GUI client, να μπορεί να κάνει push τις αλλαγές στον κώδικα στο online repository. Έτσι θα μπορεί να συνεργάζεται με άλλους προγραμματιστές για

την υλοποίηση ενός μεγάλου project. Επίσης, αυτό το GitHub Integration, όχι μόνο επιτρέπει στον προγραμματιστή να ανεβάζει τον κώδικα του, αλλά δίνει και πρόσβαση και πρότυπα κώδικα (code templates), έτσι ώστε να είναι εύκολα υλοποιήσιμα συχνά χαρακτηριστικά. Επίσης, μαζί όλα όσα περιλαμβάνει το Android Studio είναι η υποστήριξη για Google Cloud Platform. Έτσι μπορώντας να κάνει integrate το Google Cloud Messaging. Εκτός από αυτό το Android Studio παρέχει υποστήριξη Native Development Kit (NDK) το οποίο είναι ένα σύνολο από εργαλεία τα οποία επιτρέπουν στον προγραμματιστή να χρησιμοποιήσει κώδικα C και C++ στην εφαρμογή που θέλει.

4.4 Project Files

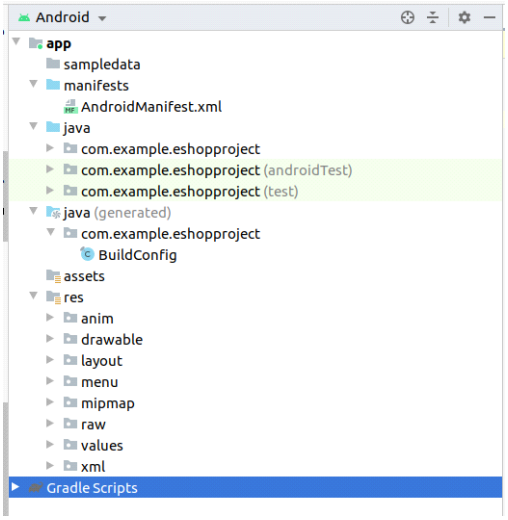
4.4.1 Δομή

Η δομή του project για την δημιουργία της Android εφαρμογής, όπως διαμορφώνεται από το Android studio είναι η εξής όπως φαίνεται και στο Σχήμα 15. Για αρχή είναι ο φάκελος manifests ο οποίος περιλαμβάνει το αρχείο manifest.xml. Στο project της πτυχιακής υπάρχει μόνο ένα αρχείο manifest.xml το οποίο περιλαμβάνει σημαντικές πληροφορίες για την εφαρμογή, για τα εργαλεία building που χρησιμοποιεί το Android, το λειτουργικό σύστημα καθώς και το Google Play. Ένα Android project μπορεί να έχει παραπάνω από ένα manifest όταν μια εφαρμογή έχει παραπάνω από ένα flavors. Flavor είναι μια παραλλαγή της εφαρμογής. Τέτοιου τύπου παραλλαγές είναι η ελεύθερη έκδοση μιας εφαρμογής και η πληρωμένη έκδοση της. Τα διάφορα flavors επίσης θα οριστούν στο αρχείο build.gradle.

Αμέσως επόμενο στην λίστα είναι ο φάκελος java ο οποίος περιλαμβάνει όλα τον πηγαίο κώδικα της java. Ο πηγαίος κώδικας της java χωρίζεται σε αρχεία που περιέχονται οι κλάσεις του project. Τέτοιες κλάσεις είναι οι model class οι οποίες απλά περιέχουν δεδομένα όπως Order,Product, Employee. Αμέσως μετά άλλες κλάσεις που ρυθμίζουν την λειτουργικότητα της διεπιφάνειας όπως οι LoginActivity class και ChartsFragment. Άλλες κλάσεις είναι οι Adapter classes που αποτελούν το ενδιάμεσο στάδιο της επεξεργασίας και της εμφάνισης των δεδομένων στην διεπιφάνεια.

Επίσης εκτός από τον πηγαίο κώδικα κάτω από το java folder υπάρχουν και οι φάκελοι com.example.(appname) που έχει άλλα 2 αρχεία java, τα ExampleInstrumentedTest και το ExampleUnitTest. Στο δευτερο αρχείο JAVA το ExampleUnitTest.java μπορεί ο προγραμματιστής να προσθέσει τα διάφορα Unit Tests τα οποία θα τεστάρουν τις μεθόδους του project. Κατά την διάρκεια της πτυχιακής δεν υπήρχε η ανάγκη να γραφτεί κώδικας για να ελεγχθούν οι υπάρχοντες μέθοδοι. Στο αρχείο ExampleInstrumentedTest.java όπου τοποθετούνται τα Instrumented Tests.

Αμέσως μετά ο επόμενος φάκελος είναι ο assets. Αυτός ο φάκελος είναι απαραίτητος για να τοποθετούνται χρήσιμα αρχεία για την εφαρμογή, όπως μουσική, μορφοποιήσεις κειμένου, βίντεο αλλά xml αρχεία. Μέχρι τώρα έχουν αναφερθεί πράγματα για κώδικα java ο οποίος ρυθμίζει την λειτουργικότητα της διεπιφάνειας, αλλά όχι για τα αρχεία τα οποία γίνεται η σχεδίαση της διεπιφάνειας. Αυτά τα αρχεία καθώς και διάφορα άλλα σχετικά αρχεία τοποθετούνται στον φάκελο res (resources). Οι υποφάκελοι οι οποίοι εμφανίζονται εδώ είναι οι anim, drawable, layout, menu, mipmap, raw, values, xml. Ο φάκελος layout περιλαμβάνει όλες τις σχεδιάσεις όλων των view είτε είναι activities, fragments ή άλλα views. Επίσης σημαντικό ρόλο παίζει ο φάκελος drawable. Μέσα σε αυτόν τον φάκελο υπάρχουν όλα τα εικονίδια που χρησιμοποιούνται στην εφαρμογή. Αυτά τα εικονίδια όπως και όλα τα εικονίδια τα οποία τοποθετούνται εκεί μπορούν να χρησιμοποιηθούν από τα xml αρχεία για να εμπλουτίσουν την σχεδίαση της διεπιφάνειας. Εκτός από τις εικόνες αυτές υπάρχουν και τα αρχεία shapes, με τα οποία μπορείς να δώσεις σχεδίαση στα ήδη υπάρχοντα views του xml της διεπιφάνειας.



Σχήμα 15 Δομή Project

Επίσης όσον αφορά την σχεδίαση της εφαρμογής και το καλύτερο user experience είναι η χρήση διάφορων animation. Τα διάφορα animation τοποθετούνται στον φάκελο anim όπου αποθηκεύονται σαν xml. Εδώ φαίνεται πως και οι εικόνες αλλά και τα animation αποθηκεύονται σαν xml αρχεία. Έτσι και τα διάφορα menu που μπορέι να έχει η εφαρμογή αποθηκεύονται σαν xml. Τέτοια menu μπορεί να είναι ένα bottom_navigation bar ή ένα navigation drawer. Τελευταίος υποφάκελος ο οποίος χρειάζεται ανάλυση είναι ο xml, οποίος περιλαμβάνει ένα αρχείο network security config, το οποίο ορίζει την διαδρομή μέχρι το πιστοποιητικό που έχει δώσει ο διαχειριστής συστήματος για την χρήση του SSL για τα API αιτήματα από την Android εφαρμογή στο Web service.

4.4.2 Manifest.xml

Ένα σημαντικό σημείο σε κάθε Android project είναι το αρχείο AndroidManifest.xml. Το αρχείο αυτό περιγράφει βασικές πληροφορίες για αυτό καθώς και για τα εργαλεία τα οποία θα χρησιμοποιηθούν για να γίνει το build, αλλά και για το λειτουργικό σύστημα του Android και το Google Play όπως φαίνεται στο Σχήμα 16. Το αρχείο αυτό θα πρέπει να υπάρχει οπωσδήποτε αλλά και να έχει και το συγκεκριμένο όνομα. Το αρχείο αυτό θα πρέπει να περιέχει πληροφορίες σχετικά με build, τα διάφορα components της εφαρμογής, τα δικαιώματα της εφαρμογής, αλλά και χαρακτηριστικά του λογισμικού και του υλικού. Για αρχή, όσον αφορά το build θα πρέπει να περιέχεται το package name της εφαρμογής. Το package name χρησιμοποιείται από το Android Build tools για να εντοπίσει τις βασικές οντότητες που περιέχονται μέσα στο project. Κατά την διάρκεια του build, το package name θα αντικατασταθεί από το application ID το οποίο έχει οριστεί στο Gradle, για χρησιμοποιηθεί ως μοναδικός προσδιοριστής της εφαρμογής κατά την μεταφορά της στο Google Play. Επίσης, όσον αφορά το Google Play στο android manifest υπάρχουν χαρακτηριστικά της εφαρμογής τα οποία δείχνουν ποια κινητά είναι ικανά να χρησιμοποιήσουν την εφαρμογή από άποψη λειτουργικού συστήματος αλλά και υλικού.

Αμέσως μετά σημαντικό κομμάτι είναι τα δικαιώματα (permissions) τα οποία χρειάζεται να χρησιμοποιήσει η εφαρμογή. Τελευταίο, αλλά πολύ σημαντικό είναι να ορίζονται τα διάφορα components όπως services, activities και receivers. Για αρχή, ως προς την δομή του, είναι και αυτό σε xml και πρέπει να έχει πολύ συγκεκριμένη δομή. Το root element είναι το <manifest> για το οποίο είναι απαραίτητο να οριστεί το package attribute το οποίο περιλαμβάνει το οποίο έχει την τιμή “com.examply.eshop”. Για αρχή αυτό χρησιμοποιείται για να οριστεί το namespace της εφαρμογής.

Το package name χρησιμεύει για να οριστεί το R class το οποίο χρησιμεύει για να έχουμε πρόσβαση στα resources του project. Η R κλάση περιέχει τους ορισμούς για όλα για όλα τα resources του project. Για παράδειγμα, στην περίπτωση μας εφόσον το package name είναι "com.example.eshopproject", τότε δημιουργείται μια κλάση com.example.eshopproject.R.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.eshopproject">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        android:networkSecurityConfig="@xml/network_security_config">
        <activity...>
        <activity android:name=".WarehousemanManagement" />
        <activity android:name=".MainActivity"...>
        <activity android:name=".ManagerActivity"...>
        <activity android:name=".LoginActivity"...>

        <service...>

        <receiver...>
    </application>

</manifest>
```

Σχήμα 16 Δομή manifest

4.4.3 Δικαιώματα

Τα δικαιώματα τα οποία ορίζονται στο manifest.xml έχουν ιδιαίτερη σημασία καθώς επηρεάζουν τις λειτουργίες που μπορεί να περιλαμβάνει η εφαρμογή. Τα 2 δικαιώματα τα οποία χρησιμοποιούνται στην εφαρμογή είναι τα INTERNET και το RECEIVE_BOOT_COMPLETED. Το πρώτο δικαίωμα έχει να κάνει με την πρόσβαση της εφαρμογής στο διαδίκτυο. Με αυτό το δικαίωμα η εφαρμογή, θα μπορεί να αλληλεπιδράσει με το web service και να έχει πρόσβαση στα βασικά δεδομένα της εφαρμογής. Χωρίς αυτό το δικαίωμα, η εφαρμογή δεν μπορεί να λειτουργήσει. Το δεύτερο δικαίωμα είναι να μπορεί η εφαρμογή να αντιλαμβάνεται πότε το κινητό έκανε BOOT. Αυτό το δικαίωμα είναι αναγκαίο έτσι ώστε να μπορεί το Notification Service να γίνεται triggered και να γίνεται εκκίνηση του με την έναρξη του κινητού. Η δήλωση του permission γίνεται μέσα στο root element, δηλαδή το manifest και είναι της μορφής "<uses-permission android:name= "PERMISSION_NAME"/>".

4.4.4 Δήλωση Application και Activities

Για αρχή πρέπει να δηλώσουμε την εφαρμογή μας στο manifest. Η εφαρμογή γίνεται με την δήλωση ενός <application> element. Μέσα σε αυτό το element θα δηλωθούν τα activities, τα services και οι receivers. Για αρχή ιδιαίτερη αναφορά πρέπει να γίνει στο networkSecurityConfig attribute. Αυτό μας δείχνει την διαδρομή για το αρχείο network_security_config στο οποίο υπάρχουν όλες οι ρυθμίσεις της εφαρμογής για το κομμάτι της επικοινωνίας μέσω δικτύου. Επίσης μπορούμε να ορίσουμε και το εικονίδιο της εφαρμογής μέσω του icon attribute.

Αμέσως μετά όπως φαίνεται στην φωτογραφία υπάρχουν 3 activities τα οποία είναι δηλωμένα. Είναι τα AdministratorManagement, ManagerActivity, LoginActivity. Όπως έχει ήδη αναφερθεί, κάθε activity είναι ένα ξεχωριστό παράθυρο στο Android χωρίς αυτό να σημαίνει πως κάθε παράθυρο έχει μόνο ένα view. Η πρώτη δήλωση είναι το AdministratorManagement. Σε αυτό βλέπουμε πως έχουμε ορίσει σαν name το AdministratorManagement το οποίο αντιστοιχεί στην κλάση του activity. Επίσης σημαντικό είναι ότι έχουμε κλειδώσει το screenOrientation σε "portrait", δεδομένου ότι στην εφαρμογή δεν υπάρχει η ανάγκη για την υποστήριξη σε "landscape". Δεδομένου αυτού του

κλειδώματος, ο editor βγάζει προειδοποιήσεις, για αυτό θέσαμε το attribute ignore έτσι ώστε να αγνοεί την LockedOrientationActivity ειδοποίηση που εντοπίζει ο editor. Επίσης βλέπουμε πως υπάρχει ένα intent filter.

Γενικά τα activities, τα services αλλά και οι broadcast receivers ενεργοποιούνται από τα intent. Όταν αναφερόμαστε σε ένα Intent μιλάμε για ένα μήνυμα το οποίο δημιουργεί ένα Intent object και ορίζει την ενέργεια που θα εκτελέσει. Τα intent filters λειτουργούν επικοινωνώντας με το σύστημα του Android. Στην περίπτωση που μια εφαρμογή, δημιουργήσει ένα intent για το σύστημα, στην συνέχεια αυτό το Intent αντικείμενο μεταφέρεται προς τον προορισμό του component του Intent. Το σύστημα βρίσκει το component και δημιουργεί ένα instance από αυτό, στο οποίο μεταφέρονται τα δεδομένα του αντικειμένου. Στην περίπτωση που παραπάνω από μια εφαρμογή μπορεί να κάνει handle το intent, ο χρήστης θα κληθεί να επιλέξει ποια εφαρμογή να επιλέξει.

4.4.5 Network Security Config

Ένα άλλο πολύ σημαντικό αρχείο στην επικοινωνία της Android εφαρμογή με το Web service είναι το αρχείο network_security_config.xml. Το αρχείο Network Security Configuration επιτρέπει την παραμετροποίηση των ρυθμίσεων δικτύου σε ένα μέρος το οποίο, δεν θα τροποποιείται από τον κώδικα της εφαρμογής. Οι ρυθμίσεις αυτές μπορούν να εφαρμοστούν για συγκεκριμένες εφαρμογές και συγκεκριμένα domains. Όπως έχει ήδη αναφερθεί, στο manifest έχει δηλωθεί το attribute networkSecurityConfig το οποίο δείχνει στο αρχείο που περιγράφουμε. Όπως φαίνεται στο Σχήμα 17 ως root element ορίζεται ένα <network-security-config> element. Αμέσως μετά ορίζουμε το configuration για ένα συγκεκριμένο domain με το element <domain-config>. Τέλος ορίζουμε το domain το οποίο η εφαρμογή θα εμπιστεύεται η εφαρμογή για την επικοινωνία, κάτι το οποίο ορίζεται με τον ορισμό της στατικής IP του server. Τελευταίο αλλά πολύ σημαντικό είναι ο ορισμός ενός <certificate> element το οποίο ορίζει σαν src την διαδρομή μέχρι το πιστοποιητικό το οποίο έχει δώσει διαχειριστής συστήματος του server στην client εφαρμογή για την επικοινωνία με αυτόν. Το αρχείο server.crt, περιέχει μια κωδικοποιημένη συμβολοσειρά η οποία η εφαρμογή θα χρησιμοποιήσει για να κάνει κωδικοποίηση και αποκωδικοποίηση τα δεδομένα τα οποία στέλνει ή λαμβάνει από τον server.

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config>
        <domain includeSubdomains="true">192.168.1.70</domain>
        <trust-anchors>
            <certificates src="@raw/server"/>
        </trust-anchors>
    </domain-config>
</network-security-config>
```

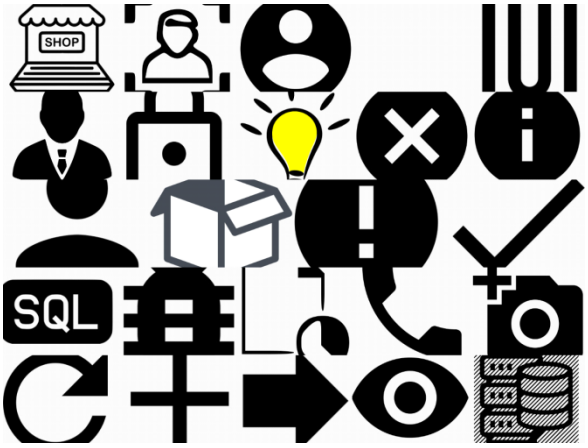
Σχήμα 17 Αρχείο ρυθμίσεων δικτύου

4.4.6 Application Icons

Για την καλύτερη σχεδίαση της διεπιφάνειας της εφαρμογής αλλά και για να γίνει πιο πλούσια η εμπειρία του χρήστη είναι η ανάγκη να υπάρχουν και διάφορα εικονίδια τα οποία θα χρησιμοποιούνται από την διεπιφάνεια της Android εφαρμογής. Οι εικόνες αυτές θα είναι πακεταρισμένες μέσα στο apk. Όσον αφορά το apk, η το building tool (Gradle) ψάχνει μέσα στον φάκελο drawable για να βρει τις εικόνες. Για την πτυχιακή χρησιμοποιήθηκαν διάφορα εικονίδια.

Στην πλειοψηφία τους ήταν τα προεπιλεγμένα εικονίδια του Android studio τα οποία βρίσκονται κάτω από Apache License Version 2.0. Στο φαίνονται όλα τα εικονίδια που χρησιμοποιήθηκαν.

Το Android studio μαζί με τα άλλα του εργαλεία, περιλαμβάνεται και ένα εργαλείο που λέγεται Image Asset Studio, το οποίο βοηθάει τον προγραμματιστή να δημιουργεί δικά του εικονίδια. Το Image Asset Studio βοηθάει για την δημιουργία launcher icon δηλαδή το εικονίδιο της εφαρμογής και notification icons δηλαδή τα εικονίδια τα οποία εμφανίζονται σαν ειδοποιήσεις στην Android συσκευή, αλλά και app bar και tab bar εικονίδια. Επίσης σημαντικό είναι πως ο προγραμματιστή μπορεί να κάνει import τα δικά του icons για να χρησιμοποιηθούν στην εφαρμογή. Το Asset Studio υποστηρίζει τις μορφές png, jpg και gif. Επίσης, σημαντικό είναι και το vector Asset Studio, το οποίο προσθέτει ένα vector γραφικό με την μορφή XML που περιγράφει ένα image. Η αποθήκευση γίνεται σε XML καθώς η συντήρηση ενός XML αρχείου είναι ευκολότερη από την επεξεργασία ενός bitmap γραφικού.



4.5 Gradle

4.5.1 Γενικά

Το Gradle είναι λογισμικό ανοιχτού κώδικα το οποίο χρησιμοποιείται για την αυτοματοποίηση του Build μιας εφαρμογής. Το Gradle παρέχει μια μεγάλη ευελιξία ώστε να μπορεί να δημιουργήσει Build από διάφορα είδη λογισμικού. Η εργαλειοθήκη του Gradle χρησιμοποιείται επίσης και από το Android Studio για την διαχείριση της διαδικασίας του Build επιτρέποντας στον προγραμματιστή να την τροποποιήσει και να κάνει τις δικές του παραμετροποιήσεις. Κάθε παραμετροποίηση αντιστοιχεί σε ένα build το οποίο μπορεί να έχει την δικιά του διαφορετική λειτουργικότητα. Για παράδειγμα ένα build να αντιστοιχεί σε ένα flavor δηλαδή μια παραλλαγή της εφαρμογής. Ένα χαρακτηριστικό παράδειγμα μπορεί να είναι ένα build μόνο για development και ένα άλλο build να είναι αυτό το οποίο βρίσκεται στην αγορά.

4.5.2 Build Gradle

Ένα σημαντικό σημείο στην περιγραφή του Build είναι το αρχείο Build Gradle το οποίο περιλαμβάνει το configuration για το build της εφαρμογής της πτυχιακής. Το σημείο αυτό εμφανίζεται στο Σχήμα 18. Αρχικά το πρώτο σημείο το οποίο είναι πολύ σημαντικό και χρειάστηκε κατά την διάρκεια του project για την χρησιμοποίηση των διάφορων βιβλιοθηκών είναι το dependencies block. Σε αυτό το block καθορίζονται τα προαπαιτούμενα για να ολοκληρωθεί το build. Το σύστημα του Gradle βοηθάει τον προγραμματιστή, όταν θέλει να χρησιμοποιήσει άλλες

εξωτερικές βιβλιοθήκες ή άλλα modules για το build του project. Οι βιβλιοθήκες αυτές μπορεί είτε να βρίσκονται σε κάποιο άλλο repository ή απλά να βρίσκονται στο τοπικό μηχάνημα. Οι βιβλιοθήκες για να είναι χρησιμοποιήσιμες πρέπει να οριστούν σαν dependencies σε αυτό το block.

Η γραμμή implementation fileTree(dir: 'libs', include: ['*.jar']) ορίζει τα dependencies τα οποία βρίσκονται στο τοπικό μηχάνημα του προγραμματιστή. Όλες οι υπόλοιπες γραμμές ορίζουν dependencies για βιβλιοθήκες οι οποίες βρίσκονται σε απομακρυσμένα repositories προγραμματιστών. Επίσης κάθε γραμμή στα implementation αποτελούν συντομεύσεις. Για παράδειγμα η γραμμή implementation 'com.google.android.material:material:1.0.0' αποτελεί μια συντόμευση για το εξής implementation group: 'com.google.android.material', name: 'material', version : '1.0.0'. Στην γραμμή η οποία εμφανίζεται με κόκκινη υπογράμμιση απλά το περιβάλλον παρέχει μια πρόταση για να πάμε σε νεότερη έκδοση της βιβλιοθήκης. Το build ολοκληρώνεται κανονικά με αυτήν την έκδοση της βιβλιοθήκης, οπότε για την ώρα δεν είναι ανάγκη να κάνουμε migrate. Επίσης παρατηρούνται και άλλες δυο γραμμές με άλλα 2 configurations εκτός από το implementation. Πρόκειται για τα testImplementation και το androidTestImplentation τα οποία προσθέτουν dependencies για τα local tests και instrumented tests. Άλλα configurations που υπάρχουν είναι το api, το compileOnly, το runtimeOnly, annotationProcessor, lintChecks και lintPublish.

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:design:26.1.0'
    implementation 'com.google.android.material:material:1.0.0'
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation 'com.google.android.material:material:1.1.0'
    implementation 'androidx.annotation:annotation:1.1.0'
    implementation 'androidx.lifecycle:lifecycle-extensions:2.2.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
    implementation 'com.squareup.okhttp3:okhttp:3.9.1'
    implementation 'com.google.code.gson:gson:2.8.6'

    implementation 'com.github.PhilJay:MPAndroidChart:v3.0.3'
}
```

Σχήμα 18 Dependencies

Το αμέσως επόμενο σημαντικό για το build είναι το android block. Σε αυτό το block ορίζονται όλες οι συγκεκριμένες παράμετροι του build. Για αρχή είναι το compileSdkVersion το οποίο ορίζει το Android level Gradle στο οποίο θα κάνει compile η εφαρμογή. Με αυτήν την παράμετρο, σημαίνει πως η εφαρμογή θα μπορεί να χρησιμοποιεί χαρακτηριστικά αυτού API επιπέδου και χαμηλότερα. Αμέσως μετά είναι το το buildToolsVersion το οποίο ορίζει την έκδοση από τα εργαλεία του SDK και τα διάφορα utilities τα οποία θα χρησιμοποιεί το gradle για να φτιάξει το build, στην περίπτωση της πτυχιακής είναι 29.0.3.

Αμέσως επόμενο block είναι το defaultConfig το οποίο περιέχει κάποια default settings. Το ποιο σημαντικό στοιχείο το οποίο περιορίζει και των αριθμό των συσκευών που θα χρησιμοποιήσουν την εφαρμογή είναι το minSdkVersion το οποίο ορίζει το μικρότερο API επίπεδο που μπορούν να τρέξουν την εφαρμογή. Στην περίπτωση της πτυχιακής το API level είναι 24 δηλαδή η εφαρμογή μπορεί να

τρέξει μόνο σε κινητά τηλέφωνα με έκδοση Android 7.0 και πάνω (Σχήμα 19). Αμέσως επόμενο χρήσιμο στοιχείο είναι το targetSdkVersion, δηλαδή την έκδοση του API στην οποία ο προγραμματιστής συνήθιζε να τεστάρει την εφαρμογή. Αυτό είναι πολύ χρήσιμο για να γα φαίνεται σε ποια έκδοση Android είναι πιο σταθερή η Android εφαρμογή. Αμέσως μετά, παρατηρούμε άλλα 2 πεδία τα οποία όμως δείχνουν την έκδοση της εφαρμογής Android που αναπτύσσουμε. Τα πεδία είναι το versionCode και το versionName. Το πρώτο δείχνει την πραγματική έκδοση της εφαρμογής σε μορφή κωδικού, ενώ το δεύτερο πεδίο δείχνει την έκδοση σε μια μορφή πιο φιλική προς τον χρήστη. Τελευταίο πεδίο στο defaultConfig block είναι το applicationId το οποίο είναι ο μοναδικός προσδιοριστής της εφαρμογής. Με την χρήση αυτού του κωδικού προσδιορίζουμε το package της εφαρμογής για να γίνει δημοσίευση.

Όσον αφορά την έκδοση του Android στην οποία θα μπορεί να τρέξει η εφαρμογή, είναι σημαντικό να γίνει κατανοητό πως κάθε έκδοση Android περιλαμβάνει και κάποιες βιβλιοθήκες οι οποίες αλλάζουν όσο ανεβαίνει η έκδοση του Android αλλά και του API level. Για την υλοποίηση του project χρησιμοποιήθηκαν βιβλιοθήκες της Java οι οποίες είναι σε έκδοση 8. Αυτή η έκδοση της Java αυτόματα ανεβάζει το API Level και φαίνεται και στο Gradle. Στο compileOptions εμφανίζεται το jdk 1.8, το οποίο δείχνει ότι χρησιμοποιούμε Java 8. Επίσης, με το πεδίο sourceCompatibility δείχνουμε την έκδοση Java η οποία χρησιμοποιείται για να κάνει compile τα αρχεία .java ενώ με την επιλογή targetCompatibility βεβαιώνουμε ότι τα αρχεία .class τα οποία θα δημιουργηθούν από .java θα είναι συμβατά με τα VM (Virtual Machines) τα οποία τα έχουν την έκδοση του targetCompatibility. Τα αρχεία .class θα τρέξουν στην έκδοση του targetCompatibility και πιο καινούργιες εκδόσεις, αλλά όχι σε παλαιότερες εκδόσεις των VM

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 29
    buildToolsVersion "29.0.3"

    defaultConfig {
        applicationId "com.example.eshopproject"
        minSdkVersion 24
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility = 1.8
        targetCompatibility = 1.8
    }

}
```

Σχήμα 19 Ρυθμίσεις συμβατότητας εφαρμογής

Το αμέσως επόμενο block το οποίο υπάρχει είναι το buildTypes το οποίο ορίζει τις ιδιότητες του Gradle, όσον αφορά το build και το package της εφαρμογής και αυτές οι ιδιότητες ορίζονται για πολλά στάδιο του κύκλου ανάπτυξης μιας εφαρμογής. Ένα χαρακτηριστικό παράδειγμα είναι η Debug version της εφαρμογής. Όταν δημιουργείται ένα Debug Build type τότε το gradle υπογράφει το APK με ένα Debug key και ενεργοποιεί τα Debug options. Στην άλλη περίπτωση που δημιουργείται ένα Release Build Type, το APK πριν υπογραφεί με ένα Release key γίνεται η διαδικασία της σμίκρυνσης για να μην πιάνει πολύ χώρο το APK αλλά και του obfuscation κατά το οποίο μπερδεύεται ο κώδικας.

4.5.3 Top-level build file

Το άλλο αρχείο στο οποίο πρέπει να γίνει ιδιαίτερη αναφορά είναι το build.gradle (Eshop). Σε αυτό το σημείο ορίζονται configurations τα οποία αφορούν όλα τα modules του project όπως φαίνεται στο Σχήμα 20. Σε αυτό το σημείο ορίζονται τα τα Gradle repositories αλλά και τα dependencies τα οποία είναι κοινά και χρησιμοποιούνται από όλα τα modules που βρίσκονται στο project.

Για αρχή είναι το buildscript block, το οποίο ρυθμίζει τα repositories και τα dependencies που χρησιμοποιεί και το ίδιο το gradle, για αυτόν τον λόγο δεν θα πρέπει ο προγραμματιστής, να προσθέσει τα δικά του. Το πρώτο block είναι το repositories το οποίο ρυθμίζει τα repositories τα οποία χρησιμοποιούνται από το Gradle για την αναζήτηση και την λήψη των dependencies. Στο project της πτυχιακής φαίνονται ήδη τα 2 default repositories google(), jcenter().

Το αμέσως επόμενο block το οποίο υπάρχει είναι τα dependencies τα οποία χρειάζεται το gradle για να μπορέσει να γίνει το build του project. Με την γραμμή 'com.android.tools.build:gradle:3.6.1' ορίζουμε την διαδρομή για την έκδοση της βιβλιοθήκης του gradle. Το πρώτο block είχε να κάνει με τις διαδρομές και τα repositories του ίδιου το Gradle, ενώ το επόμενο block το allprojects περιλαμβάνει αντίστοιχα τις διαδρομές και τα repositories για τις βιβλιοθήκες που χρειάζεται το ίδιο το project. Στην συνέχεια, παρακάτω βλέπουμε το πρώτο task το οποίο ορίζεται στο gradle. Το clean task με type delete διαγράφει το φάκελο με το build. Το clean task χρησιμοποιείται όταν αλλάζουμε κάποιες ρυθμίσεις στο settings.gradle όπου μπορεί να απαιτεί την διαγραφή των αρχείων class.

```
buildscript {

    repositories {
        google()
        jcenter()

    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.6.1'
        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        google()
        jcenter()
        maven { url "https://jitpack.io" }
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

Σχήμα 20 Build Script

4.6 OkHttp

4.6.1 Περιγραφή βιβλιοθήκης

Εφόσον ο τρόπος για να επικοινωνήσουμε με το Web API είναι το HTTP θα πρέπει να χρησιμοποιηθεί μια βιβλιοθήκη για την επικοινωνία με τον server. Για την επικοινωνία της Android εφαρμογής με το Web API χρειάστηκε μια βιβλιοθήκη η οποία να μπορεί να κάνει HTTP αιτήματα. Η βιβλιοθήκη η οποία χρησιμοποιήθηκε είναι το OkHttp. Το OkHttp είναι ένας HTTP client ο οποίος χρησιμοποιείται στην Java και βοηθάει τον προγραμματιστή να κάνει αιτήματα και να λαμβάνει απόκριση από τον server. Η βιβλιοθήκη OkHttp χρησιμοποιείται σε εκδόσεις Android με API Level μεγαλύτερο από 21. Επίσης πρέπει να υπάρχει τουλάχιστον έκδοση Java 8 ή και μεγαλύτερη.

Η βιβλιοθήκη αυτή αρχικά έχει το χαρακτηριστικό ότι μειώνει την καθυστέρηση μεταφοράς του αιτήματος, δηλαδή μειώνει τον χρόνο αποστολής από την Android εφαρμογή κατά το αίτημα και από το Web API στην εφαρμογή κατά την απόκριση. Επίσης σχετικά με αυτό είναι ότι η βιβλιοθήκη μπορεί να κάνει cache διάφορες αποκρίσεις του HTTPS, ώστε να εξοικονομήσουμε bandwidth και χρόνο και να επαναχρησιμοποιηθούν ενώ έχουν αποθηκευτεί στο file system. Επίσης σημαντικό είναι ότι το OkHttp κάνει καλή διαχείριση όταν το δίκτυο είναι προβληματικό. Δηλαδή σε περίπτωση που υπάρξουν προβλήματα σύνδεσης και χαθεί η σύνδεση, η βιβλιοθήκη θα επανέλθει στην σύνδεση, πράγμα το οποίο απαλλάσσει τον προγραμματιστή από τον σχεδιασμό της βιβλιοθήκης δικτύου.

Ένα άλλο κομμάτι το οποίο είναι ιδιαίτερα σημαντικό είναι να βεβαιωθούμε ότι τα δεδομένα θα αποσταλούν ακέραια στον server. Εφόσον το HTTP βασίζεται μόνο στο TCP τότε μπορούμε να βεβαιωθούμε για την ακεραιότητα των δεδομένων. Σε πολλές περιπτώσεις θα χρειαστεί να μεταφερθούν δεδομένα στον server που είναι ευαίσθητα, όπως αριθμοί πιστωτικών καρτών, για αυτό και ο server επιτρέπει την μεταφορά μόνο με χρήση SSL. Σε μια τέτοια περίπτωση, ο client θα πρέπει να μπορεί να χρησιμοποιήσει το συγκεκριμένο πρότυπο. Στην συγκεκριμένη βιβλιοθήκη το χρησιμοποιείται TLS 1.3, εκμεταλλευόμενο την ασφάλεια του.

4.6.2 Υλοποίηση στην εφαρμογή

Για να μπορέσει η βιβλιοθήκη να χρησιμοποιηθεί από την Android εφαρμογή είναι ανάγκη να γίνει import. Για αυτό τον λόγω πρέπει να οριστεί το κατάλληλο dependency στο gradle. Για αυτό και στο αρχείο build.gradle στα dependencies θα πρέπει να προσθέσουμε την γραμμή “implementation 'com.squareup.okhttp3:okhttp:3.9.1'”. Αυτό χρησιμεύει για να μπορέσει το εργαλείο για το building να κατεβάσει τις βιβλιοθήκες και να είναι έτοιμες προς χρήση από τον προγραμματιστή. Για να μπορεί όμως η εφαρμογή να μπορεί να κάνει τα HTTP αιτήματα χωρίς ο προγραμματιστής να παραμετροποιεί κάθε φορά την σύνδεση με το Web Service, δημιουργήθηκε μια ενδιάμεση κλάση RestClient όπου θα υπάρχει ολόκληρη η παραμετροποίηση της σύνδεσης, αλλά και για να υπάρχει και μια σταθερή σουίτα μεθόδων με το οποίες θα μπορεί να στέλνει και να παίρνει δεδομένα από τον server χωρίς να μπλέκεται ο κώδικας των μεθόδων του model με τον κώδικα που απλά περιέχει τεχνικές λεπτομέρειες του HTTP. Η κλάση αυτή εμφανίζεται στο Σχήμα 21.

Για αρχή δηλώνουμε ένα πεδίο τύπου OkHttpClient. Αυτή η μεταβλητή απλά θα περιλαμβάνει το αντικείμενο του client που θα δημιουργηθεί στην συνέχεια. Για αυτό τον λόγο στην συνέχεια στον δομητή θα δημιουργηθεί ένα αντικείμενο OkHttpClient. Με την χρήση του newBuilder μπορούμε να φτιάξουμε ένα αντικείμενο και να το παραμετροποιήσουμε. Για αρχή ορίζουμε ένα connectTimeOut το οποίο είναι ο χρόνος στον οποίο θα πρέπει ο client, δηλαδή η Android εφαρμογή να συνδεθεί με το Web Service. Αυτή η παράμετρος έχει να κάνει ξεκάθαρα με την κατάσταση του δικτύου και κατά πόσο φόρτο έχουν τα δίκτυα από τα οποία θα περάσουν τα πακέτα του IP. Για αυτό πρέπει αν για κάποιον λόγο δεν υπάρχει δίκτυο στην Android εφαρμογή να κάνει SocketTimeOutException για να αντιληφθεί πως τελικά δεν υπάρχει δίκτυο και το ορίζουμε σε 10 μονάδες της κλίμακας του δευτερολέπτου.

Αμέσως μετά είναι το writeTimeOut, το οποίο είναι ο χρόνος ο οποίος επιτρέπει η Android εφαρμογή για να στείλει τα δεδομένα στον server για να γράψει δεδομένα. Στην περίπτωση της εφαρμογής μας, από τον client στον server δεν μεταφέρονται μεγάλες ποσότητες δεδομένων, οπότε 10 δευτερόλεπτα κρίθηκαν αρκετά. Δεδομένων των διάφορων παραδειγμάτων που αναφέρονται στην τεκμηρίωση του Web Service το μεγαλύτερο δεδομένο που μπορεί να μεταφερθεί είναι ένα μια φωτογραφία προϊόντος. Άρα 10 δευτερόλεπτα κρίθηκαν αρκετά για να ανέβει μια εικόνα στον server.

Το αμέσως επόμενο σημείο το οποίο θέλει ρύθμιση είναι το readTimeout δηλαδή τον χρόνο που θέλει η εφαρμογή να διαβάσει δεδομένα από τον server. Επειδή από τον server προς το κινητό μεταφέρονται πολλά δεδομένα υπάρχει ανάγκη για περισσότερο χρόνο, δηλαδή 30 δευτερόλεπτα. Σε όλες τις παραπάνω περιπτώσεις αν υπερβεί κάποιο χρονικό όριο η σύνδεση με τον server θα υπάρξει SocketTimeOutException.

Τελευταίο σημείο του builder είναι η χρήση ενός HostNameVerifier. Αυτο το αντικείμενο ορίζει μια μέθοδο verify η οποία θα πρέπει να επιβεβαιώνει το σωστό host. Αυτή μέθοδος verify πρέπει να υλοποιείται όταν βγαίνει στην αγορά η εφαρμογή για αυτό και όταν είναι σε φάση ελέγχου η εφαρμογή δεν κάνει verify με ποιον host επικοινωνεί, συνεχίζοντας όμως να παρέχει κρυπτογράφηση για τα δεδομένα τα οποία μεταφέρονται μέσω δικτύου. Ένα άλλο χρήσιμο στοιχείο το οποίο πρέπει να αναφερθεί είναι το κλειδί API “x-api-key” το οποίο όπως θα φανεί παρακάτω θα το προσθέτει σαν header στο HTTP για δείχνει στον server ότι είναι μια πιστοποιημένη εφαρμογή και να μπορέσει να εκτελέσει API αιτήματα στο Web Service. Επίσης τελευταίο σημείο το οποίο βοηθάει στην ασφάλεια του API είναι ένα πεδίο token. Αυτό το πεδίο token θα μεταφέρεται από τον client στον server με κάθε API αίτημα μέσω της κεφαλίδας “Authorization” για να γνωρίζει το API ποιος χρήστης κάνει την κάθε ενέργεια.

```
public class RestClient {

  private final String apiKey = "1234567890";
  private String token = "tokenReturnedByServer";
  Context context;
  OkHttpClient client;
  int httpCode;

  public RestClient() {

    client = new OkHttpClient().
            newBuilder().
            connectTimeout( timeout: 10, TimeUnit.SECONDS).
            writeTimeout( timeout: 10,TimeUnit.SECONDS)
            .readTimeout( timeout: 30,TimeUnit.SECONDS).
                    hostnameVerifier(new HostnameVerifier() {
      @Override
      public boolean verify(String hostname, SSLSession session) { return true; }
    }).build();
  }

  public JSONObject doPost(String url, JSONObject body) throws JSONException, IOException {...}

  public JSONArray doGetRequest(String url) throws JSONException,IOException{...}

  public JSONObject doGetRequestSingleObject(String url) throws JSONException,IOException{...}

  public void doDeleteRequest (String url, JSONObject body) throws  JSONException,IOException {...}

  public void setToken(String token) { this.token = token; }
}
```

Σχήμα 21 Δομή κλάσης για HTTP αιτήματα

Οι τέσσερις βασικές μέθοδοι οι οποίες χρησιμοποιεί η Android εφαρμογή για την επικοινωνία με τον server είναι οι doPost(String url, JSONObject body), doGetRequest(String url), doGetRequestSingleObject(String url) και doDeleteRequest (String url, JSONObject body). Εφόσον και οι τέσσερις επικοινωνούν με τον server μπορεί να κάνουν throw IOException. Για αυτό τον λόγο η διεπιφάνεια η οποία θα χρησιμοποιήσει αυτές τις μεθόδους θα πρέπει να κάνει διαχείριση των περιπτώσεων αυτών. Αμέσως επόμενο Exception το οποίο μπορεί να στείλουν κατά την εκτέλεση του είναι το JSONException. Εφόσον το Web API δέχεται και στέλνει δεδομένα με την μορφή JSON αν υπάρξει κάποιο πρόβλημα στην μορφή του αρχείου λήψης ή αποστολής, θα πετάξει αυτό το είδους Exception. Και οι τέσσερις μέθοδοι παίρνουν σαν παράμετρο ένα URI, δηλαδή το resource στο οποίο θα κάνουν το αίτημα.

Η πρώτη μέθοδος για να πάρει δεδομένα από τον server είναι η μέθοδος doGetRequestSingleObject(String url). Η ουσία της μεθόδου είναι πως θα περάσει σαν παράμετρος ένα URI και σε αυτό το endpoint θα πάει και θα κάνει μια κλήση στο Web Service. Όπως φαίνεται

στο Σχήμα 22 φτιάχνουμε ένα αντικείμενο τύπου Request, με την χρήση του Builder, ορίζουμε το URL στο οποίο θα γίνει το αίτημα και τελευταίο ορίζουμε το header με το όνομα “x-api-key”, το οποίο υπάρχει σαν πρωτόκολλο του HTTP για να μπορεί να ελέγχει το Web Service το είδος της εφαρμογής. Στην συνέχεια το κάνουμε build το περνάμε σαν παράμετρο στον HTTP client που ορίσαμε στην κλάση και το εκτελούμε. Από την εκτέλεση της μεθόδου execute μας επιστρέφεται ένα αντικείμενο τύπου Response το οποίο περιέχει την απάντηση από τον server. Στην πορεία απλά το μετατρέπουμε σε JSONObject από απλή συμβολοσειρά. Για λόγους αποσφαλμάτωσης πριν ολοκληρωθεί η μέθοδος, υπάρχει ένα log για να καταγράφεται ότι επιστρέφει ο server. Τέλος, για να κλείσει την σύνδεση πρέπει να γίνει κλήση της μεθόδου close(). Άξιο αναφοράς είναι πως η συγκεκριμένη μέθοδος καλείται για να λάβει μόνο ένα JSON object. Ένα JSON αντικείμενο έχει μια συγκεκριμένη δομή. Περικλείεται από αγκύλες {} και μέσα σε αυτό είναι οι τιμές αυτού του αντικειμένου σαν μορφή κλειδί και τιμή. Τα κλειδί πρέπει να είναι συμβολοσειρές και οι τιμές μπορεί να είναι string, number, object, array, boolean και null. Επίσης το κλειδί με την τιμή χωρίζονται με τον χαρακτήρα “:” και κάθε ζεύγος κλειδιού με το άλλο χωρίζονται από ένα κόμμα “,”. Για αυτόν τον λόγο όταν η Android εφαρμογή περιμένει ένα τέτοιο αποτέλεσμα από Web Service χρησιμοποιείται αυτή η μέθοδος.

```
public JSONObject doGetRequestSingleObject(String url) throws JSONException,IOException{
  Request newRewuest = new  Request.Builder().url(url)
          .addHeader( name: "x-api-key", apiKey)
          .addHeader( name: "Authorization", token).build();

  Response response = client.newCall(newRewuest).execute();
  String data = response.body().string();
  Log.d( tag: "Message extracted",data);

  JSONObject jsonObject = new JSONObject(data);
  response.body().close();
  return jsonObject;
}
```

Σχήμα 22 Μέθοδος επιστροφής JSON με την GET HTTP μέθοδο

Η δεύτερη μέθοδος που χρησιμοποιεί η Android εφαρμογή για την επικοινωνία με το Web Service είναι η doGetRequest(String url) η οποία επιστρέφει δεδομένα με διαφορετική μορφή όπως εμφανίζεται στο Σχήμα 23. Γενικά το Web Service μπορεί να επιστρέψει εκτός από ένα απλό JSON αντικείμενο, μπορεί να φέρει έναν πίνακα από JSON αντικείμενα. Αυτή η μορφή λέγεται JSONArray και για αυτόν τον λόγο και έχει διαφορετικό τύπο επιστροφής η μέθοδος.

```
public JSONObject doGetRequestSingleObject(String url) throws JSONException,IOException{
  Request newRewuest = new  Request.Builder().url(url)
          .addHeader( name: "x-api-key", apiKey)
          .addHeader( name: "Authorization", token).build();

  Response response = client.newCall(newRewuest).execute();
  String data = response.body().string();
  Log.d( tag: "Message extracted",data);

  JSONObject jsonObject = new JSONObject(data);
  response.body().close();
  return jsonObject;
}
```

Σχήμα 23 Μέθοδος επιστροφής JSON με την GET HTTP μέθοδο

Η τρίτη μέθοδος του client είναι η doPost(String url, JSONObject body) η οποία χρησιμοποιείται για να στέλνει δεδομένα στο Web Service. Ουσιαστικά παίρνει σαν παράμετρο ένα endpoint στο οποίο

θα στείλει τα δεδομένα και παίρνει και σαν παράμετρο ένα JSONObject. Αυτό το JSONObject είναι οτιδήποτε μπορεί να θέλει να κάνει post η εφαρμογή. Αυτό θα μπορούσε να είναι ένα προϊόν, μια κατηγορία ή ένας υπάλληλος. Ότι σταλεί στον server θα πρέπει να είναι με το κατάλληλο content-type σαν σώμα του αιτήματος του μηνύματος, αλλά και να προστεθεί και η κατάλληλη κεφαλίδα στο HTTP μήνυμα. Για αρχή φτιάχνουμε ένα MediaType αντικείμενο με το οποίο ορίζουμε τί είδους μορφή θα έχουν τα δεδομένα. Κάνοντας parse “application/json;charset=utf-8”, ορίζεται σε Java αντικείμενο ότι έχουμε JSON format. Στην συνέχεια φτιάχνουμε ένα RequestBody αντικείμενο java το οποίο δηλώνει το σώμα του HTTP αίτημα αλλά και το content-type, με τις μεταβλητές JSON και το περιεχόμενο του αντικειμένου που θα στείλουμε σαν plain text. Στην συνέχεια φτιάχνουμε όπως πριν ένα αίτημα με την διαφορά ότι καλούμε και την μέθοδο post παίρνοντας το RequestBody που έχουμε ορίσει πιο πάνω. Έτσι με αυτόν τον τρόπο περάσαμε το σώμα του μηνύματος. Στην συνέχεια όπως πριν κάνουμε execute το request για να πάρουμε ότι στέλνει ο server σε ένα αντικείμενο Response όπου παίρνουμε τα δεδομένα σαν απλό κείμενο. Στην συνέχεια, απλά επιστρέφουμε την απόκριση του Web Service με ένα JSONObject. Η μέθοδος εμφανίζεται στο Σχήμα 24. Ιδιαίτερη αναφορά πρέπει να γίνει πως συνέχεια χρειάζονται διάφορα logs έτσι ώστε ο προγραμματιστή να ελέγχει όλα τις αποκρίσεις του Web Service. Τέτοιες αποκρίσεις είναι τι HTTP κωδικό επέστρεψε, τα δεδομένα τα οποία επέστρεψε καθώς και τους διάφορους χρόνους αποστολής και λήψης. Αυτοί οι χρόνοι ήταν που λήφθηκαν υπόψη για την παραμετροποίηση του OkHttpClient.

```
public JSONObject doPost(String url, JSONObject body) throws JSONException, IOException {
  Log.d( tag: "Rest call",  msg: "started");

  MediaType JSON = MediaType.parse("application/json;charset=utf-8");
  RequestBody requestBody = RequestBody.create(JSON,body.toString());
  Request request = new Request.Builder().url(url).post(requestBody)
          .addHeader( name: "x-api-key", apiKey)
          .addHeader( name: "Authorization", token).build();
  Response response = client.newCall(request).execute();
  String data = response.body().string();

  Log.d( tag: "Rest call",  msg: "ended");
  Log.d( tag: "Response code: ",  msg: "" + response.code());
  Log.d( tag: "Response data: ", data);
  Log.d( tag: "Request time:",  msg: "" + response.sentRequestAtMillis());
  Log.d( tag: "Response time:",  msg: "" + response.receivedResponseAtMillis());

  JSONObject jsonObject = new JSONObject(data);
  response.body().close();
  return  jsonObject;
}
```

Σχήμα 24 Μέθοδος αποστολής δεδομένων στο Web Service

4.7 Model

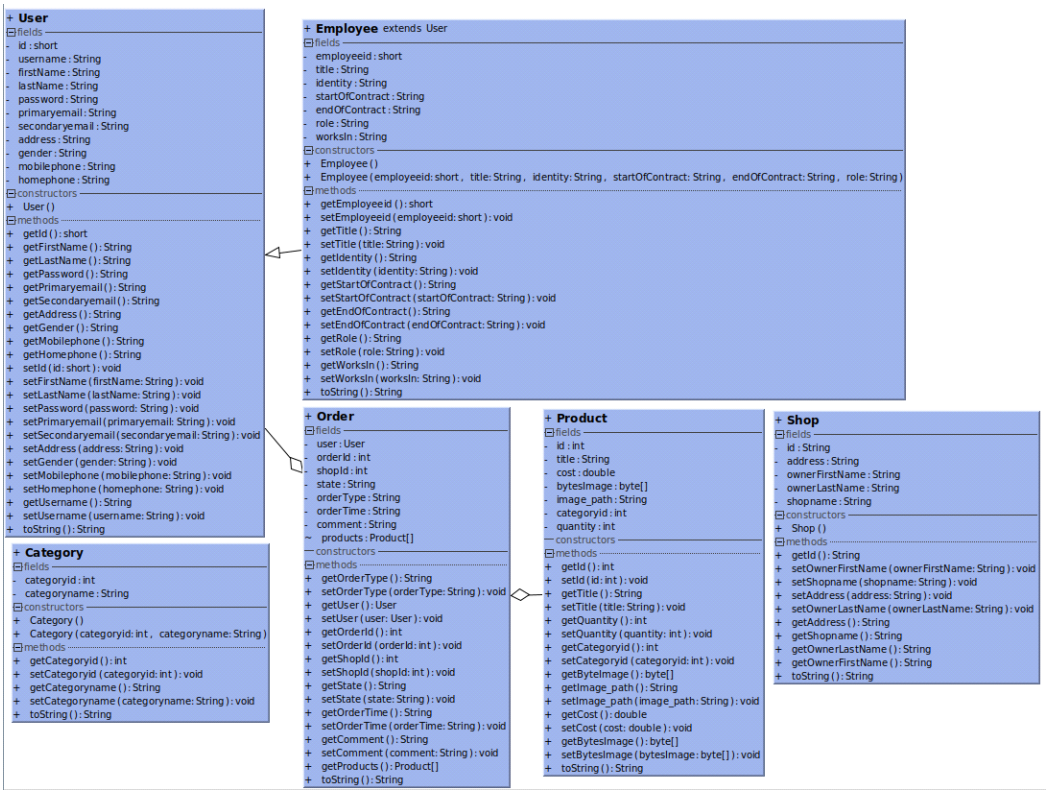
4.7.1 Gson

Αμέσως μετά από την σύνδεση και την επιστροφή των δεδομένων από τον server, περνάμε στο αμέσως επόμενο επίπεδο του project το οποίο είναι οι model κλάσεις στην Android εφαρμογή. Οι model κλάσεις είναι αυτές από τις οποίες θα προκύψουν τα αντικείμενα που θα περικλείουν τα δεδομένα. Γενικά η αντικειμενοστρέφεια είναι χρήσιμη καθώς με την χρήση της ενθυλάκωσης μπορεί ο προγραμματιστής να ομαδοποιήσει ομοειδή δεδομένα και να φτιάξει νέους τύπους δεδομένων

χτίζοντας κλάσεις πάνω στα primitive types, double, float, int, long, short, char αλλά και στην τελική κάνοντας ενθυλάκωση και άλλα αντικείμενα όπως String, HashMap.

Χάρη στην μοντελοποίηση του συστήματος σε κλάσεις μπορούμε να κάνουμε μαζικές επεξεργασίες στα δεδομένα, χωρίς να χρειάζεται να ξέρουμε την ακριβή τους λειτουργία, αλλά με την χρήση των δημόσιων τους μεθόδων να μπορούμε τις χρησιμοποιήσουμε. Η πιο απλή λειτουργία τους είναι η μεταφορά δεδομένων. Και για αυτόν τον λόγο χρειάζεται και ένας τρόπος για την μετατροπή από plain text που επιστρέφει το Web Service σε ένα POJO (Plain Java Object). Χρειάζεται κάποιου είδους mapping από την μορφή του JSON σε ένα αντικείμενο έτοιμο για χρήση από τον προγραμματιστή. Όπως αναφέρθηκε πιο πάνω για την επιστροφή δεδομένων από το Web Service έχουμε σαν τύπο επιστροφής είτε ένα JSONObject ή ένα JSONArray. Αυτό είναι το πρώτο βήμα με το οποίο η εφαρμογή Android αναγνωρίζει πως έχει ένα αντικείμενο JSON ή έναν πίνακα με JSON. Το επόμενο βήμα περιλαμβάνει μια λογική οργάνωση των δεδομένων το ένα ως προς το άλλο.

Οι model κλάσεις είναι οι εξής: Category, Order, Product, Shop, User, Employee και εμφανίζονται στο Σχήμα 25. Για την τεκμηρίωση των model κλάσεων παρακάτω παρατίθεται το διάγραμμα κλάσεων. Υπενθυμίζεται ότι στην κορυφή των ορθογωνίων παραλληλογράμμων είναι το όνομα της κλάσης. Αμέσως μετά είναι η περιοχή με τα πεδία τα οποία είναι της μορφής “visibility: variable : variable type”. Η ορατότητα είναι “-” για private, “+” για public και “~” για default. Αμέσως παρακάτω είναι οι δομητές και οι μέθοδοι. Οι μέθοδοι όπως και οι μεταβλητές είναι της μορφής “visibility: method name(parameter : Type) : Type”. Για το κρίσιμο κομμάτι του mapping από JSON σε POJO, χρειάζεται μια βιβλιοθήκη η οποία να μπορεί να το κάνει αυτό χωρίς την ανάγκη δημιουργίας ενός custom parser. Η λύση η οποία χρησιμοποιήθηκε είναι η βιβλιοθήκη Gson. Η βιβλιοθήκη Gson χρησιμοποιείται από την Java για την μετατροπή αναπαραστάσεις JSON σε Java Object και το αντίθετο. Το Gson είναι ένα project ανοιχτού κώδικα το οποίο μπορεί με την χρήση απλών μεθόδων toJson(), fromJson() να κάνει τις αντίστοιχες μετατροπές. Για να χρησιμοποιηθεί από την Android εφαρμογή θα πρέπει στο Gradle να προστεθεί στα dependencies η γραμμή “implementation 'com.google.code.gson:gson:2.8.6' “ για να μπορέσουν οι βιβλιοθήκες να γίνουν import από την Android εφαρμογή.



Σχήμα 25 Διάγραμμα κλάσεων

4.7.2 Category Class

Για την μετατροπή των διάφορων κατηγοριών σε αντικείμενα χρειάζεται πρώτα από όλα ένας σχεδιασμός μιας κλάσης η οποία θα απεικονίζει κάθε διαφορετική κατηγορία, όπως περιγράφεται και στην βάση δεδομένων. Ορίζουμε αυτήν την κλάση Category η οποία περιλαμβάνει σαν πεδία (fields) τα δεδομένα μιας κατηγορίας και εμφανίζεται στο Σχήμα 26. Και τα δυο πεδία είναι private και η πρόσβαση σε αυτά γίνεται μόνο μέσω των public μεθόδων τους. Περιλαμβάνει έναν κενό δομητή για την δημιουργία του αντικειμένου με default τιμές και έναν πλήρη δομητή που ορίζει τις τιμές όλων πεδίων όπως ορίζεται στην κλήση του δομητή. Τέλος για λόγους αποσφαλμάτωσης, ορίστηκε μια μέθοδος toString() η οποία επιστρέφει σαν συμβολοσειρά, όλα τα ονόματα των πεδίων και τις τιμές τους.

Για να μπορέσει να γίνει το mapping, σημαντικό ρόλο παίζει το annotation @SerializedName, με την χρήση αυτού του annotation η βιβλιοθήκη να καταλάβει ποιο πεδίο του JSON θα γίνει mapping σε ποιο πεδίο του αντικειμένου αλλά και το αντίστροφο. Με αυτήν την λογική αν στο JSON υπάρχει ένα ζεύγος κλειδιού τιμής, με κλειδί “categoryname” και μια τιμή, τότε αυτή του τιμή θα γίνει map στο πεδίο που έχει το annotation “categoryname”. Αντίστοιχα συμβαίνει και με το πεδίο categoryid. Στην αντίθετη περίπτωση που θέλουμε να μετατρέψουμε ένα αντικείμενο σε αναπαράσταση JSON, το πεδίο της κλάσης με @Serializedname “categoryid” θα μεταφέρει την τιμή του πεδίου στο αντίστοιχο ζεύγος κλειδιού τιμής.

```java
public class Category {
    @SerializedName("categoryid")
    private int categoryid;
    @SerializedName("categoryname")
    private String categoryname;

    public Category() {}

    public Category(int categoryid, String categoryname) {
        this.categoryid = categoryid;
        this.categoryname = categoryname;
    }

    public int getCategoryid() { return categoryid; }

    public void setCategoryid(int categoryid) { this.categoryid = categoryid; }

    public String getCategoryname() { return categoryname; }

    public void setCategoryname(String categoryname) { this.categoryname = categoryname; }

    @Override
    public String toString() {
        return "Category{" +
                "categoryid=" + categoryid +
                ", categoryname='" + categoryname + '\'' +
                '}';
    }
}
```

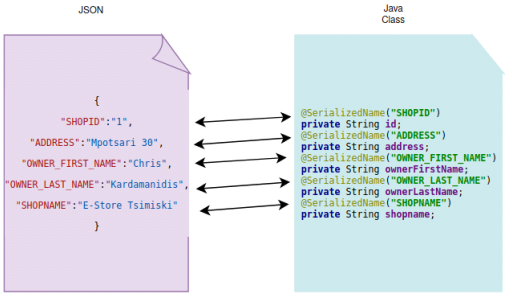
Σχήμα 26 Κλάση Category

4.7.3 Product Class

Για την αντιστοίχιση του κάθε Product από το αντίστοιχο Resource του Web Service υπάρχει η κλάση Product. Αυτή η κλάση θα περιλαμβάνει τα στοιχεία για κάθε ένα προϊόν. Τα πεδία τα οποία περιλαμβάνει είναι τα εξής: το id του προϊόντος το οποίο αντιστοιχίζεται με ένα κλειδί “productid” του json, ένα πεδίο title το οποίο κάνει match με το κλειδί “product_title”, ένα πεδίο cost που αντιστοιχίζεται με το κλειδί “productCost”και τέλος ένα πεδίο το οποίο περιέχει την εικόνα του προϊόντος με όνομα image_path και θα αντιστοιχίζεται με το πεδίο “productImage” του JSON. Τέλος έχει και άλλα 3 πεδία τα categoryid, quantity και bytesImage εκ των οποίων το quantity γίνεται map στο πεδίο του JSON με @SerializedName “quantity”.

4.7.4 Shop Class

Ένα άλλο σημείο του Web Service, επιστρέφει όλα τα Shops τα οποία περιλαμβάνονται σε αυτό το ηλεκτρονικό κατάστημα. Για αυτόν τον λόγο χρειάζεται αυτό το array από JSON objects να γίνουν mapping στα αντίστοιχα java object. Το object το οποίο θα κρατάει τα δεδομένα του Shop είναι Shop java object το οποίο παρουσιάζεται στο Σχήμα 27. Τα πεδία τα οποία περιλαμβάνει είναι το id του καταστήματος (πεδίο id) στο οποίο θα γίνει map ένα JSON με όνομα κλειδιού “SHOPID” SerializedName “SHOPID”, η διεύθυνση (address) στο οποίο θα γίνει map ένα JSON με όνομα κλειδιού “ADDRESS”, το μικρό όνομα του ιδιοκτήτη (ownerFirstName) σε ένα “OWNER_FIRST_NAME”, το επώνυμο (ownerLastName) σε ένα “OWNER_LAST_NAME”, καθώς και το όνομα του καταστήματος (shopname) σε ένα “SHOPNAME”. Η κλάση περιλαμβάνει έναν κενό δομητή για την δημιουργία ενός αντικειμένου με τις default τιμές, καθώς και έναν πλήρη δομητή για την δημιουργία αντικειμένου με τις τιμές που ορίζονται στην κλήση του. Η τροποποίηση των πεδίων και η πρόσβαση στις τιμές γίνεται μέσω των public μεθόδων get() και set().



Σχήμα 27 Mapping με την βιβλιοθήκη Gson

4.7.5 User Class

Επόμενη κλάση η οποία θα πρέπει να αναλυθεί είναι η κλάση User. Αυτή η κλάση είναι αρκετά γενική, για αυτόν τον λόγο θα πρέπει να γίνει extend από άλλες εξειδικεύσεις της. Η κλάση αυτή περιλαμβάνει τα βασικά στοιχεία ενός User. Τα πεδία τα οποία περιλαμβάνει ένας User είναι τα εξής: όνομα χρήστη (username), το πρώτο όνομα του χρήστη (firstname), το επώνυμο του χρήστη (lastName), ο κωδικός του χρήστη (password), το πρωτεύον email του (primaryemail), το δευτερεύον email (secondaryemail), το πεδίο διεύθυνσης (address), το πεδίο φύλλου (gender), το νούμερο του κινητού τηλεφώνου (mobilephone) και τέλος το νούμερο του σταθερού τηλεφώνου (homephone). Όλα αυτά τα πεδία έχουν το @Serialized annotation, με το οποίο κάθε ζεύγος κλειδιού και τιμής του json να αντιστοιχιστεί σε κάθε πεδίο. Το πεδίο username αντιστοιχίζεται με το κλειδί “username” του JSON, το πεδίο firstname με το κλειδί “firstname”, το πεδίο lastName με το κλειδί “lastname”, το πεδίο password με το κλειδί “password”, το πεδίο primaryemail με το κλειδί “primaryemail”, το secondaryemail πεδίο με το “secondaryemail”, το “address” πεδίο με το κλειδί “address”, το gender με το “gender” κλειδί, το mobilephone με το “mobilephone” και τέλος το homephone με το κλειδί “homephone”.

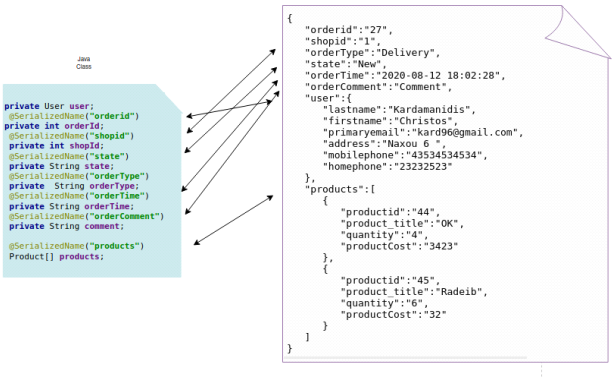
4.7.6 Employee class και διαχείριση κληρονομικότητας

Η κλάση η οποία χρησιμοποιείται για να αντιστοιχιστούν τα στοιχεία του υπαλλήλου από json σε java object είναι η Employee class. Η κλάση αυτή αποτελεί εξειδίκευση της User, καθώς οι υπάλληλοι είναι και χρήστες του συστήματος. Αυτό το οποίο χρειάζεται ιδιαίτερη αναφορά είναι το πώς έχοντας ένα JSON με τα στοιχεία ενός υπαλλήλου θα κάνουμε mapping όλα αυτά τα πεδία σε ένα αντικείμενο Employee. Η διαφοροποίηση με τις προηγούμενες περιπτώσεις είναι ότι υπάρχει χρήση κληρονομικότητας στα Java αντικείμενα. Η κλάση Employee έχει μόνο τα πεδία τα οποία την εξειδικεύουν και την διαφοροποιούν από την κλάση User. Τα πεδία, της Employee είναι τα παρακάτω: το πεδίο employeeid το οποίο αντιστοιχίζεται στο κλειδί “employeeid”, το πεδίο title στο κλειδί “title”, το πεδίο identity στο κλειδί “identity”, το πεδίο startOfContract στο κλειδί “startOfContract”, το πεδίο endOfContract στο κλειδί “endOfContract”, το πεδίο role στο κλειδί “role” και το πεδίο worksIn στο κλειδί “worksIn” το οποίο περιέχει το id του shop στο οποίο δουλεύει ο υπάλληλος. Το πρόβλημα το οποίο υπήρξε είναι πως από την μια το JSON έχει 17 πεδία για έναν Employee από τα οποία τα 7 είναι για την κλάση Employee και 10 για την κλάση User, οπότε τα στοιχεία του JSON πρέπει με κάποιο τρόπο να περάσουν και στην υπερκλάση. Η λύση υποστηρίζεται από την Gson βιβλιοθήκη όπου απλά πρέπει το κλειδί του Json να τοποθετηθεί σαν SerializedName στο πεδίο της υπερκλάσης, για να περάσει η τιμή του JSON.

4.7.7 Order class και ενθυλάκωση

Προχωρώντας στην επόμενη κλάση παρατηρούμε άλλο ένα σημείο το οποίο χρειάζεται ιδιαίτερη ανάλυση για την βιβλιοθήκη Gson. Η κλάση αυτή θα περιλαμβάνει τα στοιχεία της κάθε διαφορετικής παραγγελίας. Επειδή όμως η κλάση Order συνδέεται με άλλες κλάσεις, το mapping στα δεδομένα των κλάσεων περιπλέκεται. Για αρχή θα αναφερθούν τα απλά πεδία τα οποία γίνονται mapping. Τα πεδία τα οποία περιλαμβάνει είναι ένα αντικείμενο το οποίο προσδιορίζει μοναδικά την κάθε παραγγελία, το πεδίο με όνομα orderId και όνομα κλειδιού (SerializedName) “orderid”, το πεδίο shopId με όνομα κλειδιού το “shopId”, το πεδίο state και όνομα κλειδιού “state” το οποίο δείχνει σε τι κατάσταση είναι η παραγγελία, το πεδίο orderType με όνομα κλειδιού “orderType” και το πεδίο orderTime με όνομα κλειδιού “orderTime”. Ιδιαίτερο ενδιαφέρον υπάρχει στα άλλα 2 πεδία τα οποία είναι τύπου κλάσης. Το ένα πεδίο είναι το πεδίο τύπου User και είναι το αντικείμενο με τα στοιχεία του πελάτη και το άλλο είναι ένας πίνακας από προϊόντα, δηλαδή τύπου Product[].

Εν αντιθέσει με τα άλλα πεδία που ήταν αρχέτυποι (primitive types) εδώ μιλάμε για ενθυλάκωση άλλων αντικειμένων μέσα στα οποία πρέπει να γίνει το mapping των δεδομένων. Δηλαδή θα πρέπει να περαστούν τα δεδομένα όχι μόνο από το JSON στα πεδία του Java αντικειμένου, αλλά και στα πεδία του αντικειμένου που ενθυλακώνει το Java αντικείμενο. Αυτό γίνεται με την ακριβή απεικόνιση των αντικειμένων του JSON στο όνομα του πεδίου της μεταβλητής. Είναι γνωστό ότι τα ζεύγη κλειδιού τιμής στο JSON αρχείο μπορεί στις τιμές να περιλαμβάνει Strings, numbers, booleans, object, arrays. Στην προκειμένη περίπτωση όταν έρχεται ένα JSON αρχείο μιας order, περιλαμβάνει στο κλειδί user ένα άλλο JSON με τα στοιχεία του user στα οποία αυτόματα θα γίνει το mapping των πεδίων. Το ίδιο συμβαίνει και με τον πίνακα των προϊόντων όπως εμφανίζεται στο Σχήμα 28. Στο πεδίο products του JSON υπάρχει ένας πίνακας από products όπου αντιστοιχίζονται με τον πίνακα των java αντικειμένων.



Σχήμα 28 Mapping σε αντικείμενα με ενθυλάκωση

4.8 Notification Service

4.8.1 BootReceiver class

Όπως έχει ήδη αναφερθεί στο manifest αρχείο, υπάρχει το δικαίωμα RECEIVE_BOOT_COMPLETED. Χάρη σε αυτό το δικαίωμα η εφαρμογή, μπορεί να ανιχνεύει πότε έγινε boot το κινητό. Γενικά οι εφαρμογές Android μπορούν να στείλουν ή να λάβουν μηνύματα broadcast από άλλες εφαρμογές ή το σύστημα του Android. Αυτά τα μηνύματα Broadcast στέλνονται όταν συμβαίνει ένα γεγονός το οποίο ενδιαφέρει την εφαρμογή μας. Τέτοια μηνύματα του λειτουργικού είναι όταν η συσκευή αρχίζει να φορτίζει ή ακόμα και ότι το κινητό μόλις έκανε boot.

Γενικά τα broadcasts μηνύματα μπορούν να χρησιμοποιηθούν ως τρόπος επικοινωνίας μεταξύ των εφαρμογών. Το ιδιαίτερο με αυτήν την επικοινωνία είναι ότι διεξάγεται χωρίς την παρέμβαση του χρήστη. Ιδιαίτερη έμφαση είναι ανάγκη να δοθεί στο ότι δεν πρέπει να στέλνονται πολλά τέτοια μηνύματα που εκτελούν διεργασίες στο παρασκήνιο διότι αυτό θα έχει ως αποτέλεσμα να μειωθεί η απόδοση του συστήματος. Τα μηνύματα αυτά στέλνονται αυτόματα από το λειτουργικό σύστημα όταν διάφορα τέτοια γεγονότα συμβαίνουν. Τέτοια γεγονότα μπορεί να είναι η μετάβαση σε λειτουργία πτήσης ή και το αντίστροφο.

Τα system broadcasts είναι πακεταρισμένα σε ένα αντικείμενο Intent. Στην πορεία, από το action string του αντικειμένου μπορούμε να αντιληφθούμε και να ελέγξουμε πιο γεγονός είναι. Μαζί με τα Intent αντικείμενα συνήθως έρχονται και κάποιες έξτρα πληροφορίες οι οποίες πακεταρισμένες σαν Bundle αντικείμενα. Για παράδειγμα όταν η εφαρμογή λαμβάνει ένα Intent για το γεγονός της λειτουργίας πτήσης, ο προγραμματιστής μέσω του Bundle μπορεί να ελέγξει μια boolean τιμή αν είναι true ή false για να δει αν η λειτουργία πτήσης είναι on ή off. Η ανίχνευση του γεγονότος γίνεται με 2 τρόπους. Ο πρώτος είναι μέσω του manifest και ο δεύτερος είναι οι context-registered receivers. Για το πως θα γίνει η λήψη του μηνύματος στην πράξη στην πτυχιακή θα αναλυθεί παρακάτω στο Σχήμα 29.

Για αρχή πρέπει να υπάρξει μια κλάση BootReceiver η οποία θα δηλωθεί στο αρχείο manifest. Επίσης θα δηλωθεί ένα receiver element με name .BootReceiver, δηλαδή το όνομα της κλάσης. Μέσα στο intent filter ορίζουμε τα διάφορα actions τα οποία θα ανιχνεύσει η εφαρμογή. Τα actions τα οποία είχαν δηλωθεί είναι τα “android.intent.action.BOOT_COMPLETE” και ““android.intent.action.QUICKBOOT_POWER”, με αυτά ανιχνεύει αν έκανε boot το κινητό. Αυτές οι δηλώσεις γίνονται μέσα στο manifest αρχείο xml στο <application> element. Το αμέσως επόμενο είναι η κλάση BootReceiver. Για αρχή θα πρέπει να κάνει extend τον Broadcast Receiver για να μπορέσει κληρονομώντας τις λειτουργίες του να ανιχνεύσει τα broadcast μηνύματα. Όπως φαίνεται παρακάτω η κλάση περιλαμβάνει ένα κενό δομητή για την κατασκευή του αντικειμένου και επίσης κάνει @Override την μέθοδο onReceive η οποία υπάρχει στην κλάση BroadcastReceiver. Override σημαίνει ότι αυτή η κλάση ξαναφτιάχνει μια μέθοδο που υπάρχει στην υπερκλάση της αλλά με διαφορετικό κώδικα, επιτελώντας διαφορετική λειτουργία. Η μέθοδος public void onReceive(Context,Intent) εκτελείται όταν στέλνεται το broadcast μήνυμα από το λειτουργικό σύστημα. Το intent αντικείμενο είναι αυτό το οποίο αναφέρθηκε πριν και μεταφέρει τις πληροφορίες του μηνύματος. Αμέσως μετά ελέγχουμε αν το action το οποίο ελήφθη με την μορφή Intent είναι το “android.intent.action.BOOT_COMPLETE”, ελέγχοντας την τιμή που επιστρέφει η μέθοδος getAction του object. Στην πορεία εφόσον έχουμε λάβει το σωστό μήνυμα μπορούμε να κάνουμε ότι ενέργεια θέλουμε on boot.

Η ενέργεια η οποία θα επιλέξουμε να κάνουμε είναι να ξεκινήσουμε το Notification service. Για να γίνει η εκκίνηση του service, φτιάχνουμε ένα Intent περνώντας σαν παραμέτρους το context το οποίο έχουμε ήδη διαθέσιμο από την μέθοδο onReceive. Τέλος καλούμε την μέθοδο startService του context αντικειμένου η οποία θα ξεκινάει το service με Intent για το NotificationService class.

```java
package com.example.eshopproject;

import ...


public class BootReceiver extends BroadcastReceiver {
    public BootReceiver() { }
    @Override
    public void onReceive(Context context, Intent intent) {

        Log.d( tag: "onReceive",  msg: "started");

        if ("android.intent.action.BOOT_COMPLETED".equals(intent.getAction())) {

            Log.d( tag: "Boot completed",  msg: "code received, starting service");
            Intent serviceIntent = new Intent(context, NotificationService.class);
            context.startService(serviceIntent);

        }
    }
}
```

Σχήμα 29 Boot Receiver

4.8.2 Υλοποίηση Notification Service και HTTP μηνύματος

Για τον έλεγχο της ύπαρξης των διάφορων παραγγελιών ο υπάλληλος μπορεί να περιηγηθεί μέσω της εφαρμογής, δηλαδή να πάει στο αντίστοιχο Fragment το οποίο αναφέρονται όλες οι καινούργιες παραγγελίες. Στην περίπτωση όμως που ο υπάλληλος δεν έχει ανοιχτή την εφαρμογή, θα πρέπει κάπως να ελέγχεται αν υπάρχουν καινούργιες παραγγελίες στον server. Για το κομμάτι της ειδοποίησης του χρήστη, είναι ξεκάθαρο πως χρειάζεται ένα notification για τον χρήστη. Το κύριο κομμάτι το οποίο έπρεπε να λυθεί, είναι ο τρόπος με τον οποίο θα γίνεται η εκκίνηση αυτού του ελέγχου, αν η εφαρμογή δεν έχει ξεκινήσει ακόμα.

Η λύση η οποία προτάθηκε είναι η ύπαρξη ενός service. Ως service αναφερόμαστε σε ένα κομμάτι της εφαρμογής, το οποίο εκτελείται στο παρασκήνιο. Συνήθως τέτοια services περιλαμβάνουν διαδικασίες οι οποίες να κρατήσουν πολλή ώρα ή να είναι πολλές στον αριθμό. Τέτοιες υπηρεσίες θα μπορούσαν να είναι το κατέβασμα ενός αρχείου ή οι διαδοχικοί έλεγχοι στον server. Σημαντική λεπτομέρεια είναι ότι ένα service δεν δημιουργεί δικό του νήμα (thread), αλλά εκτελείται στο main thread. Για αυτόν τον λόγο θα πρέπει διάφορες διεργασίες οι οποίες μπορεί να μπλοκάρουν, να μην εκτελούνται στο main thread.

Στο Android υπάρχουν τρία είδη από services, τα foreground, τα background και τα bound. Ως foreground service αναφέρονται εκείνα τα οποία εκτελούν κάποια διεργασία η οποία μπορεί να παρατηρηθεί από τον χρήστη. Το δεύτερο είδος service είναι το background service το οποίο δεν είναι δυνατό να παρατηρηθεί άμεσα από τον χρήστη. Το τρίτο και τελευταίο είδος service λέγεται bound και είναι όταν ένα component μιας εφαρμογής κάνει bind αυτό το service με την χρήση της μεθόδου bindService(). Τα bound service τρέχει μόνο όσο ένα component έχει κάνει bind αυτό το service. Αυτό το service επιτρέπει την επικοινωνία με αυτό με την χρήση requests και την λήψη αποτελεσμάτων, αλλά και επικοινωνία μέσω διάφορων διεργασιών, δηλαδή IPC (InterProccess communicaton). Σημαντικό είναι πως αυτό το service μπορούν να κάνουν bind διάφορα components, αλλά όταν δεν υπάρχει κάποιο, το service, καταστρέφεται. Σε αυτό το σημείο πρέπει να αναφερθεί η διαφορά των services και των threads. Το service τρέχει ανεξάρτητα από το activity της εκάστοτε εφαρμογής. Με την χρήση του service μπορεί να εκτελεστούν οι βαριές διεργασίες ή διεργασίες δικτύου. Τα threads πρέπει να χρησιμοποιούνται όταν έχουν κάποια σχέση με το αντίστοιχο activity της εφαρμογής.

Στην πτυχιακή υπήρχε η ανάγκη για τον συνεχή έλεγχο για νέες παραγγελίες. Η λύση είναι το Notification Service. Το συγκεκριμένο service κάνει συνεχείς κλήσεις στο Web Service για να ελέγξει αν υπάρχουν νέες παραγγελίες. Αν υπάρχουν θα πρέπει να κάνει μια άλλη κλήση για να επιστρέψει τις

νέες παραγγελίες. Με την κλήση στο "/api/v1/Customers/Orders/NewOrdersCheck ελέγχει πόσες είναι οι νέες παραγγελίες και με το "/api/v1/Customers/Orders/New" επιστρέφεται σε μορφή JSON ο αριθμός νέων παραγγελιών. Η δομή της κλάσης NotificationService αναλύεται παρακάτω και περιέχει τα παρακάτω πεδία και μεθόδους που φαίνονται στο Σχήμα 30.

Για αρχή έχει 2 πεδία τα οποία θα χρειαστούν για την δημιουργία του Notification. Το πρώτο είναι το κανάλι (CHANNEL_ID) δηλαδή ένας κωδικός ο οποίος θα ορίζει σε ποιο κανάλι θα στέλνεται το notification. Το δεύτερο πεδίο θα περιέχει τον Builder του notification με τον οποίο θα ορίζονται οι παράμετροι του notification. Οι μέθοδοι οι οποίοι έχουν υλοποιηθεί είναι οι onCreate(), onStartCommand(), onBind(), createNotification(), startCheckingForOrders(). Η μέθοδος onBind() που καλείται όταν ένα component καλεί το service. Αμέσως επόμενη είναι onStartCommand. Αυτή η μέθοδος καλείται με την έναρξη του service και επιστρέφει ένα code. Ιδιαίτερη αναφορά πρέπει να γίνει στον κωδικό "START_STICKY" το οποίο επιστρέφεται. Αυτός ο κωδικός ενημερώνει το λειτουργικό σύστημα του Android αν γίνει kill αυτό το service, να το ξαναδημιουργήσει από την αρχή. Σε μια άλλη περίπτωση που χρησιμοποιούταν το "START_NOT_STICKY", το λειτουργικό σύστημα ενημερώνεται να μην ξαναδημιουργήσει το service.

Η τελευταία μέθοδος η οποία αποτελεί την αρχή του κώδικα του e-shop είναι η onCreate(). Αυτή η μέθοδος καλείται με την δημιουργία του service και θα καλέσει την μέθοδο startCheckingForOrders(). Αναφορά πρέπει να γίνει στο ότι δεν επιτρέπονται διεργασίες δικτύου στο main thread. Για αυτόν τον λόγο δημιουργούμε ένα νέο Thread αντικείμενο, στο οποίο όταν τρέχει θα καλεί την μέθοδο που αναφέραμε και για αυτό κάνουμε implement την μέθοδο run του Thread object. Τέλος, απλά κάνουμε start το καινούργιο thread.

```
public class NotificationService extends Service {
    private static final String CHANNEL_ID = "11";
    private NotificationCompat.Builder notification_builder;
    @Nullable
    @Override
    public IBinder onBind(Intent intent) { return null; }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Log.d( tag: "Notification service:",  msg: " started");
        return START_STICKY;
    }

    @Override
    public void onCreate() {
        Log.d( tag: "Notification:",  msg: " created");
        Thread thread = new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    startCheckingForOrders();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                    Log.d( tag: "Error",  msg: " in checking for new orders");
                }
            }
        });
        thread.start();
    }
    private void createNotification(int id, String orderType, String orderTime) {...}
    private void startCheckingForOrders() throws InterruptedException {...}
}
```

Σχήμα 30 Δομή κλάσηςNotification Service

Η μέθοδος startCheckingForOrders (Σχήμα 31), έχει την εξής λειτουργία. Για αρχή, όσο είναι ζωντανό το νήμα το οποίο ξεκινήσαμε παραπάνω, θα κάνει συνέχεια ελέγχους στο Web Service για νέες παραγγελίες. Το αν είναι ζωντανό ακόμα το thread το παίρνουμε από την μέθοδο isAlive() του αντικειμένου του thread που επιστρέφει η μέθοδος curenntThread() η οποία ανήκει στην κλάση Thread. Αμέσως μετά με την χρήση της μεθόδου κάνουμε sleep το thread για 5 δευτερόλεπτα. Αυτό χρειάζεται γιατί θέλουμε από τον έναν έλεγχο μέχρι τον επόμενο έλεγχο να μεσολαβεί ένα χρονικό

διάστημα 5 δευτερολέπτων. Αυτό γιατί ο server θα απασχολείται πάρα πολύ για αιτήματα που δεν είναι κρίσιμα να υπάρχει χρονική ακρίβεια. Στην περίπτωση που γινόταν πιο συχνά κλήσεις στο Web Service, θα υπήρχε θέμα με την επεξεργαστική ισχύ του κινητού αλλά και του server. Επίσης, θα υπήρχε και μεγάλη κίνηση στο δίκτυο χωρίς να υπάρχει λόγος. Στην συνέχεια δημιουργούμε ένα αντικείμενο RestClient() το οποίο θα μας χρησιμεύσει για τα HTTP αιτήματα στο Web Service. Το endpoint στο οποίο θα γίνει η κλήση είναι το "/api/v1/Customers/Orders/NewOrdersCheck". Η IP 192.168.1.70 είναι του Web Server στον οποίο βρίσκεται το Web Service το οποίο θα κάνει την επικοινωνία με την βάση δεδομένων.

Σύμφωνα με τον κώδικα PHP θα επιστρέψει σε JSON μορφή ένα αποτέλεσμα της μορφής "{"newOrders" : <number>}", όπου είναι ο αριθμός των νέων παραγγελιών. Αυτό το αποτέλεσμα το αποθηκεύουμε σε αντικείμενο JSONObject. Στην συνέχεια με την μέθοδο get("newOrders") όπου newOrders είναι το κλειδί από το JSON. Στην συνέχεια ελέγχουμε αν είναι παραπάνω από 0 οι καινούργιες παραγγελίες θα πάμε να τις πάρουμε με την ένα GET αίτημα στο endpoint "/api/v1/Customers/Orders/New". Αυτό επιστρέφει σε JSON όλες τις νέες παραγγελίες και τα προϊόντα που περιέχει η κάθε παραγγελία. Όλες αυτές οι παραγγελίες επιστρέφονται σε ένα JSONArray αντικείμενο. Το αμέσως επόμενο και κομβικό σημείο για την μετατροπή των δεδομένων είναι το αντικείμενο Gson. Με την μέθοδο fromJson(String, ClassName[].class) μπορούμε ένα array από JSON παραγγελίες σε μορφή να μετατραπεί σε έναν πίνακα από Order Java αντικείμενα. Με την μετατροπή αυτή έχουμε σε μορφή αντικειμένων πλέον όλα τα δεδομένα των παραγγελιών αλλά και των προϊόντων τους. Εφόσον έχει γίνει αυτή η μετατροπή, μπορούμε να πλέον για κάθε μια παραγγελία να δημιουργήσουμε ένα notification κάτι το οποίο θα γίνει με την χρήση της μεθόδου createNotification. Τέλος όλος αυτός ο κώδικας πρέπει να είναι μέσα σε try-catch για να πιάσει ένα πιθανό JSONException ή ένα IOException. Το πρώτο μπορεί να προκύψει κατά την μετατροπή των δεδομένων από String σε JSONArray ενώ το δεύτερο μπορεί να προκύψει στην περίπτωση που ο client, δηλαδή η Android εφαρμογή δεν μπορέσει να συνδεθεί στο Web Service.

```
private void startCheckingForOrders() throws InterruptedException {
    while (Thread.currentThread().isAlive()) {
        Thread.sleep( millis: 5000);
        RestClient restClient = new RestClient();
        restClient.context = getApplicationContext();
        try {
            JSONObject jsonObject = restClient.doGetRequestSingleObject(
                    url: "https://192.168.1.70/ptyxiaki/index.php/api/v1/Customers/Orders/NewOrdersCheck");
            int newOrders = Integer.parseInt(jsonObject.get("newOrders").toString());

            if (newOrders >0) {
                Gson gson = new Gson();
                JSONArray orders = restClient.doGetRequest( url: "" +
                        "https://192.168.1.70/ptyxiaki/index.php/api/v1/Customers/Orders/New");
                final Order[] ordersArray = gson.fromJson(orders.toString(), Order[].class);
                for (int i = 0; i < newOrders; i++) {
                    createNotification(i, ordersArray[i].getOrderType() ,ordersArray[i].getOrderTime());
                }
            }
        } catch (JSONException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Σχήμα 31 Μέθοδος ελέγχου για νέες παραγγελίες

Για την δημιουργία του notification χρειάζεται προσοχή στην έκδοση του Android. Στις εκδόσεις 8.0 και νεότερες πρέπει να η εφαρμογή να κάνει register το δικό της κανάλι στο οποίο θα δημιουργεί notifications. Για αρχή φτιάχνουμε ένα NotificationManager αντικείμενο το οποίο θα μας βοηθήσει στην δημιουργία του notification όπως φαίνεται στο Σχήμα 32. Με την χρήση της μεθόδου getSystemService(Service.NOTIFICATION_SERVICE) παίρνουμε το service. Στην συνέχεια

ορίζουμε τα στοιχεία του καναλιού επικοινωνίας. Πρώτα ορίζουμε το id το οποίο θα προσδιορίζει μοναδικά το συγκεκριμένο κανάλι. Μετά ορίζουμε τις παραμέτρους name και description που χρειάζονται για το κανάλι. Πολύ σημαντικό είναι το importance, μια παράμετρος η οποία θα οριστεί low. Για τα notifications υπάρχουν τα εξής επίπεδα. Το Urgent το οποίο κάνει έναν ήχο και εμφανίζεται σαν ένα heads-up notification. Αμέσως επόμενο είναι το “High” το οποίο παράγει ήχο με την ειδοποίηση. Αμέσως επόμενο είναι το Medium το οποίο δεν παράγει κάποιον ήχο και τέλος είναι το low. Με όλες τις παραμέτρους που αναφέραμε δημιουργούμε ένα αντικείμενο NotificationChannel στο οποίο τις κάνουμε set. Επίσης στο κανάλι αυτό ορίζουμε σαν χρώμα το μπλε και το περνάμε σαν παράμετρο στην μέθοδο createNotificationChannel του manager.

Στην συνέχεια δημιουργούμε ένα αντικείμενο Intent για να ορίσουμε σε ποιό activity θα μεταβούμε αν κάνουμε κλικ στην ειδοποίηση. Το activity το οποίο ορίζεται είναι το LoginActivity, γιατί ο χρήστης θα πρέπει να κάνει σύνδεση για να κάνει ενέργειες πάνω στις συγκεκριμένες παραγγελίες. Στην συνέχεια δημιουργούμε ένα PendingIntent object. Αυτό το αντικείμενο είναι ένα token που δίνεις σε μια ξένη εφαρμογή όπως στην περίπτωση μας τον NotificationManager και δίνεις τα δικαιώματα στην ξένη εφαρμογή να εκτελεί ένα προκαθορισμένο κομμάτι κώδικα. Η διαφορά με το Intent είναι πως με το Intent αντικείμενο αυτό το κομμάτι κώδικα εκτελείται με τα δικαιώματα της ξένης εφαρμογής. Βασικές παράμετροι είναι το request code το οποίο χρησιμοποιείται για να πάρουμε ξανά το ίδιο PendingIntent στο μέλλον, το intent που ορίσαμε παραπάνω, αλλά και το FLAG_UPDATE_CURRENT το οποίο ορίζει πως αν το συγκεκριμένο PendingIntent υπάρχει ήδη θα πρέπει να το κρατήσει, αλλά θα πρέπει να κάνει update τα καινούρια δεδομένα.

Για την δημιουργία του Notification χρησιμοποιείται ο NotificationCompat.Builder στον οποίο ορίζουμε τις παραμέτρους του notification. Για αρχή ορίζουμε το notificationid, στην πορεία ορίζουμε το εικονίδιο το οποίο είναι ένα πακέτο προϊόντος, στην συνέχεια ορίζουμε τον αύξων αριθμό id της παραγγελίας. Στην πορεία ορίζουμε το κείμενο που είναι της μορφής “"One new" + orderType + " order is shown still " + orderTime” καθώς και τον ήχο του notification. Τέλος για την μετάβαση στο άλλο activity περνάμε σαν παράμετρο το Pending Intent στην μέθοδο setContentIntent για να μεταβούμε στο LoginActivity αλλά και θέτουμε true το setOngoing και κάνουμε build(). Εφόσον έχει γίνει build απλά καλούμε την μέθοδο notify του notificationManager.

```
private void createNotification(int id, String orderType, String orderTime) {
    NotificationManager notificationManager = (NotificationManager) getSystemService(Service.NOTIFICATION_SERVICE);
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        String chanel_id = CHANNEL_ID;
        CharSequence name = "Notification";
        String description = "Notification Description";
        int importance = NotificationManager.IMPORTANCE_LOW;
        NotificationChannel mChannel = new NotificationChannel(chanel_id, name, importance);
        mChannel.setDescription(description);
        mChannel.enableLights(true);
        mChannel.setLightColor(Color.BLUE);
        notificationManager.createNotificationChannel(mChannel);
        notification_builder = new NotificationCompat.Builder( context: this, chanel_id);
    } else {
        notification_builder = new NotificationCompat.Builder( context: this);
    }
    Intent intent = new Intent( packageContext: this, LoginActivity.class);
    PendingIntent pendingIntent = PendingIntent.getActivity( context: this,  requestCode: 0,
            intent, PendingIntent.FLAG_UPDATE_CURRENT);

    Notification notification = new NotificationCompat.Builder(getBaseContext(), channelId: "notification_id")
            .setSmallIcon(R.drawable.ic_icons8_box).setContentTitle("New Order " + id)
            .setContentText("One new" + orderType + " order is shown still " + orderTime)
            .setDefaults(NotificationCompat.DEFAULT_SOUND).setContentIntent(pendingIntent)
            .setOngoing(true).setAutoCancel(true).build();
    notificationManager.notify(id, notification);
}
```

Σχήμα 32 Δημιουργία Notification

4.9 LoginActivity, Shared Preferences and Animations

Το αμέσως επόμενο activity το οποίο χρειάζεται να αναφερθεί είναι το LoginActivity. Το συγκεκριμένο activity είναι υπεύθυνο για την παρουσίαση της διεπιφάνειας σύνδεσης ενός χρήστη στο ηλεκτρονικό μας κατάστημα.

Η διεπιφάνεια χρήστη περιλαμβάνει 2 βασικά πεδία τα οποία χρειάζονται για την σύνδεση ενός χρήστη στο ηλεκτρονικό κατάστημα. Το ένα είναι το username και το άλλο είναι το password. Το συγκεκριμένο activity αποτελείται από 2 κομμάτια. Το πρώτο είναι το xml activity_login το οποίο περιλαμβάνει την σχεδίαση της διεπιφάνειας (Σχήμα 34). Για αρχή το συγκεκριμένο αρχείο αποτελείται από ένα root element RelativeLayout το οποίο επιτρέπει την τοποθέτηση των διάφορων views το ένα σε σχέση με το άλλο. Το element αυτό περιέχει 3 attributes, το "layout_width" το οποίο περιλαμβάνει το πλάτος του view, το "layout_height" καθώς και ένα background με το xml "back_gradient". Τα πρώτα 2 έχουν την τιμή "match_parent" το οποίο δείχνει ότι έχουν πλάτος και ύψος, όσο το parent view. Στην συνέχεια, μέσα σε αυτό το layout υπάρχει ένα LinearLayout. Αυτό το layout έχει την ιδιότητα ότι εμφανίζει στην σειρά τα διάφορα views που τοποθετούνται μέσα σε αυτό. Το αν τα view εμφανίζονται οριζόντια ή κάθετα ορίζεται από το attribute orientation το οποίο έχει την τιμή vertical. Επίσης σημαντικό είναι αυτό το view να εμφανιστεί στο κέντρο της οθόνης, για αυτό και περιέχει τα attributes layout_centerHorizontal και layout_centerVertical τα οποία έχουν την τιμή true.

Αμέσως μέσα υπάρχει ένα EditText το οποίο θα δεχτεί το όνομα χρήστη. Και σε αυτό ορίζουμε ένα background το xml "textbox_back"για να δώσουμε style στο textbox. Επίσης, μετά όσον αφορά τον σχεδιασμό ορίζουμε ένα drawableLeft attribute το οποίο θα παίρνει σαν παράμετρο ένα εικονίδιο χρήστη. Αμέσως μετά ορίζουμε και ένα hint attribute το οποίο εμφανίζει σαν προ εμφανιζόμενο το όνομα του πεδίου που πρέπει να εισάγει ο χρήστης δηλαδή "Username" καθώς και ορίζουμε και το σε μαύρο το χρώμα του hint με το attribute textColorHint.

Στην συνέχεια υπάρχει ένα LinearLayout με horizontal orientation το οποίο θα περιλαμβάνει ένα EditText το οποίο θα περιλαμβάνει τον κωδικό του χρήστη, αλλά και ένα ImageButton το οποίο με το που θα επιλέγεται, θα εμφανίζει το password. Αρχικά ορίζουμε διαστάσεις layout_width και layout_height. Αμέσως μετά ορίζουμε σαν hint την λέξη "Password" και σαν χρώμα textColorHint το ίδιο χρώμα που μπήκε και στο EditText του password. Επίσης σημαντικό για την αλληλεπίδραση με τον χρήστη το drawableLeft εικονίδιο το οποίο ορίζεται και είναι μια κλειδαριά. Επίσης μια διαφοροποίηση είναι το διαφορετικό inputType που είναι "textPassword" και εμφανίζει ότι πληκτρολογείται από τον χρήστη σε μορφή κωδικού. Το δεύτερο στοιχείο μέσα σε αυτό το LinearLayout είναι το ImageButton εμφάνισης πληκτρολογημένου κωδικού. Για αρχή ορίζουμε τις διαστάσεις layout_width και layout_height και ορίζουμε και σαν src το εικονίδιο "ic_remove_red_eye_black_24dp" το οποίο βρίσκεται στον φάκελο drawable και είναι ένα εικονίδιο με ένα μάτι. Τέλος ορίζουμε σαν background ένα χρώμα ώστε να είναι διαφανές και να φαίνεται μόνο το εικονίδιο.

Τέλος , υπάρχει ένα Button View το οποίο θα χρησιμοποιεί ο χρήστης για να κάνει την σύνδεση. Τα "layout_height" και "layout_width" έχουν τιμές wrap_content για να διαστάσεις όσες και το περιεχόμενο τους. Σαν text υπάρχει ένα "Login" και ορίζουμε και σαν textColor attribute το μαύρο. Τέλος ορίζω και σαν background το xml login_button_back. Σε περίπτωση αποτυχίας, υπάρχει και ένα TextView το οποίο θα εμφανίζει το μήνυμα σφάλματος της σύνδεσης. Σκοπίμως για όλα τα προηγούμενα view δεν αναφέρθηκαν τα attribute ID που έχουν. Αυτό γιατί θα πρέπει να η εξήγηση τους είναι άμεσα σχετική με τον κώδικα ο οποίος έχει πρόσβαση σε αυτά τα view για να τα τροποποιήσει.

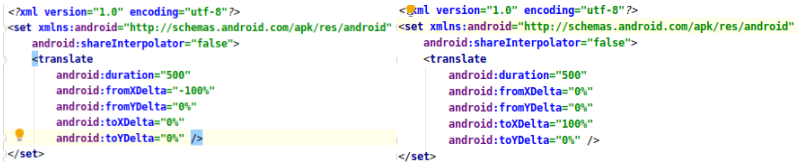
Η κλάση η οποία περιλαμβάνει την λειτουργικότητα του Login είναι το LoginActivity. Αυτό περιλαμβάνει 4 πεδία και 3 μεθόδους και έναν δομητή. Τα πεδία είναι ένα αντικείμενο τύπου LoginActivity που κρατάει το υπάρχον αντικείμενο της κλάσης. Αμέσως μετά περιλαμβάνει μια Button μεταβλητή η οποία θα κρατήσει το αντικείμενο του view για το κουμπί της σύνδεσης, ένα

ImageButton το οποίο θα κρατάει το αντικείμενο του view που είναι υπεύθυνο με την πίεση του να εμφανίζει τον κωδικό και τέλος ένα passwordEt τύπου EditText που θα έχουμε το αντικείμενο με το πεδίο που θα πληκτρολογηθεί ο κωδικός χρήστη. Αμέσως μετά υπάρχει ένα κενός δομητής τη κλάσης LoginActivity που φτιάχνει το αντικείμενο του Activity και οι μέθοδοι doLogin και performLogin.

Για αρχή θα αναλυθεί η μέθοδος onCreate η οποία αποτελεί το σημείο εκκίνησης. Πρώτα από όλα με την μέθοδο setContentView και σαν παράμετρο το R.layout.activity_login ορίζουμε ποιο είναι το xml του java Activity. Αμέσως μετά με την μέθοδο findViewById και περνόντας τα σαν παράμετρο τα id “button_login”, “show_password_button” και “passwordET” βρίσκοντας το κουμπί σύνδεσης, το κουμπί εμφάνισης του κωδικού και το πεδίο του κωδικού βρίσκουμε τα αντίστοιχα views και τα μετατρέπουμε σε αντικείμενα java για να έχουμε πρόσβαση σε αυτά. Στο κουμπί εμφάνισης κωδικού ορίζουμε με την μέθοδο setOnClickListener ορίζουμε ένα αντικείμενο OnClickListener και στην μέθοδο onClick χρησιμοποιούμε την μέθοδο setTransformationMethod και περνάμε σαν παράμετρο null για να αλλάξω δυναμικά τα attribute του view και να εμφανίζει το password. Αμέσως μετά ορίζω ένα scheduled task το οποίο μες την μέθοδο java.util.Timer().schedule() στην ποία περνάω java.util.TimerTask, δηλαδή ένα κομμάτι κώδικα το οποίο θα εκτελεστεί μετά από ένα delay. Το delay είναι η δεύτερη παράμετρος η οποία είναι 1000 ms, δηλαδή για 1 δευτερόλεπτο. Το κομμάτι το οποίο θα εκτελεστεί βρίσκεται στην μέθοδο run() όπου καλούμε την μέθοδο setTransformationMethod για το πεδίο του κωδικού και του περνάμε ένα αντικείμενο PasswordTransformationMethod() για να αποκτήσει ξανά την μορφή κωδικού το EditText.

Το δεύτερο onClickListener θα τον ορίσουμε στο αντικείμενο του κουμπιού σύνδεσης. Ο κώδικας που εκτελείται ορίζεται στην onClick μέθοδο. Για αρχή παίρνουμε στις μεταβλητές username και password τις τιμές από το view για να γίνει η σύνδεση. Σε αυτό το σημείο πρέπει να αναφερθεί ότι ανάλογα από το password και username γίνεται login σε διαφορετικό χρήστη και διαφορετική διεπιφάνεια. Οι managers συνδέονται με usernames τα οποία είναι της μορφής manager<Username> και οι administrators της μορφής administrator<Username>, άρα έχουμε δύο διαφορετικά είδη login τα οποία θα χειρίζονται από την μεταβλητή loginType. Στην συνέχεια από την μεταβλητή password, ελέγχω αν ξεκινάει από “admin” με την μέθοδο startsWith και αν ξεκινάει από αυτό ορίζω σαν loginType “admin”. Την ίδια διαδικασία κάνω και αν ξεκινάει από την λέξη manager. Αν δεν ξεκινάει από κανένα από τα 2, θα εμφανιστεί σε Toast ένα μήνυμα “Login failed”. Το Toast εμφανίζεται με την μέθοδο makeText της κλάσης Toast και παίρνει τρις παραμέτρους, το Context, την συμβολοσειρά που αναφέραμε και την διάρκεια η οποία είναι Toast.LENGTH_SHORT. Αμέσως μετά καλούμε την μέθοδο show η οποία θα το εμφανίσει. Στην συνέχεια στην μεταβλητή boolean login_successfull επιστρέφουμε το αποτέλεσμα της μεθόδου doLogin η οποία θα αναλάβει την σύνδεση με το Web Service. Αν η μέθοδος επιστρέψει false, θα επιστρέψουμε “Login not successful”.

Στην περίπτωση που η σύνδεση είναι επιτυχής, ανάλογα με το loginType ξεκινάμε διαφορετικό activity. Το ξεκίνημα του Activity γίνεται με την μέθοδο startActivity που παίρνει σαν παράμετρο από ένα διαφορετικό Intent ανάλογα με το Activity που θα ξεκινήσει. Αν έχουμε admin loginType το Intent περιέχει μέσα το this, δηλαδή το τρέχον αντικείμενο και το AdministratorManagement.class ή το ManagerActivity.class. Επίσης πριν κλείσουμε το συγκεκριμένο activity με την μέθοδο finish(), ορίζουμε με την μέθοδο overridePendingTransition ορίζουμε 2 παραμέτρους left_to_right με το αντίστοιχο animation που υπάρχει για να πετύχουμε την μετάβαση το animation από το ένα activity στο άλλο (Σχήμα 33). Τα animations είναι δύο. Και τα 2 έχουν παρόμοια μορφή. Και τα δυο αποτελούνται από ένα set element το οποίο είναι ο container για άλλα animation elements. Αμέσως μετά ορίζω ένα <translate> element το οποίο ορίζει μια κάθετη ή οριζόντια κίνηση. Στην συνέχεια ορίζω την διάρκεια (duration) και τις αρχικές και τελικές μετατοπίσεις του animation στον άξονα X’X και Y’Y.



Σχήμα 33 Δημιουργία animation

Για να ολοκληρώσουμε με την σύνδεση πρέπει να μιλήσουμε για την μέθοδο do_login() η οποία παίρνει σαν παράμετρο username, password και loginType. Μέσα σε αυτήν την μέθοδο δημιουργούμε ένα καινούργιο Thread με την κλάση HandlerThread και ορίζουμε την μέθοδο run() η οποία θα περιλαμβάνει τον κώδικα εκτέλεσης στο καινούργιο Thread. Αυτό το οποίο θα κάνουμε είναι να ελέγχουμε το loginType και να περνάμε στην μέθοδο performLogin() τα username και password ανάλογα το loginType. Επίσης σημαντικό κομμάτι στον κώδικα είναι η χρήση του CountDownLatch αντικειμένου το οποίο παίρνει σαν παράμετρο "1" το οποίο είναι μια χρονική διάρκεια. Στο τέλος της μεθόδου run() καλούμε την μέθοδο countDown() του latch αντικειμένου, το οποίο θα χρησιμεύσει παρακάτω, ώστε το main thread να περιμένει το network thread να επιστρέψει το αποτέλεσμα του. Αμέσως μετά καλούμε τις μεθόδους start() του Thread και την μέθοδο await για να το περιμένουμε να επιστρέψει αν το login_successfull είναι true ή false.

Επιπροσθέτως, χρειάζεται να γίνει ιδιαίτερη αναφορά στην μέθοδο performLogin() η οποία θα πάρει τις παραμέτρους που αναφέραμε πριν και θα κάνει το HTTP αίτημα στο Web Service. Για αρχή με χρήση ternary ορίζουμε το URI στο οποίο θα κάνουμε το αίτημα, δηλαδή το "/ptyxiaki/index.php/api/v1/Administrators/Sessions" για admin και το "/ptyxiaki/index.php/api/v1/Employees/Sessions" για Employee . Αμέσως μετά δημιουργούμε ένα αντικείμενο RestClient. Στην πορεία δημιουργούμε ένα JSONObject αντικείμενο το οποίο θα περιλαμβάνει τα username και password. Με την μέθοδο put("username", username) και put("password", password) θα περάσουμε στα κλειδιά του JSON τις τιμές των μεταβλητών. Μετά την δημιουργία του αντικειμένου αυτού, το περνάμε σαν παράμετρο στην μέθοδο doPost(uri, jsonObject), η οποία θα στείλει το JSON το οποίο ορίσαμε παραπάνω να το στείλει στο URI που ορίσαμε. Αμέσως μετά από jsonObjectResponse περιμένουμε JSON με κλειδί administratorTokken για admin loginType και userTokken κλειδί για manager loginType. Και στις 2 περιπτώσεις έχουμε στα χέρια μας ένα tokken που θα το στέλνουμε συνέχεια στο web service για την επικοινωνία μας.

Τέλος θα αποθηκεύσουμε αυτό το tokken στα SharedPreferences. Η πρόσβαση γίνεται με την δημιουργία ενός πεδίου τύπου SharedPreferences. Σε αυτήν την μεταβλητή θα φορτώσουμε ένα αντικείμενο με την κλήση της μεθοδου getDefaultSharedPreferences(getApplicationContext) της κλάσης PreferenceManager. Αμέσως μετά ορίζουμε ένα SharedPreferences.Editor πεδίο και από το SharedPreference αντικείμενο επιστρέφουμε τον Editor για να προσθέσουμε το token. Αν το token δεν είναι null τότε θα προσθέσουμε στα SharedPreferences το tokken με κλεδί "administratorTokken" ή "userTokken" αντίστοιχα για κάθε loginType. Στην συνέχεια με την μέθοδο commit() του editor κάνουμε save το key. Αν όλα αυτά γίνουν με επιτυχία επιστρέφουμε true για την επιτυχή σύνδεση και false στην περίπτωση που δεν έχουμε στα χέρια μας ένα οποιοδήποτε token.

Σχήμα 34 Διεπαφή εισόδου χρήστη

4.10 Administrator Management, Bottom Navigation View

Το πρώτο είδος σύνδεσης που θα αναλυθεί είναι του Administrator. Στην περίπτωση που κάποιος κάνει σύνδεση στην εφαρμογή σαν διαχειριστής θα οδηγηθεί σε αυτό το activity. Το Activity αποτελείται από δυο βασικά κομμάτια, το activity_administrator layout και το AdministratorManagement αρχείο Java.

Για αρχή, όσον αφορά το view, αποτελείται από ένα root element RelativeLayout στο οποίο ορίζουμε το attribute context ".AdministratorManagement". Αμέσως μετά έχει δυο βασικά view, το πρώτο είναι ένα FrameLayout στο οποίο θα φορτώνουμε κάθε διαφορετική διεπαφή του Android καθώς και ένα BottomNavigaion το οποίο είναι ένα toolbar για να κάνουμε περιήγηση στο κάθε διαφορετικό Fragment. Σε αυτό το view ορίζουμε σαν menu το αρχείο "administrator_bottom_navigation_bar" το οποίο περιλαμβάνει ένα root element menu και ορίζουμε 4 item elements μέσα σε αυτό. Το πρώτο έχει τίτλο "Shops" και ένα εικονίδιο μαγαζιού, το δεύτερο έχει τίτλο "Managers" και ένα εικονίδιο υπαλλήλου, το τρίτο έχει τίτλο "Server" και ένα σχετικό εικονίδιο και το τέταρτο έχει ένα εικονίδιο κλεισίματος και τίτλο "Logout".

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
    android:id="@+id/nav_shops"
    android:icon="@drawable/ic_shops_icon"
    android:title="Shops" />
    <item
        android:id="@+id/nav_managers"
        android:icon="@drawable/ic_manager_icon"
        android:title="Managers" />
    <item
        android:id="@+id/nav_server_details"
        android:icon="@drawable/ic_server_icon"
        android:title="Server" />
    <item
        android:id="@+id/nav_logout"
        android:icon="@drawable/ic_power_settings_new_black_24dp"
        android:title="Logout"/>
</menu>
```

Σχήμα 35 Μενού Περιήγησης διαχειριστή

Η κλάση η οποία διαχειρίζεται την περιήγηση είναι η AdministratorManagement. Η κλάση αυτή περιλαμβάνει 2 βασικά πεδία. Το πρώτο είναι ένα defaultFragment πεδίο τύπου ShopFragment και βάζουμε στην μεταβλητή ένα αντικείμενο τέτοιου τύπου. Επίσης ορίζουμε και ένα πεδίου τύπου BottonNavigationView το οποίο θα κρατάει ένα reference στο αντικείμενο της διεπιφάνειας. Επίσης εκτός από αυτά έχουμε και ένα πεδίο τύπου onNavigationItemSelectetedListener το οποίο ορίζει έναν

listener στο bottom navigation view το οποίο θα γίνεται triggered όταν επιλέγεται ένα καινούργιο Fragment. Η μέθοδος η οποία θα καλείται είναι η onNavigationItemSelected. Η μέθοδος θα έχει μια switch case δομή η οποία μόλις επιλέγεται η επιλογή με id “nav_managers” η μεταβλητή selectedFrag θα κρατάει το αντικείμενο του ManagersFragment για επιλογή. Το ίδιο θα γίνεται με το id “nav_server_details” για το ServersFragment, για το id “nav_logout” και το LogoutFragment και το “nav_shops” με το ShopsFragment(). Τέλος για να γίνει η αλλαγή στο Fragment θα πρέπει να γίνει μια σειρά ενεργειών. Από τον Fragment Manager θα πρέπει να ξεκινήσει ένα transaction και να γίνει αντικατάσταση του περιεχομένου του “administrator_fragment_container” δηλαδή του FrameLayout που αναφέρθηκε στο View και να περάσουμε σαν παράμετρο το Fragment που θέλουμε για να φορτώσουμε. Αυτά γίνονται με την χρήση του SupportFragmentManager όπως στο “ getSupportFragmentManager().beginTransaction().replace(R.id.administrator_fragment_container, selectedFrag).commit();”

```
private BottomNavigationView.OnNavigationItemSelectedListener navListener =
        new BottomNavigationView.OnNavigationItemSelectedListener() {
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
                Fragment selectedFrag = null;
                switch (menuItem.getItemId()) {
                    case R.id.nav_managers:
                        selectedFrag = new ManagersFragment();
                        break;
                    case R.id.nav_server_details:
                        selectedFrag = new ServersFragment();
                        break;
                    case R.id.nav_logout:
                        selectedFrag = new LogoutFragment();
                        break;
                    case R.id.nav_shops:
                        selectedFrag = new ShopsFragment();
                        break;
                }
                getSupportFragmentManager().beginTransaction().
                        replace(R.id.administrator_fragment_container, selectedFrag).commit();
                return true;
            }
        };
```

Σχήμα 36 Λειτουργικότητα περιήγησης μενού διαχειριστή

4.11 Logout Fragment

Το Fragment το οποίο θα αναλυθεί πρώτο είναι το LogoutFragment. Το συγκεκριμένο Fragment δεν έχει κάποιο view. Δηλαδή είναι απλά κενό το view του. Το xml είναι το administrator_fragment_logout χωρίς περιεχόμενο. Ιδιαίτερο ενδιαφέρον παρουσιάζει το LogoutFragment στον κώδικα του. Για αρχή κάνουμε finish() το συγκεκριμένο Activity και στην πορεία θα δημιουργήσουμε ένα Intent για ένα άλλο activity. Το Intent θα δεχθεί ως παραμέτρους το activity με την μέθοδο getActivity() και το LoginActivity.class. Στην συνέχεια θα κάνουμε startActivity με παράμετρο το συγκεκριμένο Intent. Τέλος με την ολοκλήρωση της μεθόδου onViewCreated δημιουργούμε πρόσβαση στα Shared Preferences σε MODE_PRIVATE. Στα Shared Preferences υπάρχουν 3 είδη MODE. Το πρώτο είναι MODE_PRIVATE όπου το αρχείο που δημιουργείται μπορεί να διαβαστεί μόνο από την εφαρμογή που το καλεί. Το επόμενο είναι το MODE_WORLD_READABLE κατά το οποίο όλες οι άλλες εφαρμογές έχουν την δυνατότητα να διαβάσουν το αρχείο και το τελευταίο είναι το MODE_WORLD_WRITEABLE κατά το οποίο όλες οι εφαρμογές μπορούν να κάνουν τροποποίηση το αρχείο αυτό. Τα τελευταία 2 καταργήθηκαν στο API 17. Στην συνέχεια του κώδικα δημιουργούμε έναν editor και κάνουμε clear ότι υπάρχει μέσα στο SharedPreferences και κάνω clear() και commit().

4.12 Server Fragment και Animations

Το αμέσως επόμενο Fragment το οποίο χρειάζεται να αναφερθεί είναι το ServerFragment στο οποίο εμφανίζονται όλες οι πληροφορίες του Server, να επιτρέπεται η αναφορά ενός bug και ο έλεγχος για νεότερες εκδόσεις της εφαρμογής. Το πρώτο κομμάτι είναι το view το οποίο είναι το administrator_fragment_server layout (Σχήμα 38). Το συγκεκριμένο layout αποτελείται από ένα root element Linear με vertical orientation. Μέσα σε αυτό υπάρχουν ένα LinearLayout με vertical orientation το οποίο περιλαμβάνει ένα TextView με text “Report a bug” και έχει και ένα drawableRight στο οποίο ορίζουμε ένα εικονίδιο για το bug. Αμέσως παρακάτω έχουμε ένα view το οποίο είναι μια μαύρη γραμμή για διαχωριστή. Παρακάτω υπάρχει ένα EditText το οποίο έχει ένα attribute inputType και έχει τιμή textMultiline για να έχει παραπάνω από μια γραμμές και ορίζω και σαν background shape το drawable whiteshape. Επίσης παρακάτω υπάρχει ένα Button για να κάνω submit το bug, ορίζω έναν τίτλο με ένα TextView με text “Server details” και ένα view διαχωριστή, μια μαύρη γραμμή. Στο κομμάτι των πληροφοριών του Server υπάρχει ένα FrameLayout το οποίο έχει πάλι σαν background το whiteshape για να είναι και λευκό το φόντο και να είναι στρόγγυλο στις γωνίες. Μέσα σε αυτό υπάρχει ένα TableLayout το οποίο ορίζει όλες τις πληροφορίες του server σε πίνακα. Το πίνακας περιέχει τέσσερα <TableRow> elements τα οποία περιλαμβάνουν τα στοιχεία της κάθε γραμμής. Η πρώτη γραμμή περιλαμβάνει 2 TextView με την έκδοση της Mysql, η δεύτερη γραμμή περιλαμβάνει τα στοιχεία τα στοιχεία για την IP του server, η τρίτη για το API version και η τέταρτη περιλαμβάνει την έκδοση της PHP. Τέλος, στο κάτω μέρος του Fragment υπάρχει ένα TextView το οποίο έχει text “Check for upgrades”, ένα εικονίδιο ανανέωσης“ic_refresh_black_24dp” και έχει το attribute clickable true έτσι ώστε με το κλικ πάνω σε αυτό να γίνεται έλεγχος για νεότερες εκδόσεις της εφαρμογής.

Για την λειτουργικότητα του ServersFragment υπάρχει και ServersFragment το οποίο θα κάνει την σύνδεση με τον Server για να πάρει τις διάφορες πληροφορίες που χρειάζεται. Τα δύο πεδία τα οποία χρειάζονται είναι το reportBugET τύπου EditText για νά έχουμε πρόσβαση στο κείμενο που έδωσε ο χρήστης για την αναφορά του bug, αλλά και bugReportB Button το οποίο θα αναλάβει να πιάσει το click event. Το πρώτο κομμάτι αποτελείται από την μέθοδο onCreateView() η οποία θα εμφανίσει τις πληροφορίες στην οθόνη. Σε αυτό το σημείο καλείται η μέθοδος initView() που θα κάνει όλη την δουλειά. Η μέθοδος θα καλέσει την μέθοδο getDetails() η οποία θα επιστρέψει ένα JSONObject με όλες τις πληροφορίες του server. Αν κάτι δεν πάει καλά στην λήψη θα εμφανίσει “Thread error. Please contact development team”. Στην συνέχεια με την μέθοδο findViewById θα δημιουργήσουμε αναφορές στα TextView με τα ID(s) “server_details_api_version”, “server_details_mysql_version”, “server_details_php_version”, “server_details_server_ip” για να έχουμε πρόσβαση σε αυτά σαν αντικείμενα της Java. Στην περίπτωση που επιστρέψει null υπάρχει εμφάνιση μηνύματος “Error getting server data” σε ένα SnackBar. Στην συνέχεια από αυτό το JSONObject με τα στοιχεία του server παίρνω όλα τα πεδία και τα αναθέτω στα textView του πίνακα της διεπιφάνειας με την χρήση της μεθόδου setText() που καλώ σε κάθε TextView.

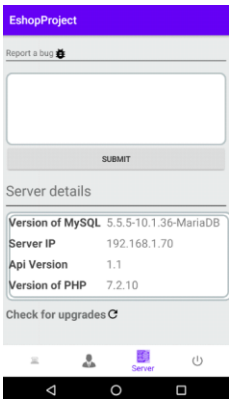
Αναφορά επίσης πρέπει να γίνει στην μέθοδο getDetails() που αναφέρθηκε πιο πριν. Όπως και σε προηγούμενη επικοινωνία με το Web Service υπάρχει ανάγκη για καινούργιο Thread. Ορίζουμε ένα CountDownLatch αντικείμενο με χρονική διάρκεια “1” η οποία θα αναγκάσει το main thread να περιμένει το NetworkThread. Στην μέθοδο run του HandlerThread φτιάχνουμε ένα RestClient αντικείμενο και καλούμε την μέθοδο “doGetRequestSingleObject(uri)” και περνάμε σαν παράμετρο ένα URI για το συγκεκριμένο Resource που είναι οι πληροφορίες του server. Το URI είναι το

“/ptyxiaki/index.php/api/v1/ServerDetails”. Μέσα στο thread καλούμε την μέθοδο countdown() του latch και έξω από το thread , το κάνουμε run() και καλούμε την μέθοδο latch.await().

Επίσης μια άλλη μέθοδος η οποία έχει υλοποιηθεί είναι αυτή που κάνει την αρχικοποίηση της report bug φόρμας η initBugReportForm() και παίρνει σαν παράμετρο το view της διεπιφάνειας του Fragment. Για αρχή με την μέθοδο findViewById βρίσκουμε το button με Id “bugSubmit”. Στην συνέχεια ορίζουμε έναν onClickListener για όταν γίνεται click το κουμπί αναφοράς bug. Μέσα στον Listener ορίζουμε την μέθοδο onClick όπου θα βρούμε το EditText με id “bugText” το οποίο περιλαμβάνει το κείμενο. Στην συνέχεια γίνεται έλεγχος για το αν το EditText περιέχει κείμενο ή όχι. Αν είναι άδειο το editText εμφανίζουμε ένα SnackBar με κείμενο “No content submitted” ενώ αν περιέχει κείμενο καλείται η μέθοδος submit η οποία θα κάνει submit το κείμενο στο server και θα εμφανίσει ένα SnackBar με κείμενο “Thanks for your feedback”.
Το τελευταίο κομμάτι του Fragment είναι ο έλεγχος για νέες εκδόσεις της εφαρμογής. Για αρχή βρίσκουμε το TextView που κάνει τον έλεγχο και ορίζουμε έναν click listener, όπου θα βάλει την τιμή “Now checking...” και θα ορίσει ένα animation για να δείχνει στον χρήστη ότι ελέγχει για αναβαθμίσεις. Με την κλάση AlphaAnimation φτιάχνουμε το animation με 2 παραμέτρους 1.0 και 0.0, που δείχνει πως το animation θα κάνει fade out, δηλαδή θα εξαφανίζεται. Στην συνέχεια ορίζουμε διάρκεια, αριθμό επαναλήψεων και τι είδος επαναλήψεις θα κάνει καλώντας τις μεθόδους setDuration(1000) setRepeatCount(6) και setRepeatMode(Animation.REVERSE). Τέλος θα οριστεί ένας AnimationListener, όπου στην μέθοδο onAnimationEnd α κάνουμε setText(“You have the latest version”) και θα αντικαταστήσουμε το εικονίδιο ανανέωσης με εικονίδιο check (Σχήμα 37).

```
public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
    view.findViewById(R.id.upgradeCheck).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            TextView textView = v.findViewById(R.id.upgradeCheck);
            textView.setText("Now checking...");
            AlphaAnimation anim = new AlphaAnimation(1.0f, 0.0f);
            anim.setDuration(1000);
            anim.setRepeatMode(Animation.REVERSE);
            anim.setRepeatCount(6);
            anim.setAnimationListener(new Animation.AnimationListener() {
                @Override
                public void onAnimationStart(Animation animation) { }
                @Override
                public void onAnimationEnd(Animation animation) {
                    textView.setText("You have the latest version");
                    textView.setCompoundDrawablesWithIntrinsicBounds(R.drawable.ic_check_black_24dp, top: 0, right: 0, bottom: 0);
                }

                @Override
                public void onAnimationRepeat(Animation animation) { }
            });
            textView.startAnimation(anim);
        }
    });
}
```

Σχήμα 37 Δημιουργία animation ελέγχου έκδοσης εφαρμογής



Σχήμα 38 Διεπαφή πληροφοριών εφαρμογής

4.13 Shops Fragment and Custom List View Adapter

Ένα άλλο σημείο το οποίο χρειάζεται ιδιαίτερη αναφορά είναι λίστα με τα καταστήματα που υπάρχει. Αυτή η λίστα υπάρχει στο ShopsFragment το οποίο υπάρχει και η σχεδίαση του υπάρχει στο αρχείο administrator_fragment_shops για να δημιουργηθεί η διεπαφή (Σχήμα 40). Το αρχείο xml περιέχει ένα root element RelativeLayout. Μέσα σε αυτό το element υπάρχει ένα ListView το οποίο θα περιέχει την λίστα με τα Shops. Αμέσως κάτω από αυτό υπάρχει ένα FloatingActionButton με εικονίδιο πρόσθεσης. Παρακάτω υπάρχει ένα ImageButton το οποίο είναι invisible και έχει ένα εικονίδιο refresh. Αυτό το εικονίδιο θα είναι χρήσιμο στην πορεία για όταν θέλουμε να κάνουμε ανανέωση της λίστας με τα καταστήματα. Τέλος υπάρχει ένα ProgressBar το οποίο έχει το εικονίδιο του refresh και δείχνει στον χρήστη ότι κάνει προσπάθεια ανανέωσης της λίστας με τα καταστήματα.

Το αρχείο Java της διεπαφής είναι το ShopsFragment το οποίο θα περιέχει 2 πεδία. Το πρώτο είναι ένα ListView και το δεύτερο είναι ένα Handler πεδίο. Η πρώτη μέθοδος η οποία θα αναλυθεί είναι η getShops() η οποία επιστρέφει ένα JSONArray αντικείμενο το οποίο θα περιέχει μια λίστα με JSON που θα επιστρέψει το Web Service. Μέσα σε αυτήν την μέθοδο δημιουργούμε ένα RestClient αντικείμενο και εκτελούμε την μέθοδο doGetRequest() στο URI "/ptyxiaki/index.php/api/v1/Administrators/Shops" το οποίο θα επιστρέψει την λίστα με τα καταστήματα. Η μέθοδος η οποία θα καλέσει την getShops είναι η μέθοδος retrieveAllShops() η οποία θα μετατρέψει το JSONArray σε έναν πίνακα από Shop αντικείμενα και θα τα επιστρέψει με την σειρά της για να απεικονιστούν στο view. Η μετατροπή θα γίνει με την αρχικοποίηση ενός Gson object και με την γραμμή "final Shop[] shoparray = gson.fromJson(shops.toString(), Shop[].class)", όπου shops είναι το JSONArray θα γίνει μετατροπή σε Shop[] και θα το επιστρέψει η μέθοδος retrieveAllShops().

Η κλήση της μεθόδου retrieveAllShops θα γίνει στην μέθοδο fillShopListView. Μέσα σε αυτήν την μέθοδο ορίζουμε ένα Thread αντικείμενο. Η ύπαρξη ενός καινούργιου Thread είναι αναγκαία καθώς HTTP αιτήματα δεν μπορούν να γίνουν από το main Thread. Επίσης μέσα σε αυτό το νέο Thread υπάρχει το πρόβλημα ότι δεν επιτρέπεται να πειράξουμε το View από άλλο Thread εκτός από το Main Thread. Για αυτό τον λόγο θα φτιάξουμε ένα Handler αντικείμενο με την χρήση του new Handler(getActivity().getApplicationContext().getMainLooper()) για να δημιουργήσουμε ένα νέο Runnable για να μπορέσουμε να κάνουμε Edit το view. Με την μέθοδο findViewById βρίσκουμε το

ListView της διεπιφάνειας με id “shops_listview” για να μπορώ να τοποθετήσω την λίστα με τα καταστήματα και το ProgressBar με ID “shop_progressbar”. Στην συνέχεια, ελέγχω αν είναι null ο πίνακας με τα καταστήματα. Αν είναι δεν είναι null σημαίνει πως μπορούμε να κάνουμε εμφάνιση των δεδομένων στο View.

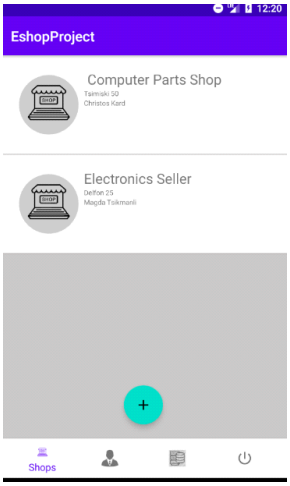
Ένα άλλο στοιχείο το οποίο χρειάζεται αναφορά είναι ότι χρησιμοποιήθηκε Java 8 για την χρήση Streams, δηλαδή στην περίπτωση για να φτιάξουμε έναν πίνακα κάνοντας map κάθε τιμής ενός πεδίου ενός αντικειμένου. Για παράδειγμα στον κώδικα στην γραμμή “String[] shopnames = Arrays.stream(shops).map(s -> s.getShopname()).toArray(size -> new String[shops.length]);” από τον πίνακα αντικειμένων τύπου Shop παίρνω από κάθε αντικείμενο Shop το πεδίο shopname με την χρήση της μεθόδου getShopname του αντικειμένου Shop. Την ίδια διαδικασία εκτελώ για τα πεδία ownerfullname, shopaddress. Έτσι έναν πίνακα με αντικείμενα Shop τα παίρνω και τα διαχωρίζω σε άλλους πίνακες από τα πεδία τους. Αυτό είναι ιδιαίτερα χρήσιμο για την εμφάνιση των στοιχείων στο View. Η εφαρμογή των πεδίων θα γίνει με την δημιουργία ενός αντικειμένου ShopsAdapter ο οποίος θα μπορεί να εμφανίσει την λίστα με τα καταστήματα στην οθόνη. Η δημιουργία γίνεται στην γραμμή “ShopsAdapter shopsAdapter = new ShopsAdapter(getContext(), shopnames, ownerfullnames, shopaddresses);” όπου περνάμε παραμετρικά όλους τους πίνακες που θα περαστούν στο view με την μέθοδο setAdapter του listview. Ο τρόπος αυτός θα αναλυθεί παρακάτω στον σχεδιασμό του Adapter.

Σε άλλη περίπτωση αν τα shops είναι null θα εμφανίσουμε μήνυμα “Error retrieving shops” σε ένα SnackBar, θα κάνουμε ορατό το ImageButton με id “shopsRefreshBtn”, θα κάνουμε ορατό το ProgressBar και θα ορίσουμε έναν onClickListener στο κουμπί ανανέωσης το οποίο θα κάνει ορατό το ProgressBar, αόρατο το ImageButton της ανανέωσης της λίστας και θα καλέσει την μέθοδο refreshView(). Η μέθοδος αυτή θα περιλαμβάνει δύο κλήσεις μεθόδων , την fillShopListView() και την setListeners() όπου η πρώτη θα κάνει ξανά τις κλήσεις στο Web Service και η δεύτερη θα ορίσει ξανά του Listeners της διεπιφάνειας. Στην μέθοδο setListeners, βρίσκουμε το FloatingActionButton με id “shop_add” και αν πατηθεί αυτό θα μεταφερθούμε σε άλλο Fragment. Θα δημιουργηθεί ένα Transaction για να γίνει αντικατάσταση του υπάρχοντος Fragment που βρίσκεται στο προσκήνιο και στον administrator_fragment_container που περιέχει το τρέχον εμφανιζόμενο Fragment. Το καινούργιο Fragment είναι το AddShopFragment() στο οποίο θα μεταφερθούμε.

```
public class ShopsAdapter extends ArrayAdapter<String> {
    Context context;
    String[] rshopnames;
    String[] raddresses;
    String[] rownerfullnames;

    ShopsAdapter(Context c, String shopnames[], String[] addresses, String[] ownerfullnames) {
        super(c, R.layout.shop_row, R.id.shopname, shopnames);
        this.context = c;
        this.rshopnames = shopnames;
        this.raddresses = addresses;
        this.rownerfullnames = ownerfullnames;
    }

    @NonNull
    @Override
    public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
        LayoutInflater layoutInflater = (LayoutInflater) getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        View shop_row = layoutInflater.inflate(R.layout.shop_row, parent,  attachToRoot: false);
        Log.d( tag: "ShopAdapter row loaded:",  msg: "=>" + position);

        TextView shopNameTv = shop_row.findViewById(R.id.shopname);
        TextView ownerFullNameTv = shop_row.findViewById(R.id.ownerfullname);
        TextView ownerTelephoneTv = shop_row.findViewById(R.id.ownertelephone);

        //set resource data to the row
        shopNameTv.setText(rshopnames[position]);
        ownerFullNameTv.setText(rownerfullnames[position]);
        ownerTelephoneTv.setText(raddresses[position]);

        Log.d( tag: "Data",  msg: " shopname " + rshopnames[position] + " full name" + rownerfullnames[position] + " address" + raddresses[position]);


        return shop_row;
    }
}
```

Σχήμα 39 Adapter εμφάνισης καταστημάτων

Για την εμφάνιση των καταστημάτων στο view γίνεται χρήση ενός Adapter. Η κλάση ShopsAdapter αποτελεί επέκταση της κλάσης ArrayAdapter από την οποία κληρονομούνται διάφορες ιδιότητες και μέθοδοι. Τα πεδία τα οποία δηλώνονται στην κλάση είναι το Context και 3 πίνακες με τα ονόματα των καταστημάτων, με τις διευθύνσεις των καταστημάτων και με τα ονόματα των ιδιοκτητών των καταστημάτων. Οι μέθοδοι οι οποίες υπάρχουν είναι οι getView() και πλήρης δομητής της κλάσης. Μέσα από τον πλήρη δομητή περνιούνται οι τιμές στα πεδία της κλάσης και μέσα στην getView() δημιουργείται κάθε ένα διαφορετικό αντικείμενο της λίστας. Μέσα στην getView() δημιουργώ ένα LayoutInflater αντικείμενο με την χρήση του LAYOUT_INFLATER_SERVICE. Στην συνέχεια με αυτό το αντικείμενο δημιουργώ ένα View shop_row στο οποίο θα κάνω set στα TextView που περιέχει τα στοιχεία του καταστήματος, ανάλογα με το position της γραμμής που εμφανίζεται. Τέλος αυτή η μέθοδος επιστρέφει την γραμμή αυτή που επεξεργαστήκαμε και την επιστρέφει για να εμφανιστεί στο ListView. Το layout βρίσκεται στο αρχείο shop_row το οποίο περιέχει το layout και την μορφοποίηση της κάθε γραμμής. Η κάθε γραμμή περιέχει ένα ImageView και 3 TextView τα οποία περιέχουν την πληροφορία της γραμμής.



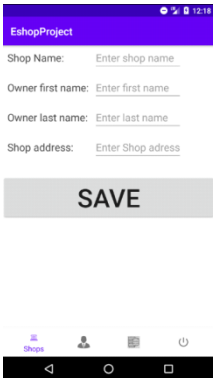
Σχήμα 40 Διεπαφή εμφάνισης καταστημάτων

4.14 Add Shop Fragment

Μετά την επιλογή του FloatingActionButton για πρόσθεση καταστήματος, μεταφερόμαστε στο Fragment AddShopFragment, το οποίο περιλαμβάνει τις βασικές πληροφορίες για την προσθήκη ενός καταστήματος. Οι πληροφορίες είναι 4, το Shop Name, το Owner first name, το Owner last name και το Shop address, τα οποία πεδία εμφανίζονται στο xml administrator_fragment_addshop. Σε αυτό έχει σχεδιαστεί η διεπιφάνεια ώστε σε ένα TableLayout, να περιέχονται διαφορετικά TableRow και να περιλαμβάνουν τα πεδία τα οποία θα γίνουν submit στο τέλος από το κουμπί “SAVE”.

Όσον αφορά την λειτουργικότητα, υπάρχει το αρχείο Java AddShopFragment το οποίο θα πάρει τις τιμές και θα τις στείλει στο Web Service. Η μέθοδος η οποία θα στείλει τα δεδομένα είναι η addShop() η οποία παίρνει σαν παράμετρο ένα αντικείμενο Shop. Στην συνέχεια θα δημιουργήσουμε ένα JSONObject στο οποίο θα κάνουμε put όλα τα πεδία του Shop αντικειμένου και τέλος θα δημιουργηθεί ένα αντικείμενο RestClient για την αποστολή του αντικειμένου στον server. Έπειτα με την μέθοδο doPost του RestClient κάνω POST το JSONObject στο endpoint “/api/v1/Administrators/Shops”. Για να φτάσουμε στην αποστολή πρέπει πρώτα από όλα να γίνει λήψη των δεδομένων από την διεπιφάνεια. Για αυτό χρησιμοποιείται η μέθοδος setListeners η οποία

ορίζει ένα onClick event στο Button που κάνει το “SAVE”. Δημιουργούμε ένα αντικείμενο τύπου Shop και κάνουμε set τα πεδία του με τις τιμές που έχουν τα πεδία της διεπιφάνειας που αναφέραμε. Για να γίνει το POST δημιουργούμε ένα καινούριο Thread όπου στην μέθοδο run() χρησιμοποιούμε την μέθοδο addShop() που εξηγήσαμε νωρίτερα. Στην συνέχεια αν πιαστεί κάποιο exception κατά την αποστολή, η μεταβλητή addShopState θα γίνει false και θα πρέπει να δείξουμε στην οθόνη μήνυμα σφάλματος. Με την χρήση της μεθόδου getMainLooper() παίρνουμε το main thread στο οποίο κάνουμε post ένα νέο runnable. Αν η μεταβλητή μας είναι true θα εμφανίσει ένα SnackBar με την συμβολοσειρά “Shop posted successfully”, αλλιώς θα υπάρχει μήνυμα “An error occured posting the shop”.



Σχήμα 41 Διεπαφή προσθήκης καταστήματος

4.15 Managers Fragment, Streams and Bundle

Για την σχεδίαση του Fragment προβολής των Manager πρέπει να αναφερθούμε στα δυο χωριστά κομμάτια που το αποτελούν. Το πρώτο είναι το αρχείο xml administrator_fragment_manager το οποίο περιλαμβάνει την σχεδίαση του. Αυτό το αρχείο αποτελείται από ένα root element RelativeLayout το οποίο καλύπτει όλη την οθόνη. Μέσα σε αυτό υπάρχει ένα FloatingActionButton το οποίο με το που γίνεται click θα μας μεταφέρει στο Fragment προσθήκης υπαλλήλου. Μέσα στο LinearLayout υπάρχει ένα Spinner το οποίο έχει το attribute spinner mode με τιμή dialog για να εμφανιστούν τα μαγαζιά σε μορφή dialog καθώς και το ListView το οποίο περιλαμβάνει την λίστα με τους Managers.

Για την λογική του Fragment υπάρχει το αρχείο java ManagersFragment. Η κλάση αυτή περιλαμβάνει 7 πεδία. Τα πρώτα 2 είναι πίνακες τύπου String με ονόματα shopnames και shopids, αμέσως μετά είναι 2 μεταβλητές τύπου Handler με ονόματα handler1, handler2 και δυο μεταβλητές currentlyDisplayedShopId και currentlyDisplayedShopName. Τέλος υπάρχει ένα πεδίο τύπου ListView το οποίο θα περιλαμβάνει την λίστα με τους managers. Πρώτα από όλα πρέπει το spinner να πάρει τις τιμές όλων των ονομάτων των καταστημάτων.

Οι μέθοδοι οι οποίοι χρησιμοποιούνται για την λήψη των δεδομένων των καταστημάτων είναι οι getShops() και retrieveAllShops() όπου η πρώτη θα καλεστεί από την δεύτερη για να κάνει την λήψη από το web service με endpoint “/api/v1/Administrators/Shops” και η δεύτερη μέθοδος θα μετατρέψει το JSONArray σε πίνακα αντικειμένων τύπου Shop. Εφόσον γίνουν αυτά θα καλεστούν οι μέθοδοι getEmployees() και retrieveEmployees(), όπου η πρώτη μέθοδος επιστρέφει σε μορφή JSONArray τους υπαλλήλους από κάποιο κατάστημα σύμφωνα με το shopid που ορίζουμε και στην συνέχεια η retrieveEmployees θα επιστρέψει έναν πίνακα από αντικείμενα τύπου Employee.

Η επόμενη μέθοδος η οποία θα χρησιμοποιηθεί είναι displayShopManagers() που παίρνει σαν παράμετρο το shopid το οποίο είναι ο μοναδικός προσδιοριστής του καταστήματος. Η μέθοδος αυτή θα καλέσει την μέθοδο retrieveEmployees() και στην συνέχεια με την χρήση Streams κάνει map όλα τα username από όλους τους employees σε έναν πίνακα String με όνομα usernames. Στην συνέχεια με χρήση του Handler object στέλνουμε ένα καινούριο Runnable στο main Thread για να μετατρέψει τον πίνακα από Managers σε μορφή κατάλληλη για το View. Για την προβολή γίνεται χρήση ενός ManagersAdapter που κληρονομεί την κλάση ArrayAdapter. Μέσα σε αυτό το Runnable θα περνάμε τον πίνακα από Employee αντικείμενα στον adapter με τον πλήρη δομητή της κλάσης.

Στην συνέχεια παραμετροποιούμε το ListView που υπάρχει στο xml με τις μεθόδους setAdapter με παράμετρο τον adapter που φτιάξαμε, καλούμε την μέθοδο destroyDrawingCache() και κάνουμε το ListView αόρατο και ξανά ορατό. Όλες αυτές οι μέθοδοι θα πρέπει να εκτελεστούν σε συγκεκριμένες στιγμές. Μέσα στην onCreateView όλα αυτά τα δεδομένα θα περαστούν ως παράμετροι στους adapters (Σχήμα 42). Μέσα στην onCreateView() βρίσκουμε το spinner και το ListView και ξεκινάμε ένα καινούργιο Thread στο οποίο καλούμε την μέθοδο retrieveAllShops() για να πάρουμε όλα τα μαγαζιά από τον server. Στην συνέχεια διαχωρίζουμε τον πίνακα με την χρήση Streams και μόλις ολοκληρωθεί η λήψη κάνουμε post ένα καινούργιο Runnable και το στέλνουμε στο main thread για να τροποποιήσει το View. Μέσα σε αυτό το Runnable δημιουργούμε έναν ArrayAdapter όπου περνάμε παραμετρικά τα ονόματα των καταστημάτων και στην συνέχεια καλούμε την μέθοδο setAdapter για το Spinner όπου περνάμε σαν παράμετρο τον adapter με τα ονόματα των μαγαζιών.

```
View view = inflater.inflate(R.layout.administrator_fragment_managers,container, attachToRoot: false);
Spinner spinner = (Spinner) view.findViewById(R.id.shop_spinner);
managersListView = (ListView) view.findViewById(R.id.managers_listview);
Thread thread = new Thread(new Runnable() {
    @Override
    public void run() {
        Shop[] shops = retrieveAllShops();
        shopnames = Arrays.stream(shops).map(s -> s.getShopname()).toArray(size -> new String [shops.length]);
        shopids = Arrays.stream(shops).map(s -> s.getId()).toArray(size -> new String[shops.length]);
        handler = new Handler(getActivity().getApplicationContext().getMainLooper());
        handler.post(new Runnable() {
            @Override
            public void run() {
                ArrayAdapter<String> adapter = new ArrayAdapter<>(getActivity(), android.R.layout.simple_spinner_item,shopnames);
                adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
                spinner.setAdapter(adapter);
                if (currentlyDisplayedShopName != null) {
                    spinner.setSelection(adapter.getPosition(currentlyDisplayedShopName));
                }
            }
        });
    }
});
thread.start();
```

Σχήμα 42 Παραμετροποίηση Master Detail διεπαφής καταστημάτων υπαλλήλων

Στην συνέχεια πρέπει να οριστεί τι θα γίνεται όταν επιλέγουμε κάποιο κατάστημα από το Spinner με τα μαγαζιά (Σχήμα 43). Για αυτόν τον λόγο θα οριστεί ένας onItemSelectedListener για να ενεργοποιηθεί όταν γίνει αυτή η επιλογή. Θα εκτελεστεί το περιεχόμενο της μεθόδου run(), όπου θα ορίσουμε στις μεταβλητές currentlyDisplayedShopId και currentlyDisplayedShopName της κλάσης τα πεδία τα στοιχεία του τρέχοντος εμφανιζόμενου καταστήματος.

```
spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
        new Thread(new Runnable() {
            @Override
            public void run() {
                currentlyDisplayedShopId = Integer.parseInt(shopids[position]);
                currentlyDisplayedShopName = shopnames[position];
                displayShopManagers(currentlyDisplayedShopId, currentlyDisplayedShopName);
            }
        }).start();
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {
    }

});
```

Σχήμα 43 Παραμετροποίηση του Spinner καταστημάτων

Τέλος πρέπει να οριστεί και η λειτουργικότητα για όταν επιλέγεται το FloatingActionButton που προσθέτει έναν καινούργιο manager (Σχήμα 44). Ορίζουμε έναν onClickListener και ελέγχουμε αν έχει επιλεχθεί ένα κατάστημα για την προσθήκη του Manager. Αν δεν έχει γίνει επιλογή καταστήματος, εμφανίζεται σε Toast το κατάλληλο μήνυμα. Στην περίπτωση που έχει γίνει επιλογή θα δημιουργηθεί ένα Bundle αντικείμενο στο οποίο με τις μεθόδους putString() και putInt() μπορούμε να περάσουμε παραμετρικά το όνομα του καταστήματος και το ID του καταστήματος. Στην συνέχεια με την μέθοδο setArguments() του Fragment AddManagerFragment, περνάμε παραμετρικά το Bundle. Στην συνέχεια με την χρήση της μεθόδου getSupportFragmentManager() ξεκινάμε ένα transaction και κάνουμε αντικατάσταση του προηγούμενου Fragment. Στην συνέχεια ορίζουμε 2 animations slide_in_left και slide_out_right για την μετάβαση στο Fragment της προσθήκης προϊόντος.

```
FloatingActionButton managersFloatingActionButton = (FloatingActionButton) view.findViewById(R.id.manager_add);
managersFloatingActionButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (currentlyDisplayedShopId == -1 || currentlyDisplayedShopName.equals(null)) {
            Snackbar.make(view, text: "No shop specified", Snackbar.LENGTH_LONG).show();
            return;
        } else {
            Bundle bundle = new Bundle();
            bundle.putString("shopname", currentlyDisplayedShopName);
            bundle.putInt("shopid",currentlyDisplayedShopId);
            Fragment fragment = new AddManagerFragment();
            fragment.setArguments(bundle);
            FragmentManager fragmentManager = getActivity().getSupportFragmentManager();
            FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
            fragmentTransaction.setCustomAnimations(android.R.anim.slide_in_left,android.R.anim.slide_out_right);
            fragmentTransaction.replace(R.id.administrator_fragment_container, fragment);
            fragmentTransaction.addToBackStack(null);
            fragmentTransaction.commit();
        }
    }
});
return view;
```

Σχήμα 44 Μετάβαση σε διεπαφή προσθήκης υπαλλήλου

4.16 Add Manager Fragment and DatePickerDialog

Για να γίνει η προσθήκη του Manager στο επιλεγμένο κατάστημα υπάρχει το AddManagerFragment, το οποίο θα πάρει τα δεδομένα από την διεπιφάνεια και θα κάνει προσθήκη του υπαλλήλου στο Web Service. Η σχεδίαση υπάρχει στο αρχείο xml administrator_fragment_addmanager. Στο αρχείο xml υπάρχει ένα root element RelativeLayout και μέσα σε αυτό υπάρχει ένα ScrolView το οποίο θα περιλαμβάνει σε ένα TableLayout όλα τα TableRow που υπάρχουν για κάθε πεδίο. Κάθε TableRow αποτελείται από ένα TextView και από ενα EditText για την προσθήκη του πεδίου. Στο πεδίο gender υπάρχει ένα spinner. Επίσης, στα πεδία Start Of Contract και End Of Contract υπάρχει και ένα ImageButton με ένα ημερολόγιο.

Όσον αφορά την λογική του Fragment υπάρχει η κλάση AddManagerFragment. Η μέθοδος η οποία ξεκινάει την λογική του Fragment είναι η OnCreateView η οποία πρώτα από όλα με την μέθοδο getArguments() μπορούμε να πάρουμε τις παραμέτρους που στέλνουμε από το προηγούμενο Fragment σε αυτό μέσω του Bundle. Για παράδειγμα, με την εξής φράση “getArguments().getString("shopname");”, μπορώ να πάρω το όνομα του καταστήματος. Το ίδιο κάνω και με το πεδίο “shopid” δηλαδή το ID του καταστήματος. Και οι 2 τιμές καταχωρούνται σε 2 μεταβλητές shopid και το shopname το οποίο θα γίνει ανάθεση στο TextView με ID “shopName”.

Ιδιαίτερη ανάλυση πρέπει να γίνει για το Calendar το οποίο θα εμφανίζεται σε pop up για την επιλογή της ημερομηνίας (Σχήμα 45). Η διαχείριση του Calendar γίνεται με τον εξής τρόπο. Μέσα στην κλάση AddManagerFragment, ορίζω ένα αντικείμενο myCalendar τύπου Calendar στο οποίο ορίζω έναν OnDateSetListener ο οποίος ενεργοποιείται όταν διαλέγεται μια ημερομηνία. Η μέθοδος η οποία θα εκτελεστεί είναι η onDateSet(), όπου ορίζουμε τα τρία βασικά σημεία του ημερολογίου, η ημερομηνία, ο μήνας και η μέρα του μήνα. Στην συνέχεια με την χρήση ενός αντικείμενο SimpleDateFormat ορίζω την μορφή της ημερομηνίας την οποία στην συνέχεια θα την κρατήσω στην μεταβλητή dialogPickerDate. Στην συνέχεια ανάλογα με το dateType, διαλέγουμε ποια ημερομηνία θα διαλέξουμε. Ανάλογα με ποιο ImageButton έχει επιλέξει παραπάνω θα υπάρχει και διαφορετικό dateType. Στην συνέχεια, ανάλογα με το ποια ημερομηνία διάλεξε ο χρήστης περνάμε σαν παράμετρο την ημερομηνία στο αντίστοιχο TextView με την χρήση της μεθόδου setText().

```
String startDate;
String endDate;
String dateType;
final Calendar myCalendar = Calendar.getInstance();
DatePickerDialog.OnDateSetListener date = new DatePickerDialog.OnDateSetListener() {
    @Override
    public void onDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth) {
        myCalendar.set(Calendar.YEAR, year);
        myCalendar.set(Calendar.MONTH, monthOfYear);
        myCalendar.set(Calendar.DAY_OF_MONTH, dayOfMonth);

        String dialogPickerDate =  new SimpleDateFormat( pattern: "yyyy-MM-dd", Locale.UK).format(myCalendar.getTime());
        Log.d( tag: "Date", dialogPickerDate);
        if (dateType == "start") {
            startDate = dialogPickerDate;
            startOfContractTV.setText(startDate);
        }
        else if (dateType == "end") {
            endDate = dialogPickerDate;
            endOfContractTV.setText(endDate);
        }
    }
};
```

Σχήμα 45 Παραμετροποίηση dialog ημερολογίου

Η χρήση του στον κώδικα του Fragment γίνεται με την μέθοδο setListeners() η οποία περιλαμβάνει τους ορισμούς onClickListener στα ImageButton των ημερολογίων και του Button το οποίο θα κάνει submit τον υπάλληλο. Η κλήση τους ημερολογίου γίνεται με την δημιουργία ενός DatePickerDialog αντικειμένου(Σχήμα 46) στο οποίο περνάμε σαν παράμετρο το Context της εφαρμογής, τον Listener που ορίσαμε παραπάνω και 3 παραμέτρους που θα πάρουμε από το myCalendar που ορίσαμε το πεδία “YEAR”, “MONTH” και “DAY_OF_MONTH”. Τέλος καλείται η μέθοδος show() η οποία θα δείξει το αναδυόμενο παράθυρο διαλόγου. Στην συνέχεια ορίζεται επίσης ένας onClickListener στο Button το οποίο θα κάνει προσθήκη τον υπάλληλο. Μέσα στην onClick() του κουμπιού θα οριστεί ένα αντικείμενο τύπου Employee το οποίο θα γεμίσουμε με τα πεδία που υπάρχουν στο View και θα δημιουργήσουμε ένα νέο Thread όπου θα καλέσουμε την μέθοδο addManager() με παραμέτρους το αντικείμενο “Employee” και το “shopid”. Στην συνέχεια αν γίνει σωστά η προσθήκη θα εμφανιστεί μήνυμα σε SnackBar “Manager posted Successfully” αλλιώς θα εμφανίσει μήνυμα “An error occurred posting the manager”. Αμέσως μετά, δημιουργούμε ένα αντικείμενο ManagersFragment() το οποίο θα κάνουμε αντικατάσταση στον administrator_fragment_container ο οποίος περιέχει το τρέχον εμφανιζόμενο Fragment.

```
public void setListeners(View view) {
    startOfContractTV = view.findViewById(R.id.add_manager_start_of_contractTV);
    endOfContractTV  = view.findViewById(R.id.add_manager_end_of_contractTV);
    ImageButton imageButton = (ImageButton) view.findViewById(R.id.startOfContractChooser);
    imageButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            new DatePickerDialog(getContext(), date, myCalendar
                    .get(Calendar.YEAR), myCalendar.get(Calendar.MONTH),
                    myCalendar.get(Calendar.DAY_OF_MONTH)).show();
            dateType = "start";
        }
    });
    ImageButton imageButtonSOC = (ImageButton) view.findViewById(R.id.endOfContractChooser);
    imageButtonSOC.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            new DatePickerDialog(getContext(), date, myCalendar
                    .get(Calendar.YEAR), myCalendar.get(Calendar.MONTH),
                    myCalendar.get(Calendar.DAY_OF_MONTH)).show();
            dateType = "end";
            endDate = new SimpleDateFormat( pattern: "yyyy-MM-dd", Locale.UK).format(myCalendar.getTime());
            endOfContractTV.setText(endDate);

        }
    });
```

Σχήμα 46 Εμφάνιση dialog ημερομηνίας

4.17 Manager Activity Navigation Drawer and Animations

Το δεύτερο είδος Activity είναι το ManagerActivity το οποίο περιλαμβάνει τις λειτουργικότητες που μπορεί περιέχει ο ρόλος του Manager. Το activity όσον αφορά την σχεδίαση, αποτελείται από το activity_manager xml, το οποίο περιλαμβάνει την σχεδίαση του Intro Page (……) για τους managers και το ManagerActivity αρχείο java.

Το πρώτο αποτελείται από ένα FrameLayout το οποίο θα είναι το container στο οποίο θα εμφανίζονται όλα τα Fragments που θα εναλλάσσονται. Το id του FrameLayout θα είναι το “manager_fragment_container” για να έχουμε πρόσβαση σε αυτό. Η εναλλαγή αυτή θα γίνεται με την χρήση ενός Navigation Drawer το οποίο θα περιλαμβάνει τις διαθέσιμες επιλογές για Fragments τα οποία θα εμφανίζονται στον container. Το Navigation Drawer ορίζεται από το Navigation View το οποίο πρέπει να βρίσκεται τέρμα κάτω στο DrawerLayout του ManagerActivity. Το Navigation View έχει ένα attribute menu το οποίο ορίζει το menu το οποίο θα εμφανίζει στο Navigation Drawer. Το menu αυτό είναι το manager_drawer_menu αρχείο xml. Το root element του αρχείου είναι το menu. Αμέσως μέσα σε αυτό υπάρχει ένα group element το οποίο περιέχει όλα τα item elements της λίστας του Navigation Drawer. Τα item elements έχουν ως title “Products”, “Orders”, “Statistics-Reports”, “About”, “Logout” καθώς και ένα άλλο item χωρίς τίτλο το οποίο δεν είναι visible. Το κάθε item επίσης έχει και ένα εικονίδιο για την κάθε επιλογή. Αυτές είναι οι επιλογές οι οποίες υπάρχουν στο Navigation Drawer. Επίσης πρέπει να γίνει ιδιαίτερη αναφορά στο LinearLayout με vertical orientation που υπάρχει στο activity_manager layout, το οποίο αποτελεί το intro page του manager. Μέσα σε αυτό υπάρχουν 7 textView τα οποία περιλαμβάνουν το τι μπορεί να κάνει κάποιος manager μέσω της εφαρμογής.

Η λειτουργικότητα αυτού του activity, ορίζεται στο αρχείο java ManagerActivity. Η κλάση αυτή εμπεριέχει 2 βασικές μεταβλητές οι οποίες έχουν να κάνουν με το view. Το ένα είναι ένα πεδίο τύπου DrawerLayout και ένα πεδίο τύπου NavigationView για να κρατήσουν τα αντίστοιχα στοιχεία του view. Στην κλάση επίσης υπάρχουν 3 μέθοδοι που θα αναλυθούν, η logout(), η introPageAnimation() και η onCreate(). Η πρώτη που θα αναλυθεί είναι η introPageAnimation() (Σχήμα 47) στην οποία δημιουργούμε ένα αντικείμενο τύπου AlphaAnimation και στον πλήρη δομητή του περνάμε τιμές 0.0f

και 1.0f πράγμα το οποίο δείχνει πως το view που θα εμφανίζεται με το animation θα κάνει fade in. Στη συνέχεια ορίζω διάρκεια και τον τύπο της επανάληψης του animation. Αυτό γίνεται με τις μεθόδους setDuration() και setRepeatMode όπου θα περάσουμε μια χρονική διάρκεια στο πρώτο, και ABSOLUTE στην δεύτερη μέθοδο για να εκτελεστεί μια φορά το animation. Στην πορεία βρίσκω όλα τα TextView με τις πληροφορίες και καλώ για κάθε TextBox την μέθοδο startAnimation και περνάω σαν παράμετρο το animation που φτιάχτηκε. Στην συνέχεια για κάθε TextBox αλλάζω το duration του animation για να εμφανιστούν με μια διαδοχική σειρά από πάνω προς τα κάτω.

```
private void introPageAnimation() {
    AlphaAnimation anim = new AlphaAnimation(0.0f, 1.0f);
    anim.setDuration(2000);
    anim.setRepeatMode(Animation.ABSOLUTE);

    findViewById(R.id.intro_prmTV).startAnimation(anim);
    anim.setDuration(7000);
    findViewById(R.id.intro_cTV).startAnimation(anim);
    anim.setDuration(10000);
    findViewById(R.id.intro_oTV).startAnimation(anim);
    anim.setDuration(14000);
    findViewById(R.id.intro_sTV).startAnimation(anim);
}
```

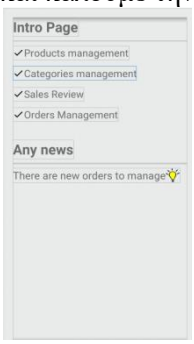
Σχήμα 47 Animation εισαγωγικής σελίδας manager

Η αμέσως επόμενη μέθοδος η οποία θα αναλυθεί είναι η logout. Η μέθοδος αυτή παίρνει σαν παράμετρο το token που έχει η εφαρμογή και θα κάνει ένα αίτημα στο Web Service για να το ακυρώσει. Όπως και στις προηγούμενες περιπτώσεις , ορίζουμε ένα JSONObject αντικείμενο στο οποίο θα περάσουμε σαν παράμετρο το token σε ένα JSON με κλειδί userTokken και θα δημιουργήσουμε ένα καινούργιο Thread για την εκτέλεση του HTTP αιτήματος στο Web Service. Στο καινούργιο Thread φτιάχνουμε ένα αντικείμενο RestClient και χρησιμοποιούμε την μέθοδο doDeleteRequest και περνάμε σαν παράμετρο το URI με endpoint “/api/v1/Employees/Sessions”. Στην συνέχεια αν το Web Service έχει επιστρέψει HTTP code 204 τότε σημαίνει πως η αποσύνδεση έγινε με επιτυχία. Αμέσως μετά δημιουργούμε έναν SharedPreferences editor με τον οποίο κάνουμε clear και commit ότι υπάρχει αποθηκευμένο. Τέλος δημιουργούμε ένα Intent αντικείμενο και περνάμε 2 παραμέτρους το LoginActivity.class στο οποίο θα μεταφερθούμε με την επιτυχή αποσύνδεση του χρήστη και το Context και εκτελούμε την μέθοδο finish() για τερματίσει το τρέχον activity και εκτελούμε την μέθοδο startActivity με παράμετρο το Intent που δημιουργήσαμε.

Σε περίπτωση που εμφανιστεί πρόβλημα σύνδεσης με τον Server θα εμφανιστεί μήνυμα “Could not connect to server” ή αλλιώς αν υπάρχει πρόβλημα με την αποκωδικοποίηση του JSON θα εμφανιστεί “Unexpected response from server” μήνυμα σε Toast. Η μέθοδος η οποία θα ξεκινήσει την όλη διαδικασία είναι η onCreate() στην οποία πρώτα από όλα θα κάνουμε assign στα object της κλάσης τα αντίστοιχα View που υπάρχουν και θα καλεστεί και η introPageAnimation().

Επίσης στην onCreate() ορίζεται και ένας OnNavigationItemSelectedListener, ο οποίος θα γίνεται trigger όταν επιλέγεται το ένα item. Για αρχή θα κάνουμε invisible το layout του intro page και ανάλογα από το id του item καλούμε διαφορετικό Fragment. Αν η getItemId() επιστρέφει το Id του item του menu που έχουμε, κάνει match στο ανάλογο case. Τα id των items του manager_drawer_menu είναι τα εξής: “nav_products”, “nav_orders”, “nav_reports”, “nav_about” και “nav_logout”. Ανάλογα με τον τίτλο καλείται και το αντίστοιχο Fragment με την χρήση του getSupportFragmentManager όπου κάνουμε replace το νέο Fragment στο manager_fragment_container. Στην περίπτωση που γίνει επιλογή του item “nav_logout” θα γίνει κλήση της μεθόδου logout() η οποία θα επιστρέψει true ή false ανάλογα αν έγινε η αποσύνδεση και θα εμφανιστεί σε Toast μήνυμα “Unable to logout”. Τέλος για την καλύτερη λειτουργία του Navigation

Drawer καλούμε την μέθοδο setChecked() στο επιλεγμένο item για να εμφανιστεί σαν χρωματισμένο και καλούμε την μέθοδο closeDrawer() του DrawerLayout για να κλείσει.



Σχήμα 48 Αρχική σελίδα Manager

4.18 Charts Fragment, Bar Chart Creation

Πρώτη διεπιφάνεια η οποία θα αναλυθεί είναι η το ChartsFragment. Αυτό το Fragment γίνεται replace στον container που αναφέρθηκε στο ManagersActivity. Το πρώτο κομμάτι που θα αναφερθεί είναι η xml σχεδίαση του Fragment η οποία βρίσκεται στο manager_chart_fragment. Για αρχή υπάρχει ένα root element RelativeLayout το οποίο περιέχει 2 elements. Το πρώτο είναι ένα textView το οποίο περιέχει την λέξη “Chart” και δεύτερο είναι ένα BarChart element το οποίο θα περιέχει όλα τα reports. Το export του chart θα γίνει κυρίως από κώδικα, για αυτόν τον λόγο και πρέπει να γίνει αναφορά στην κλάση ChartsFragment.

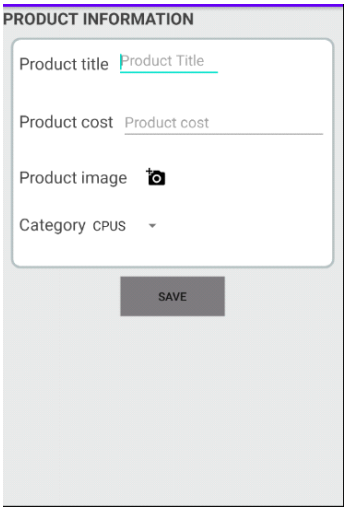
Η πρώτη μέθοδος η οποία θα αναλυθεί είναι η getProfitePerDate() η οποία επιστρέφει ένα HashMap με τιμές <String, Double>. Η υλοποίηση της γίνεται ως εξής: για αρχή δηλώνουμε ένα πεδίο HashMap το οποίο θα περιέχει το συνολικό κέρδος ανά την ημέρα. Το κέρδος είναι Double και η ημερομηνία είναι τύπου String. Στην συνέχεια ορίζω ένα αντικείμενο RestClient και ανοίγω καινούργιο Thread. Με την μέθοδο doGetRequest στο endpoint “/api/v1/Customers/Orders/All” και τα κρατάω σε ένα αντικείμενο JSONArray. Στην συνέχεια κάνω την μετατροπή από το JSONArray σε έναν πίνακα αντικειμένων Order με την χρήση της βιβλιοθήκης Gson, με τον εξής τρόπο “ final Order[] ordersArray = gson.fromJson(ordersJsonArray.toString(), Order[].class);”. Στην συνέχεια με μια διπλή for κάνουμε προσπέλαση και προσθέτουμε ανά ημερομηνία το συνολικό κέρδος. Η μία For κάνει προσπέλαση όλες τις παραγγελίες και η δεύτερη κάνει προσπέλαση όλα τα προϊόντα. Έτσι από κάθε παραγγελία μαζεύουμε το συνολικό κόστος και αν υπάρχει τέτοιο string key ημερομηνίας στο HashMap θα πάμε σε αυτό το key και θα προσθέσουμε σε εκείνη την ημερομηνία το συνολικό κέρδος από το τρέχον order που κάνουμε προσπέλαση. Όλες αυτές οι διαδικασίες γίνανε σε άλλο Thread και γιατί περιέχουν διεργασίες δικτύου και γιατί δεν πρέπει να βαραίνουμε το main thread από υπολογισμούς. Τέλος αυτό το hashmap θα επιστραφεί από την μέθοδο.

Η επόμενη μέθοδος είναι η getBarChartData() η οποία θα επιστρέφει ένα ArrayList που θα περιέχει μέσα BarEntry αντικείμενα. Η μέθοδος αυτή θα καλέσει την μέθοδο getProfitPerDate η οποία επιστρέφει το HashMap το οποίο αναλύθηκε. Για να γίνει η πρόσβαση σε κάθε μια γραμμή του HashMap χρησιμοποιήθηκε το Stream API της JAVA 8. Από το Hashmap αντικείμενο παίρνουμε το σύνολο των entries με την μέθοδο entrySet(), στη συνέχεια από αυτό το entrySet καλούμε την foreach μέθοδο όπου για κάθε stringDoubleEntry θα το προσθέτουμε στο ArrayList που θα επιστρέψει η μέθοδος. Η μέθοδος η οποία θα την καλέσει την την μέθοδο getBarChartData είναι η onCreateView η οποία θα αρχικοποιήσει το view. Για αρχή βρίσκουμε barChart αντικείμενο με την μέθοδο findViewById και ορίζουμε 2 μεταβλητές. Είναι 2 ArrayList, το πρώτο με τα heights και το άλλο με

xAxisLabel. Στην συνέχεια από την μέθοδο getProfitPerDate() καλούμε την μέθοδο entrySet() και στην συνέχεια την μέθοδο forEach που περιέχει ένα lambda expression για να κάνουμε προσπέλαση κάθε διαφορετική γραμμή του HashMap και τοποθετούμε στο xAxisLabel ArrayList την ημερομηνία και στο heights ArrayList το BarEntry αντικείμενο.

Σε αυτό το σημείο ξεκινήσουμε την παραμετροποίηση ορίζουμε ένα BarDataSet αντικείμενο όπου του περνάμε σαν παραμέτρους τα ύψη heights και το όνομα του Dataset "Dataset 1", ορίζουμε στο barDataSet αντικείμενο χρώμα με την μέθοδο setColors(), ορίζουμε Description αντικείμενο στο οποίο κάνουμε setText τον τίτλο "Days" και το κάνουμε setDescription στο barChart αντικείμενο. Τέλος ορίζουμε τον άξονα X'X με την χρήση ενός αντικειμένου Xaxis. Σε αυτό το αντικείμενο ορίζω ένα IaxisValueFormatter αντικείμενο με την μέθοδο setValueFormatter του Xaxis αντικειμένου. Στην συνέχεια με την μέθοδο setPosition ορίζω TOP position για τα labels του άξονα x'x και ορίζω με τις μεθόδους setDrawGridLines(true) και setDrawAxisLine(true) να εμφανίζεται η γραμμή του x'x και να εμφανίζονται οι γραμμές grid του bar chart και με την μέθοδο setGranularity ορίζω το μικρότερο μεσοδιάστημα που θα μπορεί να εμφανιστεί στον άξονα x'x όταν κάνουμε zoom. Αμέσως μετά με τις μεθόδους setLabelCount και setLabelRotation περνάμε τον αριθμό των labels που θα έχει ο άξονας x'x καθώς και την κλίση του Label σε μοίρες. Τέλος με την μέθοδο animateY(3000) ορίζουμε το χρονικό διάστημα στο οποίο θα κάνει animation για να εμφανιστεί ολόκληρο το barChart και κάνουμε invalidate το view ώστε να ανανεωθεί εφόσον έχουν αλλάξει οι παράμετροι του.

4.19 Add Product Fragment, Pick Image From Gallery



Σχήμα 49 Διεπιφάνεια προσθήκης προϊόντος

Το επόμενο Fragment το οποίο θα αναλυθεί είναι το AddProductFragment με το οποίο θα γίνει η προσθήκη ενός καινούργιου προϊόντος. Αρχικά όσον αφορά το xml αρχείο υπάρχει ένα root element RelativeLayout το οποίο περιέχει ένα scrollview. Μέσα σε αυτό υπάρχει ένα LinearLayout το οποίο περιέχει όλα τα TextView, τα editText τα ImageButton το spinner με τις κατηγορίες προιόντος και το Button που προσθέτει το προιόν. Όλα τα πεδία τα οποία βρίσκονται μέσα στο λευκό πλαίσιο της φωτογραφίας ανήκουν σε ένα FrameLayout το οποίο έχει background το αρχείο xml whiteshape.

Οι πρώτες 3 μέθοδοι που θα αναλυθούν είναι η fillCategoriesSpinner, getAllCategories(), getCategories(). Η getCategories με χρήση του αντικειμένου RestClient που θα αρχικοποιηθεί σε ένα

καινούργιο Thread θα χρησιμοποιήσει την doGetRequst στο endpoint "/ptyxiaki/index.php/api/v1/Categories" θα επιστρέψει ένα JSONArray με τις κατηγορίες. Μετά η μέθοδος getAllCategories() θα μετατρέψει το JSONArray σε πίνακα αντικειμένων τύπου Category. Αμέσως μετά η μέθοδος fillCategoriesSpinner θα μετατρέψει τον πίνακα και με την χρήση Streams θα πάρουμε από τον πίνακα αντικειμένων Category έναν πίνακα μόνο από τα ονόματα των κατηγοριών που είναι πεδίο του αντικειμένου Category και αυτόν τον πίνακα τον μετατρέπουμε σε έναν ArrayAdapter για να κάνουμε setAdapter στο αντικείμενο του spinner για να φορτωθούν τα ονόματα των κατηγοριών. Αμέσως μετά είναι οι μέθοδοι findAllViews που βρίσκει όλα τα views του xml και τα αναθέτει σε μεταβλητές του fragment και η μέθοδος showResultDialogBox η οποία θα εμφανίσει ένα dialog για την επιτυχή πρόσθεση του προϊόντος.

Με την κλάση Builder φτιάχνουμε ένα αντικείμενο για dialog, στο οποίο θα κάνουμε μήνυμα επιτυχίας, εικονίδιο επιτυχίας και τίτλο επιτυχίας με τις μεθόδου setMessage(), setIcon και setTitle αντίστοιχα (Σχήμα 50). Μετά από αυτά καλούμε την μέθοδο create() για την δημιουργία του dialog και την μέθοδο show() για την εμφάνιση του.

```
private void showResultDialogBox (boolean productPosted) {

    String message;

    AlertDialog.Builder builder = new AlertDialog.Builder(getContext());

    if (productPosted) {

        message = "Product added successfully";
        builder.setIcon(R.drawable.ic_done_black_24dp);
        builder.setTitle("Information");

    } else {
        message = "Error adding product";
        builder.setIcon(R.drawable.ic_error_black_24dp);
        builder.setTitle("Warning");

    }
    builder.setPositiveButton( text: "OK", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {

        }
    });
    builder.setMessage(message);
    builder.setCancelable(true);
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
}
```

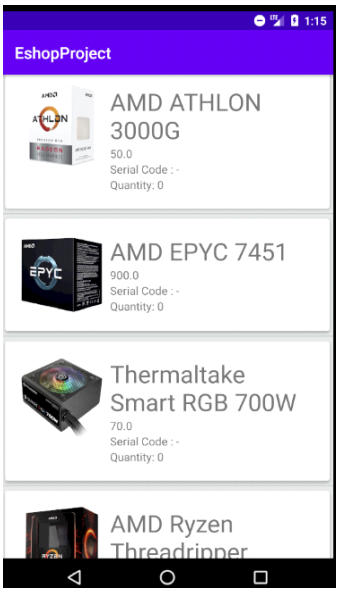
Σχήμα 50 Δημιουργία Dialog

Με το που κάνουμε click στο ImageButton με την φωτογραφική μηχανή θα πρέπει με κάποιον τρόπο να γίνει η επιλογή της φωτογραφίας του προϊόντος από το file system του κινητού. Όπως φαίνεται υπάρχουν 3 μέθοδοι για το άνοιγμα της επιλογής φωτογραφίας. Η uploadProductPhoto() που καλείται με το που επιλεχθεί η φωτογραφική μηχανή, η openGallery() που θα ανοίξει το Gallery και η ο onActivityResult που θα εκτελεστεί μετά την ολοκλήρωση του Intent για το Gallery της συσκευής (Σχήμα 51). Στην πρώτη μέθοδο θα καλεστεί η openGallery() και θα αλλάξουμε στο FrameLayout που είναι το image το ύψος του height. Στην συνέχεια στην μέθοδο openGallery() δημιουργούμε ένα Intent για το Gallery και στην μέθοδο onActivity result, ανάλογα με το resultCode παραμετροποιούμε κατάλληλα την διεπιφάνεια κάνοντας ορατό ή αόρατο το κουμπί επιλογής της φωτογραφίας του προϊόντος και ορίζουμε στην διεπιφάνεια την φωτογραφία που διαλέξαμε.

```
private void uploadProductPhoto(View v) {

    getView().findViewById(R.id.manager_product_attachment_button).setVisibility(View.INVISIBLE);
    openGallery();
    getView().findViewById(R.id.manager_add_product_uploaded_photo).setVisibility(View.VISIBLE);

    int dps = 250;
    final float scale = getContext().getResources().getDisplayMetrics().density;
    int pixels = (int) (dps * scale + 0.5f);

    getView().findViewById(R.id.add_product_imageFL).getLayoutParams().height = pixels;
    getView().findViewById(R.id.add_product_imageFL).requestLayout();
}
private void openGallery() {
    Intent gallery = new Intent(Intent.ACTION_PICK, MediaStore.Images.Media.INTERNAL_CONTENT_URI);
    startActivityForResult(gallery, PICK_IMAGE);
}

@Override
public void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    getView().findViewById(R.id.manager_add_product_uploaded_photo).setVisibility(View.VISIBLE);
    if (resultCode == RESULT_OK && requestCode == PICK_IMAGE) {
        imageUri = data.getData();
        productImageView.setImageURI(imageUri);
        product_photo_added = true;
    }

    if (resultCode == RESULT_CANCELED) {

        getView().findViewById(R.id.manager_product_attachment_button).setVisibility(View.VISIBLE);
        getView().findViewById(R.id.add_product_imageFL).getLayoutParams().height = 0;
        getView().findViewById(R.id.add_product_imageFL).requestLayout();

    }
}
```

Σχήμα 51 Λήψη εικόνας από το Gallery του κινητού

4.20 Products Fragment, Custom ListView and Picasso Library



Σχήμα 52 Διεπαφή προβολής προϊόντων

Ένα άλλο Fragment που θα αναλυθεί είναι το ProductsFragment (Σχήμα 53) το οποίο θα κάνει εμφάνιση των προϊόντων. Όσον αφορά την εμφάνιση υπάρχει το manager_products_fragment αρχείο xml το οποίο περιέχει απλά ένα ListView το οποίο θα γεμίσει από τον ProductsAdapter. Η αρχικοποίηση του adapter γίνεται στο αρχείο ProductsFragment java. Σαν πεδία θα έχει 1 πίνακα products από αντικείμενα Product και άλλους 4 πίνακες titles, links, costs, quantities στους οποίους πίνακες διαχωρίζεται ο πίνακας αντικειμένων Products με την χρήση Streams. Ο πίνακας με τα

προϊόντα δημιουργείται από τον πλήρη δομητή της κλάσης του Fragment, όπου περνιέται σαν παράμετρος ένας πίνακας με προϊόντα. Αν δεν είναι περασμένα τα προϊόντα, θα εμφανιστούν όλα τα προϊόντα που υπάρχουν στο μαγαζί και επιστρέφονται από την μέθοδο getProducts() που κάνει ένα HTTP αίτημα στο Web Service όπως αναλύθηκε σε προηγούμενη ενότητα και θα επιστρέψει αντικείμενα Product. Στην συνέχεια δημιουργούμε ένα αντικείμενο ProductsAdapter που θα εμφανίσει κάθε διαφορετικό προϊόν. Η ανάθεση στις μεταβλητές του ProductsAdapter γίνεται στον πλήρη δομητή του και η ανάθεση στα TextView του κάθε product_row, γίνεται στην μέθοδο getView η οποία θα φορτώσει το περιεχόμενο για κάθε στοιχείο του ListView. Εδώ χρειάζεται να αναφερθεί η χρησιμοποίηση της βιβλιοθήκης Picasso με την οποία μπορούμε να φορτώσουμε κατευθείαν την εικόνα στο ImageView από το διαδίκτυο.

```
public class ProductsAdapter extends ArrayAdapter<String> {
    Context context;
    String[] rProductNames;
    String[] rCosts;
    String[] rImagesPath;
    int[] rQuantities;

    ProductsAdapter(Context c, Product[] products) {
        super(c, R.layout.product_row, R.id.productTitle_tv, titles);
        this.context = c;
        this.rProductNames = titles;
        this.rCosts = costs;
        this.rImagesPath = links;
        this.rQuantities = quantities;
    }

    @NonNull
    @Override
    public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
        LayoutInflater layoutInflater = (LayoutInflater) getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        View product_row = layoutInflater.inflate(R.layout.product_row, parent, attachToRoot: false);
        Log.d( tag: "ProductAdapter row loaded:", msg: "=>" + position);

        TextView productTitleTV =  product_row.findViewById(R.id.productTitle_tv);
        TextView productCostTV =  product_row.findViewById(R.id.productCostTv);
        ImageView productImage =  product_row.findViewById(R.id.product_image);
        TextView productQuantity = product_row.findViewById(R.id.quantityOfProduct);

        //set resource data to the row
        productCostTV.setText(rCosts[position]);
        productTitleTV.setText(rProductNames[position]);
        Picasso.with(getContext()).load( path: "http://" + rImagesPath[position]).into(productImage);
        productQuantity.setText(productQuantity.getText().toString() + quantities[position]);


        return product_row;
    }
}
```

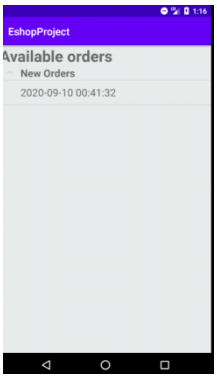
Σχήμα 53 Adapter προβολής προϊόντων

4.21 Orders Fragment and Custom ExpandableListView

Η επόμενη διεπιφάνεια που θα αναλυθεί είναι το Fragment OrdersFragment το οποίο περιλαμβάνει την λίστα των νέων παραγγελιών (Σχήμα 54). Το πρώτο μέρος είναι η σχεδίαση του View, το αρχείο xml managers_orders_fragment το οποίο είναι πολύ απλό στην υλοποίηση του καθώς περιλαμβάνει ένα root Element LinearLayout με vertical orientation και μέσα περιέχει 2 view, ένα TextView με τον τίτλο και ένα ExpandableListView για να εμφανίσει τις νέες παραγγελίες.

Το δεύτερο κομμάτι αφορά την λειτουργικότητα του Fragment που βρίσκεται στο αρχείο OrdersFragment. Η κλάση του Fragment περιλαμβάνει 4 πεδία. Ένα πεδίο ExpandableListView, ένα πεδίο mainAdapter τύπου OrdersAdapter που θα εμφανίσει την λίστα των παραγγελιών στο ExpandableListView. Για αρχή όπως έγινε σε προηγούμενο σημείο ένα HTTP αίτημα έτσι θα γίνει και εδώ ένα GET στο endpoint “/v1/Customers/Orders/New” με την χρήση της μεθόδου getOrders() η οποία επιστρέφει ότι έχει επιστρέψει ο Server σε ένα πίνακα αντικειμένων από Order αντικείμενα. Για την αρχικοποίηση των δεδομένων χρησιμοποιείται η μέθοδος initListData() η οποία θα προσθέσει

στην λίστα orderGroups το group “New Orders”, στην συνέχεια θα καλέσουμε την μέθοδο getOrders() που θα επιστρέψει τον πίνακα με τις παραγγελίες και από αυτόν τον πίνακα με χρήση Stream θα προσθέσουμε τις ημερομηνίες σε μια λίστα στο HashMap ordersPerGroup στο κλειδί “New Orders”. Τέλος στην μέθοδο onCreateView θα καλέσουμε την μέθοδο setAdapter για το expandableListView που ορίστηκε στο πεδίο της κλάσης



Σχήμα 54 Διεπαφή νέων παραγγελιών

Για την απεικόνιση των δεδομένων των παραγγελιών στην διεπαφή (Σχήμα 54), σε ένα ExpandableListView θα πρέπει να κάνουμε extend την κλάση BaseExpandableListAdapter στον OrdersAdapter, ο οποίος θα απεικονίσει τις παραγγελίες. Η κλάση περιλαμβάνει 4 πεδία, το context της εφαρμογής, μια λίστα από συμβολοσειρές (String) η οποία θα περιέχει τα ονόματα των κατηγοριών δηλαδή “New Orders”, το επόμενο πεδίο είναι το listItem το οποίο περιλαμβάνει ένα HashMap που έχει ως κλειδί (String) το όνομα του group και τελευταίο πεδίο είναι ένας πίνακας από παραγγελίες οι οποίες θα εμφανιστούν στο ExpandableListView.

Οι βασικές μέθοδοι οι οποίες κάνει override είναι η getGroupCount() που επιστρέφει τον αριθμό των groups, η getChildrenCount() η οποία ανάλογα με το groupPosition(), δηλαδή την θέση του Group μας επιστρέφει τον αριθμό των παιδιών του (Σχήμα 55). Αμέσως μετά είναι η μέθοδος getGroup() που επιστρέφει ανάλογα με το groupPosition() το αντικείμενο του group που θέλουμε, κάτι το οποίο κάνει και η getChild() η οποία επιστρέφει το παιδί με τους δείκτες θέσης του group και του child. Οι επόμενες 2 μέθοδοι είναι οι getGroupId() και getChildId() στις οποίες περνάμε παραμετρικά την θέση του group για την επιστροφή τού και τις θέσεις group και child για το παιδί. Η μέθοδος hasStableIds() επιστρέφει false έτσι ώστε κάθε φορά να ξαναδημιουργείται το View. Επίσης ορίζεται και η μέθοδος isChildSelectable() η οποία παίρνει παραμετρικά την θέση της ομάδας και την θέση του item για να ελέγξει αν αυτό είναι επιλέξιμο στην οποία μέθοδο επιστρέφουμε true για όλα τα items από όλα τα groups. Η είσοδος των δεδομένων στον adapter γίνεται με την χρήση μεθόδων get και set για το πεδίο orders το οποίο κρατάει τον πίνακα των παραγγελιών για εμφάνιση.

```
class OrdersAdapter extends BaseExpandableListAdapter {
    Context context; List<String> listGroup; HashMap<String, List<String>> listItem; Order[] orders;
    public OrdersAdapter(Context context, List<String> listGroup, HashMap<String, List<String>> listItem) {...}
    @Override
    public int getGroupCount() { return listGroup.size(); }
    @Override
    public int getChildrenCount(int groupPosition) {
        return this.listItem.get(this.listGroup.get(groupPosition)).size();
    }
    @Override
    public Object getGroup(int groupPosition) { return this.listGroup.get(groupPosition); }
    @Override
    public Object getChild(int groupPosition, int childPosition) {
        return this.listItem.get(this.listGroup.get(groupPosition)).get(childPosition);
    }
    @Override
    public long getGroupId(int groupPosition) { return groupPosition; }
    @Override
    public long getChildId(int groupPosition, int childPosition) { return 0; }
    @Override
    public boolean hasStableIds() { return false; }
    @Override
    public boolean isChildSelectable(int groupPosition, int childPosition) { return true; }
    @Override
    public View getGroupView(int groupPosition, boolean isExpanded, View convertView, ViewGroup parent) {...}
    @Override
    public View getChildView(int groupPosition, int childPosition, boolean isLastChild, View convertView, ViewGroup parent) {...}
    public Order[] getOrders() { return orders; }
    public void setOrders(Order[] orders) { this.orders = orders; }
}
```

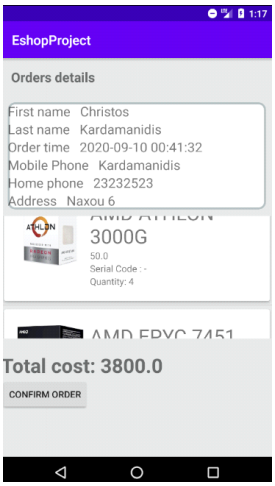
Σχήμα 55 Custom Expandable ListView Adapter Structure

Ιδιαίτερη βάση πρέπει να δοθεί στις μεθόδους getGroupView και getChildView οι οποίες επιστρέφουν το Group ή το child ανάλογα με το groupPosition και childPosition που θα δοθεί (Σχήμα 56). Στην πρώτη μέθοδο, ανάλογα με το groupPosition παίρνουμε με την μέθοδο το getGroup που αναφέρθηκε πιο πάνω το object και το μετατρέπουμε σε String group. Στην συνέχεια αν το convertView που έχει περαστεί στην μέθοδο είναι null θα πάμε να το δημιουργήσουμε με την χρήση του LAYOUT_INFLATER_SERVICE όπου δημιουργούμε ένα LayoutInflater αντικείμενο και με την μέθοδο inflate θα φτιάξουμε μια αναφορά στο list_group αρχείο xml το οποίο περιλαμβάνει την δομή του Group δηλαδή ένα TextView το οποίο θα εμφανίζει το όνομα του Group δηλαδή “New Orders”. Αμέσως μετά με την μέθοδο findViewById βρίσκω το parent TextView και κάνω ανάθεση τιμής το group. Η ίδια διαδικασία γίνεται και στην μέθοδο getChildView() η οποία περιλαμβάνει την επεξεργασία κάθε child του Group. Η διαφορά σε αυτήν είναι ότι ορίζεται ένας OnClickListener με την επιλογή του child για να μας κατευθύνει στο Fragment της προβολής παραγγελίας. Με την χρήση του getSupportFragmentManager() θα ξεκινήσουμε ένα transaction για να επιστρέψουμε στον container των Fragments το Fragment τύπου OrderAcceptFragment που φτιάξαμε παραπάνω και περνάμε στον πλήρη δομητή του την παραγγελία από το child που επιλέχθηκε.

```
@Override
public View getGroupView(int groupPosition, boolean isExpanded, View convertView, ViewGroup parent) {
    String group = (String) getGroup(groupPosition);
    if (convertView == null) {
        LayoutInflater layoutInflater = (LayoutInflater) this.context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        convertView = layoutInflater.inflate(R.layout.list_group,  root: null);
    }
    TextView textView = convertView.findViewById(R.id.list_parent);
    textView.setText(group);
    return  convertView;
}
@Override
public View getChildView(int groupPosition, int childPosition, boolean isLastChild, View convertView, ViewGroup parent) {
    String child = (String) getChild(groupPosition, childPosition);
    if ( convertView == null) {
        LayoutInflater layoutInflater = (LayoutInflater) this.context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        convertView = layoutInflater.inflate(R.layout.list_item,  root: null);
    }
    TextView textView = convertView.findViewById(R.id.list_child);
    textView.setText(child);
    textView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            OrderAcceptFragment acceptOrderFragment = new OrderAcceptFragment(orders[childPosition]);
            ((AppCompatActivity)context).getSupportFragmentManager().beginTransaction().
                    replace(R.id.manager_fragment_container, acceptOrderFragment).commitNow();
        }
    });
    return convertView;
}
```

Σχήμα 56 Μέθοδοι του getChildView(), getGroupView()

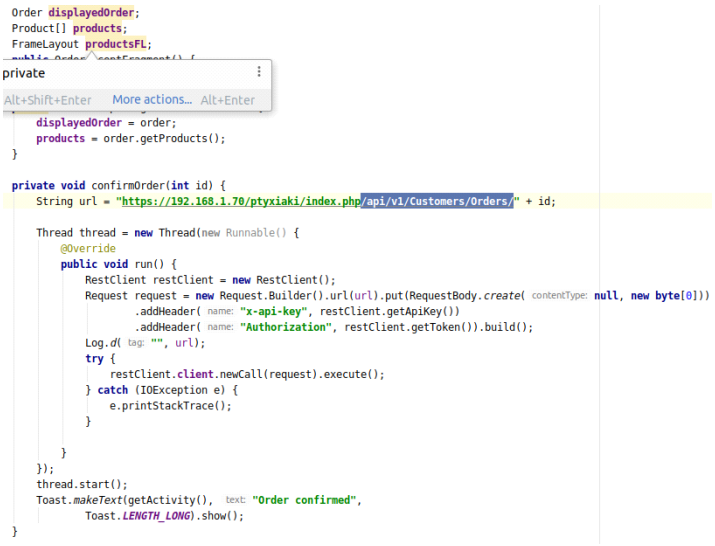
4.22 Order Accept Fragment



Σχήμα 57 Διεπιφάνεια έγκρισης παραγγελίας

Το επόμενο Fragment το οποίο θα αναλυθεί είναι το OrderAcceptFragment το οποίο εμφανίζει τα στοιχεία της παραγγελίας που θα είναι επιλεγμένη και θα εμφανίσει τα στοιχεία της όπως στην διεπιφάνεια στο Σχήμα 57. Το manager_fragment_accept_order.xml αποτελεί το view αυτού του Fragment, μέσα στο οποίο περιλαμβάνεται ένα root element ScrollView και μέσα περιλαμβάνει ένα LinearLayout με vertical orientation. Μέσα υπάρχουν τα TextView για το μικρό όνομα του πελάτη, το επώνυμο του, την ώρα της παραγγελίας, το σταθερό και κινητό τηλέφωνο του πελάτη καθώς και την διεύθυνση του.

Αμέσως παρακάτω καλείται το Fragment με τα προϊόντα της προβαλλόμενης παραγγελίας. Η προβαλλόμενη παραγγελία είναι σε ένα πεδίο τύπου Order μέσα στο OrderAcceptFragment και παίρνει τιμή από τον πλήρη δομητή της και τα προϊόντα που προβάλλονται είναι σε έναν πίνακα Products[] σαν πεδίο μέσα στην κλάση. Εκτός από τον πλήρη δομητή υπάρχουν και άλλες 3 μέθοδοι. Η μέθοδος confirmId() που παίρνει σαν παράμετρο το ID της παραγγελίας που θα επιβεβαιωθεί από το κατάστημα η οποία όπως στο Σχήμα 58 θα κάνει ένα PUT ερώτημα στο endpoint “/api/v1/Customers/Orders/{IdOfOrder}”. Για να γίνει το HTTP αίτημα θα ξεκινήσει ένα νέο Thread και με την χρήση του RestClient παραμετροποιούμε το αίτημα μας, του προσθέτουμε τα 2 headers ασφάλειας που έχουν οριστεί και εκτελούμε το call που κάνει confirm την παραγγελία



Σχήμα 58 HTTP PUT αίτημα

Στην συνέχεια ορίζουμε στον κώδικα της μεθόδου onCreate να αρχικοποιήσει το View περνώντας τα πεδία της εμφανιζόμενης παραγγελίας στην διεπιφάνεια όπως στο Σχήμα 59. Στην συνέχεια με την χρήση του getChildFragmentManager ξεκινάμε ένα transaction για να αντικαταστήσουμε στο FrameLayout με ID accept_order_productsFL ότι υπάρχει και κάνουμε replace με το productsFragment αντικείμενο το οποίο ορίσαμε πιο πάνω και θα εμφανίσει τα προϊόντα που του περνάμε σαν παράμετρο με την μορφή πίνακα.

Σχήμα 59 Προβολή προϊόντων παραγγελίας

4.23 Επίλογος

Σε αυτό το κεφάλαιο έγινε μια γενικότερη περιγραφή ολόκληρης της εφαρμογής Android. Αρχικά στο θεωρητικό της υπόβαθρο με τις διάφορες λεπτομέρειες όσον αφορά την αρχιτεκτονική του λειτουργικού του Android. Στην συνέχεια με τα βασικά εργαλεία για την ανάπτυξη της εφαρμογής του Android. Τέτοια εργαλεία είναι το Android Studio αλλά και το Grade. Στην συνέχεια έγινε μια λεπτομερή ανάλυση για όλα τα επίπεδα της εφαρμογής. Αρχικά ως προς τα διάφορα αρχεία τα οποία περιλαμβάνει. Αρχεία, όπως Resources, icons και διάφορα άλλα, μέχρι τα αρχεία xml τα οποία περιλαμβάνουν την σχεδίαση της διεπαφής αλλά και της μεταφοράς των δεδομένων από το Web Service στην εφαρμογή και την απεικόνιση τους στην διεπαφή.

Κεφάλαιο 5ο: Συμπεράσματα και προτάσεις βελτίωσης

Σε αυτήν την ενότητα αναλύονται τα συμπεράσματα της διπλωματικής εργασίας και παρουσιάζονται προτάσεις βελτίωσης της. Συνοψίζοντας το περιεχόμενο της πτυχιακής, κατά την υλοποίηση έγινε λεπτομερής ανάλυση σε τρία βασικά επίπεδα. Το πρώτο περιελάμβανε την μοντελοποίηση των δεδομένων σε μια βάση δεδομένων MariaDb όπου θα αποθήκευε τα δεδομένα του ηλεκτρονικού καταστήματος. Το δεύτερο επίπεδο συγκαταλέγει την χρήση PHP και ενός Apache Web Server για την δημιουργία ενός Web Service το οποίο θα αποκωδικοποιούσε τα αιτήματα HTTP και θα αλληλεπιδρούσε με την βάση δεδομένων. Η μεταφορά των δεδομένων αναφέρθηκε ότι γίνεται μέσω του HTTP. Με αυτόν τον τρόπο η εφαρμογή Android μπορεί να επικοινωνεί με την βάση δεδομένων. Στην συνέχεια κρίσιμο ήταν να αναλυθεί ο τρόπος της Android εφαρμογής να επικοινωνεί με το Web Service αλλά και να μετατρέψει τα δεδομένα σε αντικείμενα Java. Τέλος στην Android εφαρμογή έγινε παρουσίαση των διάφορων Activities και Fragments τα οποία περιλαμβάνουν τις βασικές λειτουργικότητες των administrator και των managers. Σε αυτό το σημείο, εφόσον έγινε η πλήρης ανάλυση της πτυχιακής σε όλα τα επίπεδα, είναι ανάγκη να γίνει αναφορά και σε μελλοντικές βελτιώσεις που μπορεί να γίνουν στην λειτουργία του λογισμικού που δημιουργήθηκε. Κάποιες λειτουργίες οι οποίες θα μπορούσαν να προστεθούν αφορούν τις αναφορές που βλέπει ο manager. Για παράδειγμα, θα μπορούσαν αυτές οι ράβδοι με τα επίπεδα πωλήσεων να μην χρειάζεται να τα δει ο manager μέσω της εφαρμογής, αλλά να μπορεί να κάνει εξαγωγή αυτών των στατιστικών σε κάποιο αρχείο PDF και να μπορεί να αποθηκευτεί και στον αποθηκευτικό χώρο της συσκευής του κινητού. Έτσι θα είναι διαθέσιμο και χωρίς ο χρήστης να κάνει υποχρεωτικά σύνδεση στην εφαρμογή. Επίσης εναλλακτικά θα ήταν χρήσιμο αυτό το PDF να σταλεί σε κάποιο email που θα έχει ορίσει ο χρήστης από την εφαρμογή. Έτσι θα μπορεί να στέλνεται η αναφορά και σε άλλους εργαζόμενους του μαγαζιού. Επίσης μια άλλη επιλογή που θα μπορούσε να υπάρξει είναι η δημιουργία μιας λίστας πελατολογίου. Αυτή η λίστα θα μπορούσε να περιέχει τα στοιχεία όλων των πελατών του ηλεκτρονικού μαγαζιού. Αυτή η αναφορά όπως και η προηγούμενη θα ήταν χρήσιμη και για τους διευθύνοντες του μαγαζιού και για την εξαγωγή συμπερασμάτων. Έτσι θα μπορούσε ο υπάλληλος να διαμορφώσει στρατηγικές πωλήσεων σε σχέση με το ποιοι πελάτες προτιμούν το ηλεκτρονικό κατάστημα και ποια είναι τα κριτήρια τους για αυτήν τους την επιλογή. Κάτι τέτοιο θα μπορούσε να γίνει με χρήση αλγορίθμων μηχανικής μάθησης για να βρεθούν ποιοι είναι οι λόγοι που ο κάθε πελάτης διαλέγει το κάθε προϊόν. Έτσι με όλη αυτήν την γνώση, θα μπορούσε να δημιουργηθεί ένα recommendation system το οποίο ανάλογα με τα στοιχεία των προηγούμενων πελατών που προτίμησαν αυτό το προϊόν να συστηθούν μελλοντικά προϊόντα που θα ήθελε να διαλέξει ο πελάτης. Επίσης με βάση αυτά τα στοιχεία θα μπορούσε να γίνει μια μελλοντική εκτίμηση βέλτιστης οικονομικής βελτίωσης της επιχείρησης, Επίσης με χρήση αυτής της γνώσης μπορεί ο manager να διαλέγει ποιο προϊόν να έχει προτεραιότητα στην λίστα των προτεινόμενων προϊόντων του πελάτη. Τέλος, εκτός από τους πελάτες, θα μπορούσε να δημιουργηθεί μια επιλογή για εξαγωγή και των προϊόντων σε PDF, το οποίο θα μπορούσε να αποσταλεί σε κάποιο email. Σε αυτό το email θα μπορούσαν να περιέχονται τα στοιχεία όλων των προϊόντων, όπως οι τιμές τους, τα επίπεδα πωλήσεων τους, αλλά και οι εικόνες τους. Επίσης, όσον αφορά τις αναφορές θα μπορούσε να γίνει αποστολή τους στο διαδίκτυο και να αποθηκευτούν και αυτές στον Web Server όπου βρίσκεται το Web Service. Έτσι θα μπορούν αυτές οι πληροφορίες που είναι σε μορφή αρχείου, να είναι άμεσα διαθέσιμες και για μελλοντικές εφαρμογές οι οποίες θα χτιστούν και θα χρησιμοποιούν το Web Service που υλοποιήθηκε. Όπως ήδη αναφέρθηκε, το Web Service μπορεί να χρησιμοποιηθεί και από εφαρμογές οι οποίες τρέχουν σε πολλά διαφορετικά λειτουργικά συστήματα. Γενικά η δημιουργία των

Web Services αλλά και της εφαρμογής Android ήταν ιδιαίτερα χρήσιμη διότι βοηθάει τον προγραμματιστή να καταλαβαίνει βασικά ζητήματα αρχιτεκτονικής των εφαρμογών αλλά και πως αυτά επηρεάζουν την καλύτερη εμπειρία του χρήστη των εφαρμογών που συνδέονται σε αυτό. Χαρακτηριστικό παράδειγμα ήταν μαζική αποστολή και επιστροφή δεδομένων η οποία χρειαζόταν ιδιαίτερη προσοχή από τον προγραμματιστή στου χρόνους απόκρισης οι οποίοι σε άλλη περίπτωση θα οδηγούσαν τον χρήστη την εφαρμογής σε μεγάλη αναμονή, ακόμα και κατάρρευση της.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Βιβλία

[1] L. Welling and L. Thomson, *Ανάπτυξη Web Εφαρμογών με PHP και MySQL*, 4th ed. M. Γκιούρδας 2020.

[2] W. Stallings and L. Brown, *ΑΣΦΑΛΕΙΑ ΥΠΟΛΟΓΙΣΤΩΝ ΑΡΧΕΣ ΚΑΙ ΠΡΑΚΤΙΚΕΣ,* 3rd ed. ΕΚΔΟΣΕΙΣ ΚΛΕΙΔΑΡΙΘΜΟΣ.

[3] W. STALLINGS, *ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ Αρχές Σχεδίασης,* 8th ed. ΕΚΔΟΣΕΙΣ ΤΖΙΟΛΑ.

[4] J. KUROSE and K. ROSS, *Computer Networking A Top-Down Approach,* 6th ed. Pearson.

Internet Site

[5] “PHP: Hypertext Preprocessor”, *Php.net*. [Online]. Available: https://www.php.net.

[6] “Open Source Database RDBMS for the Enterprise | MariaDB”, *Mariadb.com*. [Online]. Available: https://www.mariadb.com.

[7] “Android Developers”, *Android Developers.* [Online]. Available: https://www.mariadb.com.

[8]"Home - HTTPD - Apache Software Foundation", *Cwiki.apache.org*. [Online]. Available: https://cwiki.apache.org/confluence/display/HTTPD/Home.

[9]"HTTP Request Message - HTTP Requests Documentation", *Documentation.help*. [Online]. Available: https://documentation.help/DogeTool-HTTP-Requests-vt/http_request.htm.

[10]"Welcome! - The Apache HTTP Server Project", *Httpd.apache.org*, 2020. [Online]. Available: https://httpd.apache.org/.

[11]"What is a web server?", *MDN Web Docs*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server.

[12]I. Square, "OkHttp", *Square.github.io*. [Online]. Available: https://square.github.io/okhttp/.

[13]"google/gson", *GitHub*. [Online]. Available: https://github.com/google/gson.

[14]"PhilJay/MPAndroidChart", *GitHub*. [Online]. Available: https://github.com/PhilJay/MPAndroidChart.

[15]"Royalty-free vector clip art, svg files and graphics | Public domain vectors", *Publicdomainvectors.org*. [Online]. Available: https://publicdomainvectors.org.

ΠΑΡΑΡΤΗΜΑ Α:ΚΩΔΙΚΑΣ ΓΙΑ WEB SERVICES

CustomSQL Resource

```
if ($method == "POST") {
    if (isset($json['query'])) {

        $query = $json['query'];
        $rows = array();

        if ($result = $mysqli->query($query)) {
            while ($r = $result->fetch_assoc()) {
                $rows[] = $r;
            }
        }
        print json_encode($rows, JSON_PRETTY_PRINT);
    }
}
```

Employees Resource, Sessions Subresource

```
if ($method == "DELETE") {
    $sql = "UPDATE sessions SET STATUS = ? , SESSION_END = ? WHERE USERTOKKEN LIKE ? ";
    if (!($stmt = $mysqli->prepare($sql))) {
        print "Prepared failed:(" . $mysqli->errno . ") " . $mysqli->error;
    }
    $status = "CLOSED";
    $date = date("Y-m-d H:i:s");
    $tokken = $json['userTokken'];
    if (!($stmt->bind_param("sss", $status , $date, $tokken))) {
        print "Binding parameters failed:(" . $stmt->errno . ")" . $stmt->error;
    }
    if (!$stmt->execute()) {
        print "Execute failed:(" . $stmt->errno . ")" . $stmt->error;
    }

    if ($stmt->affected_rows == 1) {
        header('HTTP/1.1 204 Resource updated successfully');
    }
    else if ($stmt->affected_rows == 0) {
        echo json_encode(array('error' => "Session could not be closed. Invalid tokken passed"));

    }
  }
}
```

Categories Resource

```
else if ($method == "DELETE") {
    if (sizeof($json) > 1) {
        header('HTTP/1.1 400 Bad Request');
    }
    if(isset($json['categoryname'])) {
        $categoryname = $json['categoryname'];
    }
    if (isset($categoryname)) {
        $sql = "DELETE FROM CATEGORIES WHERE CATEGORYNAME=?";
        $stmt = $mysqli->prepare($sql);
        $stmt->bind_param("s", $categoryname);
        if($stmt->execute()) {
            $stmt->close();
            header('HTTP/1.1 204 No Content');
        } else {
            $stmt->close();
            header('HTTP/1.1 500 Internal Server Error');
        }
    } else {
        $mysqli->query("DELETE FROM CATEGORIES");
        header('HTTP/1.1 200 OK');
    }
}
```

```
else if ($method == "POST") {
    if(isset($json['categoryname'])) {
        $categoryname = $json['categoryname'];
    } else {
        header('HTTP/1.1 400 Bad Request');
        exit;
    }

    $sql = "INSERT INTO CATEGORIES(CATEGORYNAME) VALUES(?)";
    $stmt = $mysqli->prepare($sql);
    $stmt->bind_param("s", $categoryname);
    if($stmt->execute()) {
        echo json_encode(array('categoryid' => "$mysqli->insert_id"));
        $stmt->close();
        header('HTTP/1.1 201 Created');
    } else {
        header('HTTP/1.1 500 Internal Server Error');
    }
}
```

Products Resource

```
else if ($method == "POST") {
    $product_title = $json['product_title'];
    $product_cost = $json['product_cost'];
    $product_image = $json['product_image'];
    $categoryid = $json['categoryid'];

    $productImageSavePath = "./ProductsImages/" . "$product_title" . ".jpg";
    base64ToJpeg($json['product_image'], $productImageSavePath);
    $sql = "INSERT INTO Products(PRODUCT_TITLE, PRODUCT_COST, PRODUCT_IMAGE, ISMODIFIED, CATEGORYID)";
    $sql .= " VALUES(?, ?, ?, false, ?)";
    $stmt = $mysqli->prepare($sql);
    $stmt->bind_param('sdsi', $product_title, $product_cost, $productImageSavePath, $categoryid);
    if($stmt->execute()) {
        $stmt->close();
        echo json_encode(array('pid' => "$mysqli->insert_id"));
        header('HTTP/1.1 201 Created');
    } else {
        $stmt->close();
        header('HTTP/1.1 500 Internal Server Error');
    }
}
```

Customer Resource

Session Subresource

```
if ($method == "DELETE") {
    $sql = "UPDATE sessions SET STATUS = ? , SESSION_END = ? WHERE USERTOKKEN LIKE ? ";
    if (!($stmt = $mysqli->prepare($sql))) {
        print "Prepared failed:(" . $mysqli->errno . ") " . $mysqli->error;
    }
    $status = "CLOSED";
    $date = date("Y-m-d H:i:s");
    $tokken = $json['userTokken'];
    $errorOccured = false;
    if (!($stmt->bind_param("sss", $status , $date, $tokken))) {
        $errorOccured = true;
    }
    if (!$stmt->execute()) {
        $errorOccured = true;
    }
    if ($errorOccured) {
        header('HTTP/1.1 500 Internal Server Error');
    }
    if ($stmt->affected_rows == 1) {
        header('HTTP/1.1 204 No Content');
    }
    else if ($stmt->affected_rows == 0) {
        header('HTTP/1.1 500 Internal Server Error');
        echo json_encode(array('error' => "Session could not be closed. Invalid tokken passed"));
    }
}
```

```php
if ($subresource == "Sessions") {
    if ($method == "POST") {
        $username = $json['username'];
        $password = $json['password'];
        $errorOccured = false;
        $sql = "SELECT * FROM Users WHERE USERNAME = ? AND PASSWORD = ?";
        if (!($stmt = $mysqli->prepare($sql))) {
            $errorOccured = true;
        }
        if (!$stmt->bind_param("ss", $username, $password)) {
            $errorOccured = true;
        }
        if (!$stmt->execute()) {
            $errorOccured = true;
        }
        $result = $stmt->get_result();
        $matchedUsers = $result->num_rows;
        $row = $result->fetch_assoc();
        $userid = $row['USERID'];
        $stmt->close();

        if ($matchedUsers == 1) {
            $sql = "INSERT INTO Sessions(STATUS, USER_ID, SESSION_START, USERTOKKEN) VALUES(?,?,?,?)";
            if (!($stmt = $mysqli->prepare($sql))) {
                print "Prepared failed:(" . $mysqli->errno . ") " . $mysqli->error;
            }
            $status="CONNECTED";
            $date = date("Y-m-d H:i:s");
            $tokken = md5(date("Y-m-d H:i:s"));

            if (!($stmt->bind_param("siss", $status, $userid, $date, $tokken))) {
                print "Binding parameters failed:(" . $stmt->errno . ")" . $stmt->error;
            }
            if (!$stmt->execute()) {
                print "Execute failed:(" . $stmt->errno . ")" . $stmt->error;
            }
            if ($errorOccured) {
                header('HTTP/1.1 500 Internal Server Error');
            }
            echo json_encode(array('userTokken' => "$tokken"));
            header('HTTP/1.1 201 Created');
            $stmt->close();
        }
        else if ($matchedUsers == 0) {
            header('HTTP/1.1 500 Internal Server Error');
            echo json_encode(array('error' => "Invalid credentials"));
        }
    }
```

Orders Subresource

```php
if ($method == "GET") {
    if ($subsubresource == "New") {
        $result = mysqli_query($mysqli, "CALL GetNewOrders()");
        if($result) {
            $counter = 0;
            $return_data = array();
            while ($row = $result->fetch_assoc()) {
                $query = "CALL GetOrderProducts(" .  $row['ORDERID'] . ")";
                $mysqli->next_result();
                $productResults = mysqli_query($mysqli, $query);
                $productsIndata = array();
                print $mysqli->error;
                if($productResults) {
                    while ($row_product = $productResults->fetch_assoc()) {

                    $productsIndata[] = array ("productid" => $row_product["PRODUCTID"],
                    "product_title" => $row_product["PRODUCT_TITLE"],
                    "quantity" => $row_product["QUANTITY"],
                    "productCost" => $row_product["PRODUCT_COST"],
                    "productImage" =>  $_SERVER['SERVER_ADDR'] . "/ptyxiaki" . substr($row_product['PRODUCT_IMAGE'], 1));
                    }
                }

                $return_data[] = array(
                            "orderid" => $row['ORDERID'],
                            "shopid" => $row['SHOPID'],
                            "orderType" => $row['ORDERTYPE'],
                            "state" => $row['STATE'],
                            "orderTime" => $row['ORDER_TIMESTAMP'],
                            "orderComment" => $row['ORDER_COMMENT'],
                            "user" => array (
                                "lastname" => $row['LAST_NAME'],
                                "firstname" => $row['FIRST_NAME'],
                                "primaryemail" => $row['PRIMARY_EMAIL'],
                                "address" => $row['ADDRESS'],
                                "mobilephone" => $row['MOBILEPHONE'],
                                "homephone" => $row['HOMEPHONE']
                                    ),
                            "products" => $productsIndata
                            );
            }
            }
            echo json_encode($return_data, JSON_UNESCAPED_SLASHES | JSON_PRETTY_PRINT);
        exit;
    }
    if ($subsubresource == "All") {
}
```

```
if ($subsubresource == "New") {
if ($subsubresource == "All") {
    $result = mysqli_query($mysqli, "CALL GetOrders()");
        if($result) {
            $counter = 0;
            $return_data = array();
            while ($row = $result->fetch_assoc()) {
                $query = "CALL GetOrderProducts(" .  $row['ORDERID'] . ")";
                $mysqli->next_result();
                $productResults = mysqli_query($mysqli, $query);
                $productsIndata = array();
                print $mysqli->error;
                if($productResults) {
                    while ($row_product = $productResults->fetch_assoc()) {
                        $productsIndata[] = array ("productid" => $row_product["PRODUCTID"],
                                                   "product_title" => $row_product["PRODUCT_TITLE"],
                                                   "quantity" => $row_product["QUANTITY"],
                                                   "productCost" => $row_product["PRODUCT_COST"]);
                    }
                }
                $return_data[] = array("orderid" => $row['ORDERID'],
                                       "shopid" => $row['SHOPID'],
                                       "orderType" => $row['ORDERTYPE'],
                                       "state" => $row['STATE'],
                                       "orderTime" => $row['ORDER_TIMESTAMP'],
                                       "orderComment" => $row['ORDER_COMMENT'],
                                       "user" => array (
                                          "lastname" => $row['LAST_NAME'],
                                          "firstname" => $row['FIRST_NAME'],
                                          "primaryemail" => $row['PRIMARY_EMAIL'],
                                          "address" => $row['ADDRESS'],
                                          "mobilephone" => $row['MOBILEPHONE'],
                                          "homephone" => $row['HOMEPHONE']),
                                       "products" => $productsIndata
                                       );
            }
        }
            echo json_encode($return_data, JSON_UNESCAPED_SLASHES | JSON_PRETTY_PRINT);
    exit;

}

if ($subresource == "Orders") {
  $orderState = "undefined";
  if ($subsubresource == "NewOrdersCheck") {
    $orderState = "New";
    $result = mysqli_query($mysqli, "CALL CheckForNewOrders();");
    if($result) {
      $return_data = array();
      $row = $result->fetch_assoc();
      echo json_encode(array('newOrders' => $row['newOrdersCounter']), JSON_PRETTY_PRINT);
    }
    exit;
  } else if ($subsubresource == "Product") {
    if ($method == "POST") {
      $sql = "INSERT INTO ORDER_DETAILS(ORDERID, PRODUCTID, QUANTITY)";
      $sql .= " VALUES(?, ?, ?)";
      if (!($stmt = $mysqli->prepare($sql))) {
        $error = "Prepared failed:(" . $mysqli->errno . ") " . $mysqli->error;
      }
      if (!$stmt->bind_param("iii", $json['orderid'], $json['productid'], $json['quantity'])) {
        $error = "Binding parameters failed:(" . $stmt->errno . ")" . $stmt->error;
      }
      if (!$stmt->execute()) {
        $error = "Execute failed:(" . $stmt->errno . ")" . $stmt->error;
        $stmt->close();
        $pattern = "Duplicate entry for";
        if (preg_match("/Execute failed/", $error)) {
          $sql = "UPDATE ORDER_DETAILS SET QUANTITY = QUANTITY + ? WHERE PRODUCTID = ? AND ORDERID = ?";
          if(!($stmt = $mysqli->prepare($sql))) {
            $error = "Prepared failed:(" . $mysqli->errno . ") " . $mysqli->error;
          }
          $stmt->bind_param("iii", $json['quantity'], $json['productid'], $json['orderid']);
          $stmt->execute();
          header('HTTP/1.1 201 Created');
          exit;
        }
      }
      $rowid = $mysqli->insert_id;
      if ($stmt->affected_rows == 1) {
        echo json_encode(array("orderid" => "$rowid"));
        header('HTTP/1.1 201 Created 201');
      }
      else {
        header('HTTP/1.1 500 Internal Server Error');
        echo json_encode(array('error' => "Product could not be posted " . $error));
      }
      exit;

    }
  }
```

Registration Subresource

```php
if ($subresource == "Registration") {
    if ($method == "POST") {
        $username = $json['username'];
        $firstname = $json['firstname'];
        $lastname = $json['lastname'];
        $password = $json['password'];
        $primaryemail = $json['primaryemail'];
        $secondaryemail = $json['secondaryemail'];
        $address = $json['address'];
        $gender = $json['gender'];
        $mobilephone = $json['mobilephone'];
        $homephone = $json['homephone'];
        $sql = "INSERT INTO Users(USERNAME, FIRST_NAME, LAST_NAME, PASSWORD, PRIMARY_EMAIL, ";
        $sql .= "SECONDARY_EMAIL, ADDRESS, GENDER, MOBILEPHONE, HOMEPHONE) VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
        $errorOccured = false;
        if (!($stmt = $mysqli->prepare($sql))) {
            $errorOccured = true;
        }
        if (!$stmt->bind_param("sssssssss", $username, $firstname, $lastname, $password, $primaryemail, $secondaryemail, $address, $gender, $mobilephone, $homephone)) {
            $errorOccured = true;
        }
        if (!$stmt->execute()) {
            $errorOccured = true;
        }
        $rowid = $mysqli->insert_id;
        $newsletter = $json['newsletter'];
        $stmt->close();
        $sql = "INSERT INTO Customers(CUSTOMERID, NEWSLETTER, BANNED) VALUES(?, ?, false)";
        if (!($stmt = $mysqli->prepare($sql))) {
            $errorOccured = true;
        }
        if (!$stmt->bind_param("is", $rowid, $newsletter)) {
            $errorOccured = true;
        }
        if (!$stmt->execute()) {
            $errorOccured = true;
        }
        if ($errorOccured) {
            header('HTTP/1.1 500 Internal Server Error');
        }
        if ($stmt->affected_rows == 1) {
            echo json_encode(array("customerid" => "$rowid"));
            header('HTTP/1.1 201 Created');
        }
        else {
            header('HTTP/1.1 500 Internal Server Error');
            echo json_encode(array('error' => "Employee could not be posted"));
        }
        $stmt->close();
    }
```

Administrators Resource

Shop Subresource

```php
if ($subresource == "Shops") {
    if ( $method == "GET") {
        $result = mysqli_query($mysqli, "SELECT * FROM  SHOPS");
        if($result) {
            $counter = 0;
            $return_data = [];
            $productDetails = array();
            while ($row = $result->fetch_assoc()) {
                $productDetails[] = array("SHOPID" => $row['SHOPID'], "ADDRESS" => $row['ADDRESS'], "OWNER_FIRST_NAME" => $row['OWNER_FIRST_NAME'],
                                          "OWNER_LAST_NAME" => $row['OWNER_LAST_NAME'], "SHOPNAME" => $row['SHOPNAME']);
            }
        echo json_encode($productDetails, JSON_PRETTY_PRINT);
        }
    }
    if ( $method == "POST") {
        $address = $json["address"];
        $ownerFirstName = $json["firstName"];
        $ownerLastName = $json["lastName"];
        $shopname = $json["shopname"];

        $sql = "INSERT INTO SHOPS(ADDRESS,OWNER_FIRST_NAME,OWNER_LAST_NAME,SHOPNAME) VALUES(?, ?, ?, ?)";

        if (!$stmt = $mysqli->prepare($sql)) {
            print "Prepared failed:(" . $mysqli->errno . ") " . $mysqli->error;
        }
        if (!$stmt->bind_param("ssss", $address, $ownerFirstName, $ownerLastName, $shopname)) {
            print "Binding parameters failed:(" . $stmt->errno . ")" . $stmt->error;print "Binding parameters failed:(" . $stmt->errno . ")" . $stmt->error;
        }

        if($stmt->execute()) {
            echo json_encode(array('shopid' => "$mysqli->insert_id"));
            $stmt->close();
            header('HTTP/1.1 200 OK');
        } else {
            print $stmt->error;
            header('HTTP/1.1 500 Internal Server Error');
        }
    }
}
```

Employees subresource

```
if ($subresource == "Employees") {
    if ( $method == "GET") {
        $sql = "SELECT * FROM employees INNER JOIN users ON employees.EMPLOYEEID = users.USERID ";

        if ($param == "Managers") {
            $sql .= "WHERE ROLE = 'Manager'";
        } else if ($param == "Salesmen") {
            $sql .= "WHERE ROLE = 'Salesman'";
        } else if ($param == "Warehousemen") {
            $sql .= "WHERE ROLE = 'Manager'";
        } else if ($param != "undefined") {
            header("HTTP/1.0 404 Not Found");
            exit;
            }

        if (isset( $_GET['shopid'])) {
            $sql .= " AND employees.WORKS_IN = " . "'" .  $_GET['shopid'] . "'";
        }
        $result = mysqli_query($mysqli,$sql);
        if($result) {
            $counter = 0;
            $return_data = [];
            $employeeDetails = array();
            while ($row = $result->fetch_assoc()) {
                $employeeDetails[] = array("employeeid" => $row['EMPLOYEEID'],
                                           "username" => $row['USERNAME'],
                                           "firstname" => $row['FIRST_NAME'],
                                           "lastname" => $row['LAST_NAME'],
                                           "password" => $row['PASSWORD'],
                                           "primaryemail" => $row['PRIMARY_EMAIL'],
                                           "secondaryemail" => $row['SECONDARY_EMAIL'],
                                           "address" => $row['ADDRESS'],
                                           "gender" => $row['GENDER'],
                                           "mobilephone" => $row['MOBILEPHONE'],
                                           "homephone" => $row['HOMEPHONE'],
                                           "title" => $row['TITLE'],
                                           "identity" => $row['IDENTITY'],
                                           "startOfContract" => $row['START_OF_CONTRACT'],
                                           "endOfContract" => $row['END_OF_CONTRACT'],
                                           "role" => $row['ROLE'],
                                           "worksIn" => $row['WORKS_IN']);


            }
            echo json_encode($employeeDetails, JSON_PRETTY_PRINT);
            }
    }
```

```
if ($subresource == "Employees") {
    if ( $method == "GET") {
    if ( $method == "POST") {
        $role = "role";
        if ($param == "Managers") { $role = "Manager";}
        if ($param == "Salesmen") { $role = "Salesman";}
        if ($param == "Warehousemen") { $role = "Warehouseman";}
        $username = $json['username'];
        $firstname = $json['firstname'];
        $lastname = $json['lastname'];
        $password = $json['password'];
        $primaryemail = $json['primaryemail'];
        $address = $json['address'];
        $gender = $json['gender'];
        $secondaryemail = $json['secondaryemail'];
        $mobilephone = $json['mobilephone'];
        $homephone = $json['homephone'];
        $sql = "INSERT INTO Users(USERNAME, FIRST_NAME, LAST_NAME, PASSWORD, PRIMARY_EMAIL, SECONDARY_EMAIL, ADDRESS, GENDER, MOBILEPHONE, HOMEPHONE)";
        $sql .= " VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
        if (!($stmt = $mysqli->prepare($sql))) {
            print "Prepared failed:(" . $mysqli->errno . ") " . $mysqli->error;
        }
        if (!$stmt->bind_param("ssssssssss", $username, $firstname, $lastname, $password, $primaryemail, $secondaryemail, $address, $gender, $mobilephone, $homephone)) {
            print "Binding parameters failed:(" . $stmt->errno . ")" . $stmt->error;
        }
        if (!$stmt->execute()) {
            print "Execute failed:(" . $stmt->errno . ")" . $stmt->error;
        }
        $rowid = $mysqli->insert_id;
        $stmt->close();
        $title = $json['title'];
        $identity = $json['identity'];
        $startOfContract = $json['startOfContract'];
        $endOfContract = $json['endOfContract'];
        $worksIn = $json['worksIn'];
        $sql = "INSERT INTO Employees(EMPLOYEEID, TITLE, IDENTITY, START_OF_CONTRACT, END_OF_CONTRACT, ROLE, WORKS_IN) VALUES(?, ?, ?, ?, ?, ?, ?)";
        if (!($stmt = $mysqli->prepare($sql))) {
            print "Prepared failed:(" . $mysqli->errno . ") " . $mysqli->error;
        }
        if (!$stmt->bind_param("issssss", $rowid, $title, $identity, $startOfContract, $endOfContract, $role, $worksIn)) {
            print "Binding parameters failed:(" . $stmt->errno . ")" . $stmt->error;
        }
        if (!$stmt->execute()) {
            print "Execute failed:(" . $stmt->errno . ")" . $stmt->error;
        }
        if ($stmt->affected_rows == 1) {
            echo json_encode(array("employeeid" => "$rowid"));
            header('HTTP/1.1 201 Created');
        }
        else {
            echo json_encode(array('error' => "Employee could not be posted"));
            header('HTTP/1.1 500 Internal Server Error');
        }
    }
    if ( $method == "DELETE") {
    }
}
```

Sessions subresource

```php
if ( $method == "POST") {
    $sql = "SELECT * FROM administrators WHERE USERNAME = ? AND PASSWORD = ?";
    if (!($stmt = $mysqli->prepare($sql))) {
        print "Prepared failed:(" . $mysqli->errno . ") " . $mysqli->error;
    }
    if (!$stmt->bind_param("ss", $username, $password)) {
        print "Binding parameters failed:(" . $stmt->errno . ")" . $stmt->error;
    }
    if (!$stmt->execute()) {
        print "Execute failed:(" . $stmt->errno . ")" . $stmt->error;
    }

    $result = $stmt->get_result();
    $matchedUsers = $result->num_rows;
    $row = $result->fetch_assoc();
    $adminid = $row['ADMIN_ID'];
    $stmt->close();

    if ($matchedUsers == 1) {
        $sql = "INSERT INTO Adminsessions(STATUS, ADMIN_ID, SESSION_START, USERTOKKEN) VALUES(?,?,?,?)";

        if (!($stmt = $mysqli->prepare($sql))) {
            print "Prepared failed:(" . $mysqli->errno . ") " . $mysqli->error;
        }
        $status="CONNECTED";
        $date = date("Y-m-d H:i:s");
        $tokken = md5(date("Y-m-d H:i:s"));

        if (!($stmt->bind_param("siss", $status, $adminid, $date, $tokken))) {
            print "Binding parameters failed:(" . $stmt->errno . ")" . $stmt->error;
        }
        if (!$stmt->execute()) {
            print "Execute failed:(" . $stmt->errno . ")" . $stmt->error;
        }
        echo json_encode(array('administratorTokken' => "$tokken"));
        $stmt->close();
    }
    else if ($matchedUsers == 0) {
        echo json_encode(array('error' => "Invalid credentials"));

    }
}
if ( $method == "DELETE") {
    $sql = "UPDATE adminsessions SET STATUS = ? , SESSION_END = ? WHERE USERTOKKEN LIKE ? ";
    if (!($stmt = $mysqli->prepare($sql))) {
        print "Prepared failed:(" . $mysqli->errno . ") " . $mysqli->error;
    }
    $status = "CLOSED";
    $date = date("Y-m-d H:i:s");
    if (!($stmt->bind_param("sss", $status , $date, $tokken))) {
        print "Binding parameters failed:(" . $stmt->errno . ")" . $stmt->error;
    }
    if (!$stmt->execute()) {
        print "Execute failed:(" . $stmt->errno . ")" . $stmt->error;
    }
    if ($stmt->affected_rows == 1) {
        header('HTTP/1.1 204 Resource updated successfully');
    }
    else if ($stmt->affected_rows == 0) {
        echo json_encode(array('error' => "Session could not be closed. Invalid tokken passed"));
    }
}
if ( $method == "GET") {
    $result = mysqli_query($mysqli, "SELECT * FROM adminsessions S INNER JOIN administrators A ON S.ADMIN_ID = A.ADMIN_ID");
    if($result) {
        $counter = 0;
        $return_data = [];
        $sessionDetails = array();
        while ($row = $result->fetch_assoc()) {
            $sessionDetails[] = array("sessionid" => $row['SESSION_ID'], "status" => $row['STATUS'], "username" => $row['USERNAME'],
            "sessionStart" => $row['SESSION_START'], "sessionEnd" => $row['SESSION_END'], "usertokken" => $row['USERTOKKEN']);
        }
    echo json_encode($sessionDetails, JSON_PRETTY_PRINT);
    }

}
```

ΠΑΡΑΡΤΗΜΑ Β:ΚΩΔΙΚΑΣ Android

XML LoginActivity

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/back_gradient">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="430dp"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:orientation="vertical">
        <EditText
                android:layout_width="300dp"
                android:layout_height="50dp"
                android:id="@+id/usernameET"
                android:background="@drawable/textbox_back"
                android:drawableLeft="@drawable/ic_person_black_24dp"
                android:hint="Username"
                android:textColorHint="#000000"
                android:padding="5dp"
                android:layout_marginLeft="10dp"/>
        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:layout_marginTop="20dp">
            <EditText
                android:layout_width="300dp"
                android:layout_height="50dp"
                android:id="@+id/passwordET"
                android:background="@drawable/textbox_back"
                android:hint="Password"
                android:textColorHint="#000000"
                android:drawableLeft="@drawable/ic_lock_outline_black_24dp"
                android:inputType="textPassword"
                android:layout_marginLeft="10dp"/>
            <ImageButton
                android:layout_width="70dp"
                android:layout_height="50dp"
                android:src="@drawable/ic_remove_red_eye_black_24dp"
                android:background="#0000"
                android:id="@+id/show_password_button" />
        </LinearLayout>
        <Button
            android:id="@+id/button_login"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Login"
            android:background="@drawable/login_button_back"
            android:layout_marginTop="20dp"
            android:layout_marginLeft="10dp"
            android:textColor="#FFFF" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/login_error"/>
    </LinearLayout>
</RelativeLayout>
```

AdministratorActivity

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".AdministratorManagement">
    <FrameLayout
        android:id="@+id/administrator_fragment_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_above="@id/administrator_bottomnvigationtoolbar"/>
    <com.google.android.material.bottomnavigation.BottomNavigationView
        android:id="@+id/administrator_bottomnvigationtoolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:background="?android:attr/windowBackground"
        app:menu="@menu/administrator_bottom_navigation_bar"/>
</RelativeLayout>
```

ShopsFragment

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="20dp"
    android:background="#FFFFFF">

    <ImageView
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:src="@drawable/ic_shops_icon"
        android:background="@drawable/circle" />


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="SHOPNAME"
            android:id="@+id/shopname"
            android:textStyle="normal"
            android:textSize="20sp"/>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="OWNER NAME AND LAST NAME"
            android:id="@+id/ownerfullname"
            android:textStyle="normal"
            android:textSize="10sp"/>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="OWNER TELEPHONE"
            android:id="@+id/ownertelephone"
            android:textStyle="normal"
            android:textSize="10sp"/>
    </LinearLayout>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <ListView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/shops_listview"
        android:background="#C5C4C4"></ListView>
    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:src="@drawable/ic_add_black_24dp"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:id="@+id/shop_add"
        android:layout_marginBottom="20dp"
        />
    <ImageButton
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:src="@drawable/ic_refresh_black_24dp"
        android:visibility="invisible"
        android:id="@+id/shopsRefreshBtn"
        style="@style/Widget.AppCompat.ImageButton"
        />
    <ProgressBar
        style="?android:attr/progressBarStyleLarge"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/shop_progressbar"
        android:progressDrawable="@drawable/ic_refresh_black_24dp"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        />
</RelativeLavout>
```

ManagersFragment

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <Spinner
            android:id="@+id/shop_spinner"
            android:layout_width="match_parent"
            android:layout_height="50dp"
            android:focusedByDefault="false"
            android:spinnerMode="dialog">
        </Spinner>
        <ListView
            android:id="@+id/managers_listview"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:listSelector="#D1E3E0" />
    </LinearLayout>
    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/manager_add"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="20dp"
        android:src="@drawable/ic_add_black_24dp"
        />
</RelativeLayout>
```

```
private Employee[] retrieveEmployees(String roleOfEmployee , int shopid) {
    Gson gson = new Gson();
    JSONArray employees = geEmployees(roleOfEmployee, shopid);
    if (employees == null) {
        return null;
    }
    final Employee[] employeesArray = gson.fromJson(employees.toString(), Employee[].class);
    return employeesArray;

}
private JSONArray geEmployees(String roleOfEmployee, int shopid) {
    RestClient rc = new RestClient();
    try {
        return rc.doGetRequest( url: "http://192.168.1.70/ptyxiaki/index.php/api/v1/Administrators/Employees/" + roleOfEmployee + "?shopid=" + shopid);
    } catch (SocketTimeoutException e) {
        e.printStackTrace();
    } catch (JSONException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return null;
}
```

```
private void displayShopManagers(int shopid, String shopname) {
    Employee[] managers = retrieveEmployees( roleOfEmployee: "Managers", shopid);
    String[] usernames = managers != null ? Arrays.stream(managers).map(s->s.getUsername()).toArray(size->new String[managers.length]) : new String[]{};
    handler1 = new Handler(getActivity().getApplicationContext().getMainLooper());
    handler1.post(new Runnable() {
        @Override
        public void run() {
            ManagersAdapter managersAdapter = new ManagersAdapter(getContext(), managers, usernames);
            managersListView.setAdapter(managersAdapter);
            managersListView.destroyDrawingCache();
            managersListView.setVisibility(ListView.INVISIBLE);
            managersListView.setVisibility(ListView.VISIBLE);
        }
    });
}
```

```
private JSONArray getShops() {
    RestClient rc = new RestClient();
    try {
        return rc.doGetRequest( url: "http://192.168.1.70/ptyxiaki/index.php/api/v1/Administrators/Shops");
    } catch (SocketTimeoutException e) {
        e.printStackTrace();
    } catch (JSONException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return null;
}
private Shop[] retrieveAllShops() {
    Gson gson = new Gson();
    Log.d( tag: "Time started retrieving shops",  msg: "==>" +System.currentTimeMillis());
    JSONArray shops = getShops();
    Log.d( tag: "Time stopped retrieving shops",  msg: "==>" +System.currentTimeMillis());
    if (shops == null) {
        return null;
    }
    final Shop[] shoparray = gson.fromJson(shops.toString(), Shop[].class);

    return shoparray;

}
```

Open in browser (Ctrl+Click, Ctrl+B)

ServerFragment

```
private JSONObject getDetails() throws InterruptedException {
    final CountDownLatch latch = new CountDownLatch(1);
    final JSONObject[] jsonObject = new JSONObject[1];
    HandlerThread thread = new HandlerThread( name: "NetworkThread") {
        @Override
        public void run() {
            RestClient restClient = new RestClient();
            try {
                jsonObject[0] = restClient.doGetRequestSingleObject( url: "https://192.168.1.70/ptyxiaki/index.php/api/v1/ServerDetails");
                latch.countDown();
            } catch (IOException e) {
                e.printStackTrace();
            } catch (JSONException j) {
                j.printStackTrace();
            }
        };
    };
    thread.start();
    latch.await();
    return jsonObject[0];
}
```

```
private void initBugReportForm(View view) {
    bugReportB = (Button) view.findViewById(R.id.bugSubmit);
    bugReportB.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            reportBugET = (EditText) view.findViewById(R.id.bugText);
            if (!reportBugET.getText().toString().isEmpty()) {
                submitBug(reportBugET.getText().toString());
                Snackbar snackbar = Snackbar.make(view,  text: "Thanks for your feedback", Snackbar.LENGTH_SHORT);
                snackbar.show();
            }
            else {
                Snackbar snackbar = Snackbar.make(view,  text: "No content submitted", Snackbar.LENGTH_SHORT);
                snackbar.show();
            }
        }
    });
}
```

ProductsFragment

```
private Product[] getProducts() throws InterruptedException {
    final Product[][] products = new Product[1][1];
    CountDownLatch latch = new CountDownLatch(1);
    Thread thread = new Thread(new Runnable() {
        @Override
        public void run() {
            RestClient restClient= new RestClient();
            JSONArray jsonArray = null;
            try {
                jsonArray = restClient.doGetRequest( url: "https://192.168.1.70/ptyxiaki/index.php/api/v1/Products");
                Gson gson = new Gson();
                Log.d( tag: "", jsonArray.toString());
                products[0] = gson.fromJson(jsonArray.toString(), Product[].class);
            } catch (JSONException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
            latch.countDown();
        }
    });

    thread.start();
    latch.await();
    return products[0];
}
```

OrdersFragment

```
private Order [] getOrders(final String orderState) throws InterruptedException {

    final CountDownLatch latch = new CountDownLatch(1);
    final JSONArray[] jsonArrays = new JSONArray[1];

    HandlerThread handlerThread = new HandlerThread( name: "NetworkThread") {
        @Override
        public void run() {
            RestClient restClient = new RestClient();
            try {
                jsonArrays[0] = restClient.doGetRequest( url: "http://192.168.1.70/ptyxiaki/index.php/api/v1/Customers/Orders/New");
                latch.countDown();
            } catch (JSONException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    };
    handlerThread.start();
    latch.await();
    Gson gson = new Gson();
    final Order[] ordersArray = gson.fromJson(jsonArrays[0].toString(), Order[].class);
    return ordersArray;
}
```

ManagerActivity

```
private boolean logout(String tokken) throws JSONException, InterruptedException {
    final JSONObject[] jsonObject = new JSONObject[1];
    jsonObject[0] = new JSONObject();
    jsonObject[0].put( name: "userTokken", tokken);
    final CountDownLatch latch = new CountDownLatch(1);
    HandlerThread thread = new HandlerThread( name: "NetworkThread") {
        @Override
        public void run() {
            try {
                RestClient restClient = new RestClient();
                restClient.doDeleteRequest( url: "http://192.168.1.70/ptyxiaki/index.php/api/v1/Employees/Sessions", jsonObject[0]);
                if (restClient.httpCode == 204) {
                    SharedPreferences preferences = getSharedPreferences( name: "PREFERENCE", Context.MODE_PRIVATE);
                    SharedPreferences.Editor editor = preferences.edit();
                    editor.clear();
                    editor.commit();
                    finish();
                    Intent menuIntent = new Intent(getApplicationContext(), LoginActivity.class);
                    startActivity(menuIntent);
                }
            } catch (IOException i) {
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        Toast.makeText(getBaseContext(), text: "Could not connect to server", Toast.LENGTH_LONG);
                    }
                });
            } catch (JSONException j) {
                Toast.makeText(getBaseContext(), text: "Unexpected response from server", Toast.LENGTH_LONG);
                j.printStackTrace();
            }
            latch.countDown();
        }
    };
    thread.start();
    latch.await();
    return true;
}
```

Open in browser (Ctrl+Click, Ctrl+B)

AddShopFragment

```
private void addShop(Shop shop) throws IOException, JSONException {
    JSONObject object = new JSONObject();

    object.put( name: "shopname", shop.getShopname());
    object.put( name: "address", shop.getAddress());
    object.put( name: "firstName", shop.getOwnerFirstName());
    object.put( name: "lastName", shop.getOwnerLastName());

    Log.d( tag: "Shop to be inserted(OBJECT)", shop.toString());
    Log.d( tag: "Shop to be inserted(JSONObject)",object.toString());
    RestClient restClient = new RestClient();
    restClient.doPost( url: "http://192.168.1.70/ptyxiaki/index.php/api/v1/Administrators/Shops", object);

}
```

AddManagerFragment

```
void addManager(Employee employee, int shopid) throws IOException,JSONException {
    JSONObject object = new JSONObject();

    object.put( name: "endOfContract", employee.getEndOfContract());
    object.put( name: "startOfContract", employee.getStartOfContract());
    object.put( name: "title", employee.getTitle());
    object.put( name: "identity", employee.getIdentity());
    object.put( name: "worksIn", shopid);
    object.put( name: "username", employee.getUsername());
    object.put( name: "firstname", employee.getFirstName());
    object.put( name: "lastname", employee.getLastName());
    object.put( name: "password", employee.getPassword());
    object.put( name: "primaryemail", employee.getPrimaryemail());
    object.put( name: "secondaryemail", employee.getSecondaryemail());
    object.put( name: "address", employee.getAddress());
    object.put( name: "mobilephone", employee.getMobilephone());
    object.put( name: "homephone", employee.getHomephone());
    object.put( name: "gender", employee.getGender());
    Log.d( tag: "Manager to be inserted(OBJECT)", employee.toString());
    Log.d( tag: "Manager to be inserted(JSONObject)",object.toString());
    RestClient restClient = new RestClient();
    restClient.doPost( url: "http://192.168.1.70/ptyxiaki/index.php/api/v1/Administrators/Employees/Managers", object);
}
```

AddProductFragment

```
private void postProduct(Product product, int categoryid) throws InterruptedException, JSONException {

    byte[] image_bytes = product.getByteImage();
    String base64image = Base64.encodeToString(image_bytes, Base64.NO_WRAP);
    JSONObject postJsonObject = new JSONObject();
    postJsonObject.put( name: "categoryid", categoryid);
    postJsonObject.put( name: "product_title", product.getTitle());
    postJsonObject.put( name: "product_cost", product.getCost());
    postJsonObject.put( name: "product_image", base64image);
    postJsonObject.put( name: "quantity", product.getQuantity());

    final CountDownLatch latch = new CountDownLatch(1);
    final JSONObject[] jsonObject = new JSONObject[1];
    HandlerThread thread = new HandlerThread( name: "NetworkThread") {
        @Override
        public void run() {
            RestClient restClient = new RestClient();
            try {
                jsonObject[0] = restClient.doPost( url: "http://192.168.1.70//ptyxiaki/index.php/api/v1/Products", postJsonObject);
                latch.countDown();
            } catch (IOException e) {
                e.printStackTrace();
                showResultDialogBox( productPosted: false);
            } catch (JSONException j) {
                j.printStackTrace();
                showResultDialogBox( productPosted: false);
            }
        };
    };
    thread.start();
    latch.await();
    if (jsonObject[0].get("pid") != null) {
        showResultDialogBox( productPosted: true);
    }
}
```

```
private JSONArray getCategories() throws InterruptedException {
    final CountDownLatch latch = new CountDownLatch(1);
    final JSONArray[] jsonArray = new JSONArray[1];
    HandlerThread thread = new HandlerThread( name: "NetworkThread") {
        @Override
        public void run() {
            RestClient restClient = new RestClient();
            try {
                jsonArray[0] = restClient.doGetRequest( url: "http://192.168.1.70/ptyxiaki/index.php/api/v1/Categories");
                latch.countDown();
            } catch (IOException e) {
                e.printStackTrace();
                Log.d( tag: "Connection:", msg: " Could not be established");
                return;

            } catch (JSONException j) {
                j.printStackTrace();
            }
        };
    };
    thread.start();
    latch.await();
    return jsonArray[0];
}
private Category[] getAllCategories() throws InterruptedException {
    Gson gson = new Gson();
    JSONArray categories = getCategories();
    try {
        JSONArray jsonObject = getCategories();
    } catch (InterruptedException e) { }
    final Category[] categoriesArray = gson.fromJson(categories.toString(), Category[].class);
    return categoriesArray;
}
```

ChartsFragment

```
private HashMap <String, Double> getProfitPerDate () throws InterruptedException {
    HashMap <String, Double> stringIntegerHashMap = new HashMap <~>();
    RestClient restClient = new RestClient();
    final CountDownLatch latch = new CountDownLatch(1);
    Thread thread = new Thread(new Runnable() {
        @Override
        public void run() {
            try {

                JSONArray ordersJsonArray = restClient.doGetRequest( url: "http://192.168.1.70/ptyxiaki/index.php/api/v1/Customers/Orders/All");
                Gson gson = new Gson();
                final Order[] ordersArray = gson.fromJson(ordersJsonArray.toString(), Order[].class);
                for (Order o: ordersArray) {

                    double sum = 0 ;
                    for (Product p: o.getProducts()) {
                        sum += p.getCost();
                    }

                    String key = o.getOrderTime().split( regex: " ")[0];
                    if (!stringIntegerHashMap.containsKey(key))
                        stringIntegerHashMap.put(key, sum);
                    else
                        stringIntegerHashMap.put(key, stringIntegerHashMap.get(key).doubleValue() + sum);

                   Log.i( tag: "Total cost",  msg: " " + sum + " for date" + key);
                }
                latch.countDown();
            } catch (JSONException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    });
    thread.start();
    latch.await();
    return stringIntegerHashMap;
}
```

```
barChart = view.findViewById(R.id.barChart);

ArrayList<BarEntry> heights = new ArrayList<~>();
ArrayList<String> xAxisLabel = new ArrayList<String>();

AtomicInteger i = new AtomicInteger();
try {
    getProfitPerDate().entrySet().forEach(stringDoubleEntry -> {
        heights.add(new BarEntry(i.getAndIncrement(), stringDoubleEntry.getValue().intValue()));
        xAxisLabel.add(stringDoubleEntry.getKey());

    });
} catch (InterruptedException e) {
    e.printStackTrace();
}
BarDataSet barDataSet = new BarDataSet(heights, label: "Dataset 1");
barDataSet.setColors(ColorTemplate.COLORFUL_COLORS);
Description description = new Description();
description.setText("Days");
barChart.setDescription(description);
BarData barData = new BarData(barDataSet);
barChart.setData(barData);
//we need to set XAXIS value formatter
XAxis xAxis = barChart.getXAxis();
xAxis.setValueFormatter(new IAxisValueFormatter() {
    @Override
    public String getFormattedValue(float value, AxisBase axis) {
        return xAxisLabel.get((int) value); }});
xAxis.setPosition(XAxis.XAxisPosition.TOP);
xAxis.setDrawGridLines(true);
xAxis.setDrawAxisLine(true);
xAxis.setGranularity(1f);
xAxis.setLabelCount(getBarChartData().size());
xAxis.setLabelRotationAngle(270);
barChart.animateY( durationMillis: 3000);
barChart.invalidate();
```