Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτη

Σχολή Τεχνολογικών Εφαρμογών

Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή Εργασία

Τίτλος: Τουριστικός οδηγός με καθοδήγηση GPS σε περιβάλλον Android

Μιχάλης Μαρούλης (ΑΜ:2728)
Ειρήνη Χαϊδεμενάκη (ΑΜ:2706)

Επιβλέπων καθηγητής:Παπαδάκης Νικόλαος

Επιτροπή Αξιολόγησης:Παπαδάκης, Τζαγκαράκης, Ρουσσάκης

Ημερομηνία Παρουσίασης:17/03/2015

Ευχαριστίες

Με την ολοκλήρωση της πτυχιακής μας εργασίας, θα θέλαμε να ευχαριστήσουμε τον επιβλέποντα καθηγητή κ. Παπαδάκη Νικόλαο για την καθοδήγηση και την επίβλεψη της εργασίας μας. Τέλος, ένα μεγάλο ευχαριστώ στις οικογένειες μας, για τη στήριξη καθ΄ολη την διάρκεια των σπουδών μας.

Abstract

Tourist guide with GPS guidance on Android environment

The aim of the thesis is to develop an application on Android environment, supported by mobile phones (smart phones), which are based on the Android operating system. The application is designed to serve the users easy guidance and entertainment for a specific area.

The application serves to fulfill various functions, such as search, online booking with the respective prices- offers giving possibility to cancel a reservation and finding alternatives to hotels, car rental offices, excursion agencies, ferry and air routes as well as simple information for museums and attractions.

For the application are used Web Service technologies, MySql databases, and the application development program for the Android platforms, Android Studio, which includes the Java programming language and XML language. Also, the Android SDK Tools (software development kit) was used, which includes the necessary libraries (software libraries), a mobile device simulator (emulator) and the debugger code (debugger).

Περιεχόμενα

Πίνακας Εικόνων

1 Εισαγωγή

1.1 Περίληψη

Στόχος της πτυχιακής είναι η ανάπτυξη μίας εφαρμογής σε περιβάλλον Android, η οποία υποστηρίζεται από κινητά τηλέφωνα (smart phones), τα οποία είναι βασισμένα στο λειτουργικό σύστημα Android. Η εφαρμογή αποσκοπεί στην εξυπηρέτηση των χρηστών για εύκολη καθοδήγηση και ψυχαγωγία για μία συγκεκριμένη περιοχή.

Η εφαρμογή εξυπηρετεί στο να εκτελούνται διάφορες λειτουργίες, όπως αναζήτηση, online κρατήσεις με τις αντίστοιχες τιμές-προσφορές δίνοντας δυνατότητα ακύρωσης μίας κράτησης, αλλά και εύρεση εναλλακτικών λύσεων για ξενοδοχεία, γραφεία ενοικιάσεων οχημάτων, εκδρομικά γραφεία, δρομολόγια ακτοπλοϊκών και αεροπορικών γραμμών. Επίσης ο χρήστης μπορεί να αποσπάσει απλές πληροφορίες για τα μουσεία και τα αξιοθέατα της πόλης.

Στην εφαρμογή χρησιμοποιήθηκαν τεχνολογίες Web Service, MySql βάσεις δεδομένων, καθώς και το πρόγραμμα ανάπτυξης εφαρμογών για τις πλατφόρμες Android, Android Studio, το οποίο περιλαμβάνει τη γλώσσα προγραμματισμού Java και τη γλώσσα σήμανσης XML. Έγινε χρήση του Android SDK Tools (software development kit), το οποίο περιλαμβάνει τις απαραίτητες βιβλιοθήκες (software libraries), έναν προσομοιωτή κινητής συσκευής (emulator) και το πρόγραμμα εντοπισμού σφαλμάτων του κώδικα (debugger).

1.2 Γνωριμία με το Android

1.2.1 Τι είναι το Android

Το Android είναι ένα λειτουργικό σύστημα που απευθύνεται κυρίως σε κινητές συσκευές, όπως smart phones και tablets με οθόνες αφής. Βασισμένο στον πυρήνα του Linux και με γνώμονα ότι είναι μια open source πλατφόρμα, δίνει τη δυνατότητα σε μηχανικούς λογισμικού να γράφουν κώδικα και να δημιουργούν εφαρμογές (applications), με τη χρήση της αντικειμενοστραφής γλώσσας προγραμματισμού Java και με τη βοήθεια βιβλιοθηκών λογισμικού (software libraries) της Google.

Εικόνα 1: Λογότυπο Android

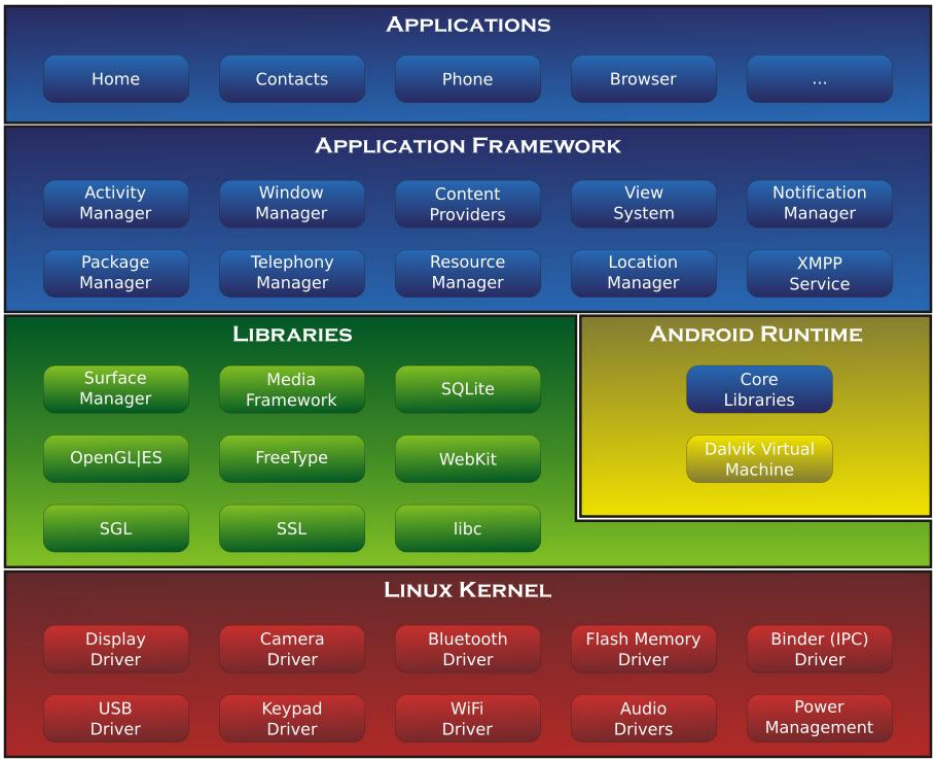
1.2.2 Ιστόρια του Android

Η ιδέα του Android ξεκίνησε από τον Andy Rubin ως ένα λειτουργικό σύστημα για κινητά τηλέφωνα γύρω στα τέλη του 2003. Έπειτα από δύο χρόνια η πρωτοποριακή αυτή ιδέα έφτασε στα αυτιά ενός έκτον δύο ιδρυτών της Google, όπου και αποφάσισαν να εξαγοράσουν την ιδέα του Rubin. Σαν αποτέλεσμα το project του Rubin, το επονομαζόμενο Android Inc, ανήκει πλέον στη Google από το 2005, η οποία δουλεύοντας σιωπηλά για τα επόμενα δύο χρόνια πάνω στο Android, το 2007 ανακοινώνει δημόσια ότι δούλευε σε ένα Open Source λειτουργικό σύστημα κινητής τηλεφωνίας. Την ίδια χρονιά και έπειτα από την παραπάνω ανακοίνωση της Google, ιδρύεται ένας οργανισμός ονόματι Open Handset Alliance στον οποίο συμμετέχουν δεκάδες εταιρίες του τεχνολογικού κόσμου, όπως η HTC, LG, Intel, Samsung και πολλές άλλες, οι οποίες σε συνεργασία με τη Google, έχουν ως σκοπό να κυκλοφορήσουν στην αγορά τα πρώτα smart phones με λειτουργικό σύστημα Android. Όπως και έγινε ένα χρόνο μετά από την HTC, με την εμφάνιση του HTC Dream.

Εικόνα 2: Λογότυπο Open Handset Alliance

1.3 Αρχιτεκτονική του Android

Όπως κάθε λειτουργικό σύστημα έτσι και το Android πρέπει να παρέχει μία διαφάνεια στον τρόπο διασύνδεσης του υλικού και των εκτελούμενων εφαρμογών-προγραμμάτων. Αυτό δίνει τη δυνατότητα στο χρήστη να εκμεταλλευτεί τους πόρους του συστήματος με τρόπο κατανοητό και εύκολο προς αυτόν. Συγκεκριμένα το Android αποτελείται από μία στοίβα λογισμικών τμημάτων (software stack), όπως θα δούμε παρακάτω. Για να καταλάβουμε καλύτερα το σχήμα, θα πρέπει να κοιτάξουμε μέσα σε κάθε επίπεδο ξεχωριστά ξεκινώντας από το κατώτερο.



Εικόνα 3: Android Architecture

1.3.1 Πυρήνας

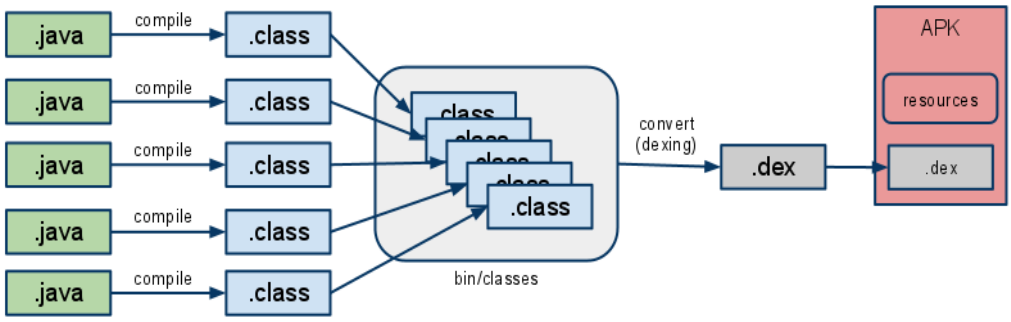
Στο τελευταίο επίπεδο της στοίβας βρίσκεται ο πυρήνας του Android, βασισμένος στον πυρήνα Kernel του Linux τροποποιημένος κατάλληλα από τη Google για κινητές συσκευές. Ο πυρήνας αυτός έχει μεγάλη συμβατότητα με ποικίλες αρχιτεκτονικές επεξεργαστών. Αυτό τον καθιστά συμβατό με μεγάλο εύρος συσκευών που μπορούν να υποστηρίξουν το λειτουργικό σύστημα του Android. Ο πυρήνας είναι υπεύθυνος για τη διασύνδεση του hardware της κάθε συσκευής με το λειτουργικό σύστημα, παρέχοντας τα κατάλληλα προγράμματα οδήγησης (hardware drivers). Ενδεικτικά μερικές λειτουργίες του πυρήνα είναι η διαχείριση της μνήμης, η διαχείριση ενέργειας, η διαχείριση της κάμερας, η δικτύωση και αρκετές άλλες.

1.3.2 Βιβλιοθήκες και χρόνος εκτέλεσης

Στο αμέσως επόμενο επίπεδο βρίσκονται οι βασικές βιβλιοθήκες του συστήματος, γνωστές ως Εγγενής Βιβλιοθήκες (Native Libraries), οι οποίες είναι γραμμένες σε γλώσσα προγραμματισμού C και C++. Οι βιβλιοθήκες αυτές μπορούν να ενσωματωθούν από εφαρμογές έτσι ώστε μία εφαρμογή να εκμεταλλευτεί τις λειτουργίες που παρέχει η κάθε βιβλιοθήκη. Ουσιαστικά αποτελούν τα API's (Application Programming Interface) που είναι διαθέσιμα στους προγραμματιστές και δεν πρέπει να

θεωρούνται αυτόνομες εφαρμογές. Τα API's διαχωρίζονται βάση του level, το οποίο είναι ένας ακέραιος αριθμός και ανάλογα με τον τύπο της συσκευής και την έκδοση του λειτουργικού συστήματος, ο προγραμματιστής έχει στη διάθεση του συγκεκριμένες βιβλιοθήκες και αντικείμενα (objects).

Στο ίδιο επίπεδο συναντάμε το τμήμα Android Runtime το οποίο περιέχει τις βασικές βιβλιοθήκες της Java (core libraries) καθώς και την εικονική μηχανή Dalvik. Οι εφαρμογές που απευθύνονται στο Android είναι γραμμένες στη γλώσσα προγραμματισμού Java και μέσω του compiler Java Virtual Machine παράγονται bytecodes τα οποία έπειτα μεταφράζονται σε αρχεία .dex (Dalvik EXecutable) και .odex (Optimized Dalvik Executable) με τη χρήση της εικονικής μηχανής Dalvik. Τα αρχεία αυτά έχουν υποστεί συμπίεση και έχουν σχεδιαστεί για συστήματα με περιορισμένη μνήμη και επεξεργαστική ισχύ, προσφέροντας αποδοτικότερη διαχείριση μνήμης και καλύτερη διαχείριση ενέργειας. Κάθε εφαρμογή που εκτελείται, χρησιμοποιεί ένα δικό της στιγμιότυπο της Dalvik VM (Virtual Machine) και είναι ανεξάρτητη από άλλες εφαρμογές που εκτελούνται παράλληλα, εξασφαλίζοντας την ομαλή λειτουργία του συστήματος. Τέλος, τα παραπάνω αρχεία περιέχονται στα πακέτα εγκατάστασης εφαρμογών APK (Android Application Package) μαζί με άλλα χρήσιμα αρχεία όπως πόρους (resources) και το αρχείο περιορισμών (manifest file) και έχουν την κατάληξη .apk.



Εικόνα 4: Δημιουργία APK πακέτου

1.3.3 Πλαίσιο εφαρμογών

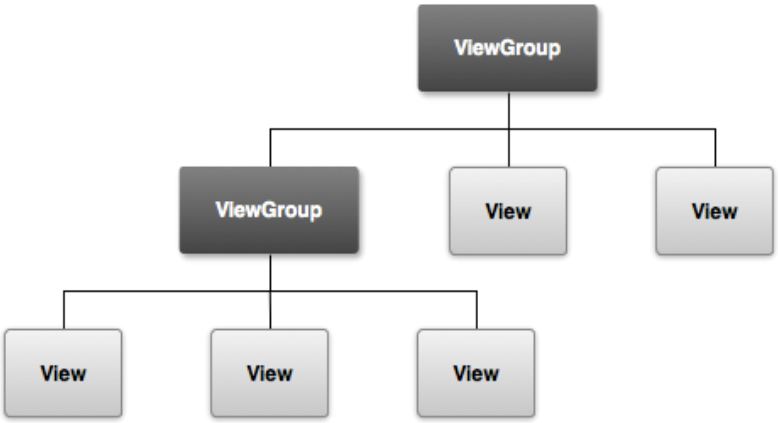
Το επίπεδο Application Framework απευθύνεται κυρίως στους προγραμματιστές λογισμικού. Σε αυτό το επίπεδο συναντάμε όλες εκείνες τις δυνατότητες που θα μπορεί να ενσωματώσει ο προγραμματιστής στην εφαρμογή του, μέσω των διαφόρων API's που έχει στη διάθεσή του. Από τη μεγάλη ποικιλία των API's στο επίπεδο αυτό θα αναφερθούμε στα σημαντικότερα.

Η κλάση Activity Manager ανήκει στο πακέτο android.app και διαχειρίζεται τις εφαρμογές που εκτελούνται κάθε φορά, καθώς και τον κύκλο ζωής της κάθε εφαρμογής (αναλυτικά για τον κύκλο ζωής μιας εφαρμογής, σε επόμενο κεφάλαιο). Για περισσότερες πληροφορίες της κλάσης Activity Manager μπορείτε ανατρέξετε στον παρακάτω σύνδεσμο: http://developer.android.com/reference/android/app/ActivityManager.html

Η κλάση Content Provider ανήκει στο πακέτο android.content και είναι ένα από τα κύρια δομικά στοιχεία του Android, το οποίο επιτρέπει το διαμοιρασμό δεδομένων από μία εφαρμογή σε μία άλλη. Το Android περιλαμβάνει προκαθορισμένα κάποιους παρόχους περιεχομένου που διαχειρίζονται δεδομένα όπως στοιχεία επικοινωνίας (επαφές χρήστη), ήχο, βίντεο και φωτογραφίες. Όταν μία εφαρμογή χρειάζεται δεδομένα από μία άλλη, τότε πρέπει να κάνει αίτηση στην εφαρμογή η οποία είναι εξουσιοδοτημένη ως πάροχος περιεχομένου, χρησιμοποιώντας την κλάση ContentResolver. Για περισσότερες πληροφορίες επισκεφτείτε τους παρακάτω συνδέσμους:
http://developer.android.com/reference/android/content/ContentProvider.html
http://developer.android.com/reference/android/content/ContentResolver.html

Η πιο σημαντική κλάση στο Android είναι η κλάση View, η οποία ανήκει στο πακέτο android.view. Στο πακέτο αυτό περιέχονται διεπαφές χρήστη και κλάσεις οι οποίες σχετίζονται με τη σχεδίαση στην οθόνη. Όλα τα ορατά στοιχεία μιας διεπαφής ανήκουν στην κλάση View (Buttons,

TextViews κ.α.). Εκτός των ορατών στοιχείων, υπάρχουν και μη ορατά στοιχεία οθόνης που ανήκουν στην κλάση ViewGroup και ένα παράδειγμα είναι τα στοιχεία διάταξης της οθόνης (layouts). Το ViewGroup αναπαριστά ένα ορθογώνιο τμήμα στην οθόνη της συσκευής και μπορεί να περιέχει άλλα αντικείμενα View (βλ. εικόνα). Επιπλέον στο πακέτο android.view συμπεριλαμβάνονται όλα τα interfaces (listeners), τα οποία είναι υπεύθυνα για την αλληλεπίδραση της εφαρμογής με το χρήστη, ανιχνεύοντας τα συμβάντα (events). Ένα τέτοιο παράδειγμα είναι το πάτημα ενός κουμπιού. Περισσότερα για το android.view: http://developer.android.com/reference/android/view/package-summary.html



Εικόνα 5: Ιεραρχία ενός ViewGroup

1.3.4 Εφαρμογές

Στην κορυφή της στοίβας συναντάμε το επίπεδο των εφαρμογών (Applications) στο οποίο λαμβάνουν χώρα όλες οι εφαρμογές που είναι προεγκατεστημένες στη συσκευή (browser, contacts list κ.α.), καθώς και εφαρμογές που έχει εγκαταστήσει ο χρήστης για τη δική του εξυπηρέτηση. Το επίπεδο αυτό είναι το επίπεδο το οποίο αλληλεπιδρά άμεσα με το χρήστη, προσφέροντάς του τη δυνατότητα να περιηγηθεί στη μεγάλη ποικιλία των εφαρμογών που διαθέτει ο κόσμος του Android.

1.4 Εκδόσεις και χαρακτηριστικά του Android

Η ιστορία του λειτουργικού Android ξεκινάει κατά την περίοδο 2007-2008 με τις εκδόσεις Alpha και Beta, οι οποίες ήταν πειραματικές (pre-commercial versions) και δεν κυκλοφόρησαν ποτέ στην αγορά. Η πρώτη επίσημη έκδοση ήταν η 1.0 με API level 1, η οποία παρουσιάστηκε το Σεπτέμβριο του 2008 και ενσωματώθηκε από την HTC στο μοντέλο HTC Dream, γνωστό και ως T-Mobile G1. Μερικές από τις λειτουργίες του ήταν το Android Market (δυνατότητα στο χρήστη να κατεβάζει εφαρμογές), πρόσβαση στο Gmail και συγχρονισμός με την Gmail εφαρμογή, YouTube video player, Google Maps, υποστήριξη Wi-Fi και Bluetooth, εμφάνιση ειδοποιήσεων στο Status bar και διάφορες άλλες. Ένα χρόνο αργότερα κυκλοφόρησε η πρώτη αναβαθμισμένη έκδοση (1.1) με API level 2 με κάποιες επιπλέον δυνατότητες όπως για παράδειγμα, εύρεση διαθέσιμων σχόλιων και λεπτομερειών για μία τοποθεσία όταν ο χρήστης κάνει χρήση των Google Maps. Από εκεί και έπειτα ακολούθησαν δεκάδες αναβαθμισμένες εκδόσεις τις οποίες θα δούμε συνοπτικά παρακάτω.

1.4.1 Cupcake 1.5

Η έκδοση αυτή, βασισμένη στον πυρήνα Linux 2.6.27, ανακοινώθηκε τον Απρίλιο του 2009 με API level 3 και εισήγαγε κάποιες νέες λειτουργίες, οι βασικότερες από τις οποίες είναι οι εξής:

- Δυνατότητα πρόβλεψης λέξεων κατά την πληκτρολόγηση (auto complete) και δημιουργία λεξικού με λέξεις που ορίζει ο χρήστης.
- Εισαγωγή των Widgets τα οποία είναι μικροεφαρμογές που μπορούν να τρέξουν στην αρχική οθόνη της συσκευής, π.χ. εμφάνιση του καιρού στην περιοχή μας.
- Υποστήριξη εγγραφής και αναπαραγωγής βίντεο της μορφής MPEG-4 και 3GP.
- Δυνατότητα αντιγραφής και επικόλλησης στον Web browser.
- Δυνατότητα upload (ανέβασμα) βίντεο κατευθείαν στο YouTube.
- Δυνατότητα upload εικόνων κατευθείαν στο Picasa.

Εικόνα 6: Λογότυπο Cupcake

1.4.2 Donut 1.6

Το Σεπτέμβριο του 2009 παρουσιάστηκε η έκδοση Donut 1.6 με API level 4, βασισμένη στον πυρήνα Linux 2.6.29, με κάποιες νέες προσθήκες, με τις σημαντικότερες να αναφέρονται παρακάτω:

- Βελτίωση της φωνητικής και γραπτής αναζήτησης, συμπεριλαμβανομένου το ιστορικό επισκέψεων του χρήστη (bookmark history) και τις επαφές.
- Δυνατότητα μετατροπής κειμένου σε ομιλία για πολλές γλώσσες.
- Ευκολότερη αναζήτηση και προβολή στιγμιότυπων εφαρμογών στο Android Market.
- Υποστήριξη για οθόνες ανάλυσης WVGA (768x480).
- Μαζική διαγραφή φωτογραφιών.

Εικόνα 7: Λογότυπο Donut

1.4.3 Eclair 2.0

Τον Οκτώβριο του 2009 κυκλοφορεί η νέα έκδοση του Android, βασισμένη στον πυρήνα Linux 2.6.29, η Eclair 2.0 με API level 5, με νέες προσθήκες όπως:

- Ο χρήστης έχει τη δυνατότητα να συνδεθεί σε διάφορες υπηρεσίες όπως το Email με πολλαπλούς λογαριασμούς και οι εφαρμογές αυτές συγχρονίζουν το περιεχόμενό τους.
- Υποστήριξη πρωτοκόλλου Bluetooth 2.1.
- Πατώντας πάνω σε μία επαφή ο χρήστης έχει τη δυνατότητα να επιλέξει αν θα στείλει μήνυμα, Email ή θα πραγματοποιήσει μία τηλεφωνική κλήση.
- Αναβάθμιση του Google Maps στην έκδοση 3.1.2.
- Νέες δυνατότητες της camera όπως εισαγωγή του flash, ψηφιακό zoom (μεγέθυνση), εξισορρόπηση φωτεινότητας και εισαγωγή εφέ χρώματος.
- Οι δύο επόμενες εκδόσεις ήταν οι Eclair 2.0.1 με API level 6 και η Eclair 2.1 με API level 7 με ελάχιστες διαφορές στο API που διέθεταν.

Εικόνα 8: Λογότυπο Eclair

1.4.4 Froyo 2.2-2.2.3

Τον Μάιο του 2010 κυκλοφορεί η Froyo 2.2 με API level 8, βασισμένη στον πυρήνα Linux 2.6.32, με σημαντικές προσθήκες όπως:

- Διαθέσιμα αυτόματα updates μέσω του Android Market.
- Δυνατότητα διαμερισμού του internet μέσω wi-fi (wireless fidelity: ασύρματη πιστότητα) σε άλλες συσκευές.
- Προστασία συσκευής με κωδικό.
- Δυνατότητα επαναφοράς εργοστασιακών ρυθμίσεων και διαγραφή όλων των προσωπικών δεδομένων.

Εικόνα 9: Λογότυπο Froyo

1.4.5 Gingerbread 2.3-2.3.2

Τον Δεκέμβριο του 2010 κυκλοφόρησε η νέα έκδοση του Android 2.3, βασισμένη στον πυρήνα Linux 2.6.35, με όνομα Gingerbread με API level 9. Εδώ έγιναν βελτιώσεις όπως:

- Υπέστη βελτίωση στο τρόπο διαχείριση της ενέργειας με τη δυνατότητα τερματισμού οποιασδήποτε εφαρμογής που τρέχει στο προσκήνιο.
- Εισήχθη η εφαρμογή λήψεις (downloads) μέσω της οποίας οι χρήστες έχουν μία λίστα με τα αρχεία που έχουν κατεβάσει στη συσκευή τους.
- Υποστήριξη για οθόνες ανάλυσης WXGA (1366x768) και πάνω.
- Υποστηρίζει το πρωτόκολλο SIP δίνοντας τη δυνατότητα τηλεφωνίας μέσω διαδικτύου.
- Ακολούθησαν ένα χρόνο μετά οι βελτιωμένες εκδόσεις, Android 2.3.3-2.3.7 Gingerbread, με API level 10 με μικρές διαφορές.

Εικόνα 10: Λογότυπο Gingerbread

1.4.6 Honeycomb 3.0

Το Φεβρουάριο του 2011 η Google ανακοίνωση την καινούργια της έκδοση 3.0, με όνομα Honeycomb και API level 11, βασισμένη στον πυρήνα Linux 2.6.36 και λίγο αργότερα τις εκδόσεις 3.1 και 3.2 με API level 12 και 13 αντίστοιχα με μικρές διαφορές, η οποία προοριζόταν κυρίως για tablets. Σημαντικά νέα χαρακτηριστικά είναι:

- Εκ νέου σχεδιασμός πληκτρολογίου, λόγω του ότι τα tablets διαθέτουν μεγαλύτερη οθόνη.
- Δυνατότητα προβολής των τρέχων εφαρμογών (multitasking) και μετάβαση σε μία από αυτές.
- Δυνατότητα σύνδεσης του tablet με άλλες συσκευές εισόδου όπως πληκτρολόγια μέσω Bluetooth ή USB σύνδεσης.
- Προσθήκη της Action Bar στο πάνω μέρος της οθόνης σε κάθε εφαρμογή, δίνοντας τη επιλογή στο χρήστη να έχει πρόσβαση σε περαιτέρω ρυθμίσεις.
- Προσθήκη της System Bar στο κάτω μέρος της οθόνης η οποία ενσωματώνει επαναλαμβανόμενες λειτουργίες.
- Υποστήριξη συσκευών με περισσότερους από έναν πυρήνες (multi-core processors).

Εικόνα 11: Λογότυπο Honeycomb

1.4.7 Ice Cream Sandwich 4.0-4.0.2

Τον Οκτώβριο του 2011 έκανε την εμφάνιση της η έκδοση 4.0 του Android βασισμένο στον πυρήνα Linux 3.0.1, με το όνομα Ice Cream Sandwich και API level 14. Σε αυτήν την έκδοση είχαμε βελτιώσεις στην ταχύτητα και στην απόδοση του συστήματος καθώς και σημαντικές προσθήκες όπως:

- Εγγραφή βίντεο σε 1080p.
- Εισήχθη το wi-fi Direct, ένα πρότυπο το οποίο επιτρέπει στις συσκευές να συνδεθούν εύκολα με άλλες χωρίς να απαιτείται σημείο ασύρματης πρόσβασης.
- Προσθήκη αναγνώρισης προσώπου για να ξεκλειδώσει η συσκευή (face unlock).
- Δυνατότητα τερματισμού μίας εφαρμογής από τη λίστα πρόσφατων εφαρμογών μέσω ολίσθησης.
- Βελτιωμένος μηχανισμός εύρεσης λαθών στο πληκτρολόγιο.

Ακολούθησαν ενημερωμένες εκδόσεις της Ice Cream Sandwich με API level 15, οι 4.0.3 και 4.0.4, οι οποίες εισήγαγαν καινούργια API για τους προγραμματιστές και μεγάλες βελτιώσεις στην κάμερα.

Εικόνα 12: Λογότυπο Ice Cream Sandwich

1.4.8 Jelly Bean 4.1

Τον Ιούλιο του 2012 κυκλοφορεί η έκδοση Jelly Bean 4.1 με API level 16, βασισμένη στον πυρήνα Linux 3.0.31 και χρησιμοποιήθηκε πρώτα από το Nexus 7 tablet. Μερικά χαρακτηριστικά είναι:

- Δυνατότητα απενεργοποίησης των ειδοποιήσεων στις εφαρμογές.

- Τα Widgets και οι συντομεύσεις εφαρμογών μπορούν αυτόματα να προσαρμόσουν το μέγεθος εμφάνισής τους στην αρχική οθόνη της συσκευής (home screen).
- Πολλαπλούς λογαριασμούς χρηστών σε tablet, συναντάμε στην επόμενη έκδοση της Jelly Bean 4.2, βασισμένη στον πυρήνα Linux 3.4.0, καθώς και βελτιώσεις προσβασιμότητας όπως zoom με δύο δάχτυλα και triple-tap για μεγέθυνση ολόκληρης της οθόνης.
- Υποστήριξη OpenGL ES 3.0 για βελτιωμένη εμπειρία παιχνιδιού, αυτόματη συμπλήρωση (auto complete) ενός τηλεφωνικού αριθμού στην εφαρμογή Phone, υποστήριξη ανάλυση 4K (4096x2160), ιστορικό wi-fi ακόμα και όταν αυτό είναι απενεργοποιημένο αλλά και βελτιώσεις στην ασφάλεια συναντάμε στην τελευταία έκδοση της Jelly Bean, την 4.3 με API level 18.

Εικόνα 13: Λογότυπο Jelly Bean

1.4.9 KitKat 4.4

Τον Οκτώβριο του 2013 η Google ανακοινώνει την έκδοση KitKat 4.4 με API level 19 και ενσωματώνεται για πρώτη φορά στο Google Nexus 5. Ακολούθησαν οι αναβαθμίσεις τις KitKat 4.4.1, 4.4.2, 4.4.3 καταλήγοντας στην 4.4.4 με ελάχιστες τροποποιήσεις. Η έκδοση αυτή έχει τροποποιηθεί κατάλληλα ώστε να είναι συμβατή σε μεγαλύτερο εύρος συσκευών από ότι οι προηγούμενες εκδόσεις αλλά η ελάχιστη απαιτούμενη RAM είναι 512 MB. Οι σημαντικότερες προσθήκες είναι:

- Βελτιωμένα γραφικά κάνουν την εμπειρία του χρήστη καλύτερη από ποτέ.
- Απώλεια notification bar και control bar.
- Το Hangouts app αντικαθιστά τα SMS και MMS.
- Εισαγωγή του Android Runtime (ART), το οποίο είναι ένα νέο περιβάλλον χρόνου εκτέλεσης με σκοπό να αντικαταστήσει το Dalvik VM. Το ART δεν είναι ενεργοποιημένο ως προεπιλογή αλλά καθορίζεται από το χρήστη.

Εικόνα 14: Λογότυπο KitKat

1.4.10 Lollipop 5.0

Τον Νοέμβριο του 2014 κυκλοφορεί η Lollipop 5.0 με API level 21 η οποία είναι η τελευταία γνωστή μέχρι σήμερα έκδοση του Android. Οι κυριότερες αλλαγές είναι στο γραφικό περιβάλλον (user interface), το οποίο έχει σχεδιαστεί από την αρχή καθώς και η επίσημη αντικατάσταση του Dalvik VM από το ART. Μερικές ακόμα προσθήκες είναι:

- Υποστήριξη επεξεργαστών 64-bit.
- Αναβάθμιση του OpenGL ES σε 3.1.
- Βελτιώσεις στη χρήση και το όριο ζωής της μπαταρίας, γνωστό και ως Project Volta.
- Πρόσφατες χρησιμοποιούμενες εφαρμογές, μένουν στη μνήμη ακόμα και μετά την επανεκκίνηση της συσκευής.

Εικόνα 15: Λογότυπο Lollipop

Για περισσότερες πληροφορίες για όλες τις εκδόσεις του Android επισκεφτείτε το παρακάτω σύνδεσμο: http://en.wikipedia.org/wiki/Android_version_history

1.5 Δομή εργασίας

Στο παρόν κεφάλαιο αναφερθήκαμε θεωρητικά πάνω στο λειτουργικό σύστημα του Android, το οποίο αποτελεί το 1ο κεφάλαιο με τίτλο «Εισαγωγή». Στο 2ο κεφάλαιο «Χρήσιμα Εργαλεία» παρουσιάζονται όλα τα εργαλεία-προγράμματα που χρησιμοποιήθηκαν στην παρούσα πτυχιακή. Στο 3ο κεφάλαιο «Μεθοδολογία Υλοποίησης» περιγράφονται όλες οι λειτουργίες της εφαρμογής καθώς και η σχεδίαση της βάσης δεδομένων. Στο 4ο κεφάλαιο με τίτλο «Προγραμματισμός στο Android», παρουσιάζεται αναλυτικά ο κώδικας που περιλαμβάνει η εφαρμογή μας, γραμμένος σε γλώσσα

προγραμματισμού Java και στη γλώσσα σήμανσης XML, καθώς και τα αρχεία PHP που συγγράψαμε για την επικοινωνία και αλληλεπίδραση με τη βάσης δεδομένων. Τέλος στο 5ο και τελευταίο κεφάλαιο με τίτλο «Βιβλιογραφία», αναφέρονται όλες οι πηγές που βοήθησαν στην εκπόνηση αυτής της εργασίας.

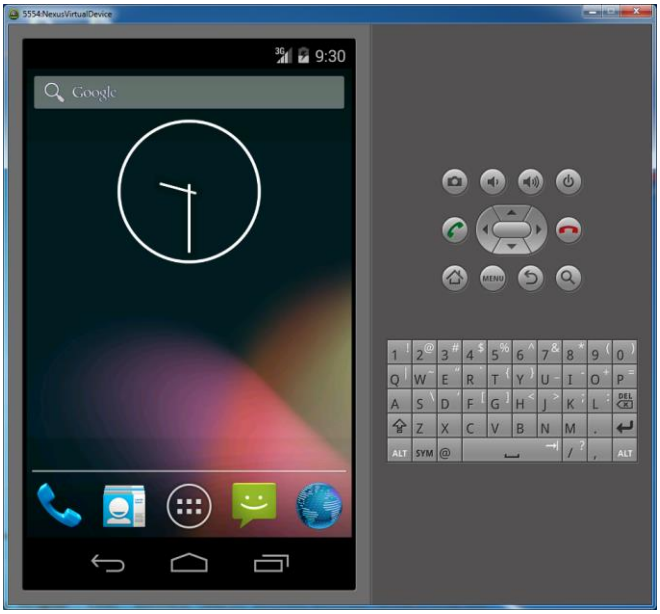
2 Χρήσιμα εργαλεία

Για να δημιουργήσουμε μία εφαρμογή Android απαιτείται η χρήση της αντικειμενοστραφής γλώσσας προγραμματισμού Java. Το λειτουργικό σύστημα Android είναι μία πλατφόρμα ανοικτού κώδικα (open source), προσφέροντας τη δυνατότητα στους προγραμματιστές να συνθέσουν και να υλοποιήσουν δικές τους εφαρμογές δωρεάν. Για την δημιουργία μίας εφαρμογής είναι αναγκαία η εγκατάσταση βασικών προγραμμάτων όπως θα δούμε στη συνέχεια, τα οποία διατίθενται ελεύθερα και δωρεάν στο διαδίκτυο.

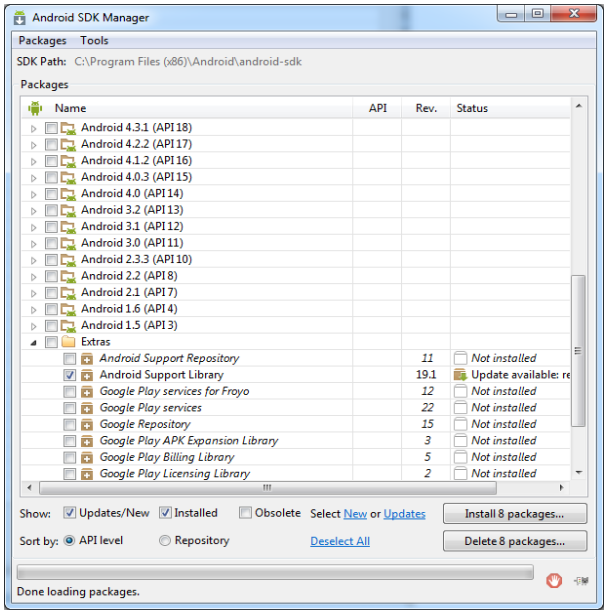
2.1 Εργαλεία ανάπτυξης κώδικα Android

Το Java Development Kit (JDK) είναι μία συλλογή από εργαλεία που επιτρέπουν στους προγραμματιστές να γράφουν και να δοκιμάζουν εφαρμογές γραμμένες σε Java. Η πλατφόρμα JDK διατίθεται δωρεάν από την εταιρεία Oracle στον παρακάτω σύνδεσμο: http://www.oracle.com/technetwork/articles/javase/index-jsp-138363.html

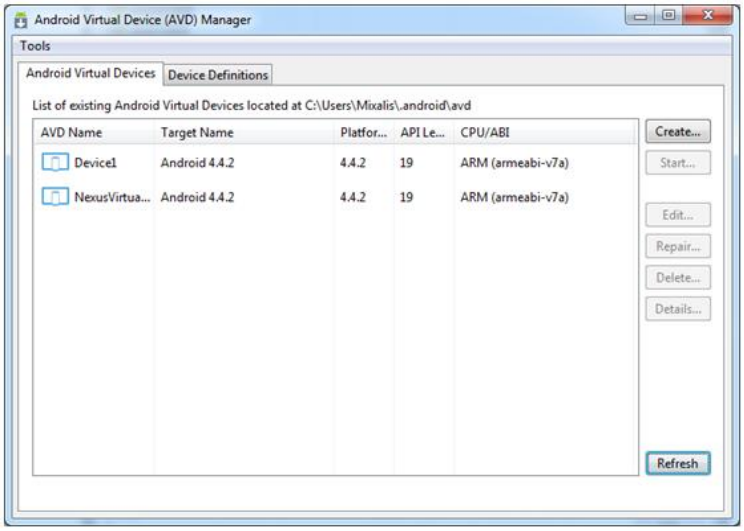
Το Android SDK (Android Software Development Kit) είναι ένα πακέτο το οποίο περιλαμβάνει τα απαραίτητα εργαλεία για την ανάπτυξη, εκσφαλμάτωση και υλοποίηση μίας Android εφαρμογής. Το SDK περιέχει όλες τις βιβλιοθήκες και τα API που είναι διαθέσιμα στην εκάστοτε έκδοση του λειτουργικού καθώς και έναν εξομοιωτή Android, στον οποίο μπορούν οι προγραμματιστές να δοκιμάσουν και να διορθώσουν τις εφαρμογές τους. Αξίζει να αναφέρουμε ότι ο εξομοιωτής συμπεριφέρεται περίπου όπως ένα κινητό τηλέφωνο και χάρη στο AVD Manager ο κάθε προγραμματιστής μπορεί να δημιουργήσει και να προσαρμόσει τον δικό του εξομοιωτή συσκευής ανάλογα με τις ανάγκες της εφαρμογής του. Επιπλέον το Android SDK παρέχει κάποια δείγματα (samples) και δοκιμαστικές εφαρμογές οι οποίες αποσκοπούν στην εξοικείωση των προγραμματιστών με την ανάπτυξη εφαρμογών Android.



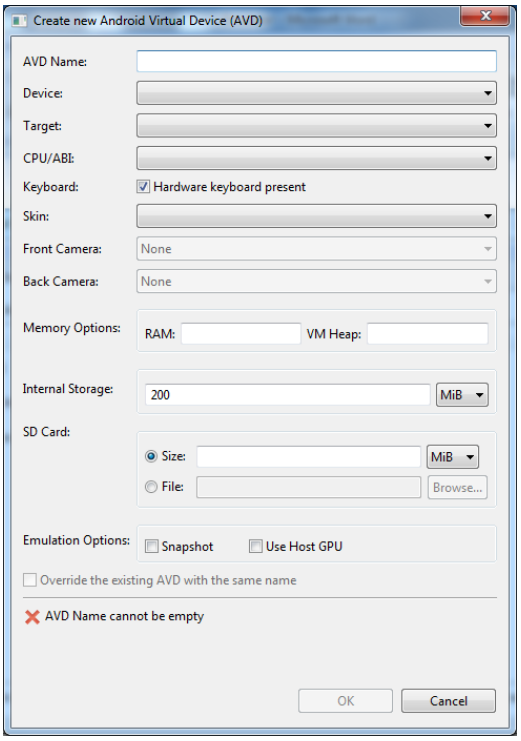
Εικόνα 16: Αρχική οθόνη Emulator



Εικόνα 17: Παράθυρο SDK Manager



Εικόνα 18: Παράθυρο AVD Manager



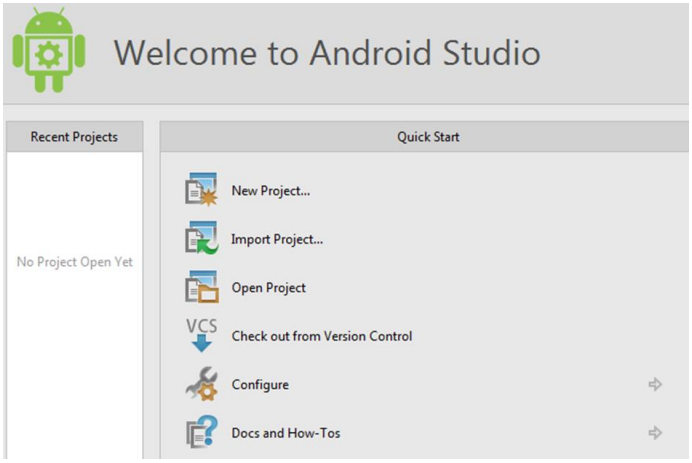
Εικόνα 19: Παράθυρο δημιουργίας εικονικής συσκευής

Το Android Studio IDE (Integrated Development Environment) είναι μία ολοκληρωμένη πλατφόρμα στην οποία μπορούν οι προγραμματιστές να συνθέτουν και να εκτελούν εφαρμογές για το λειτουργικό σύστημα Android. Το Android Studio είναι το επίσημο περιβάλλον ανάπτυξης εφαρμογών Android και διατίθεται δωρεάν στο παρακάτω link: http://developer.android.com/sdk/index.html. Το πακέτο που είναι διαθέσιμο προς λήψη περιέχει το προγραμματιστικό περιβάλλον Android Studio και το πακέτο ανάπτυξης εφαρμογών Android SDK.

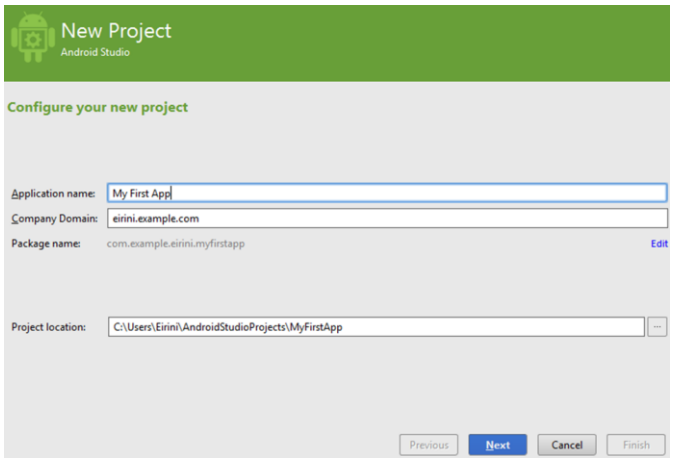
2.1.1 Εσωτερικά στο Android Studio

Σε αυτό το σημείο θα αναφερθούμε περιληπτικά στη διαδικασία που πρέπει να ακολουθηθεί για τη δημιουργία ενός καινούργιου Project στο Android Studio και θα κοιτάξουμε την ανατομία στους σημαντικότερους φακέλους του.

Εκτελώντας το πρόγραμμα για πρώτη φορά εμφανίζεται ένα παράθυρο στο οποίο πρέπει να επιλέξουμε τι θέλουμε να κάνουμε. Στη δικιά μας περίπτωση επιλέγουμε New Project. Στο αμέσως επόμενο παράθυρο ορίζουμε την ονομασία που θα έχει η εφαρμογή μας, το όνομα του πακέτου και την τοποθεσία που θα αποθηκευτεί η εφαρμογή στο σκληρό δίσκο.

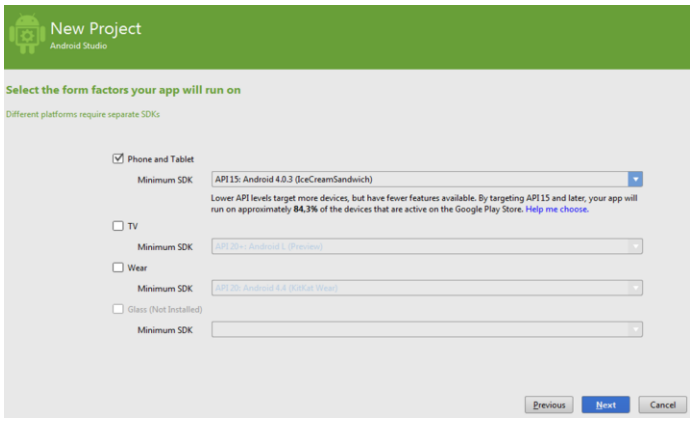


Εικόνα 20: Αρχική οθόνη Android Studio

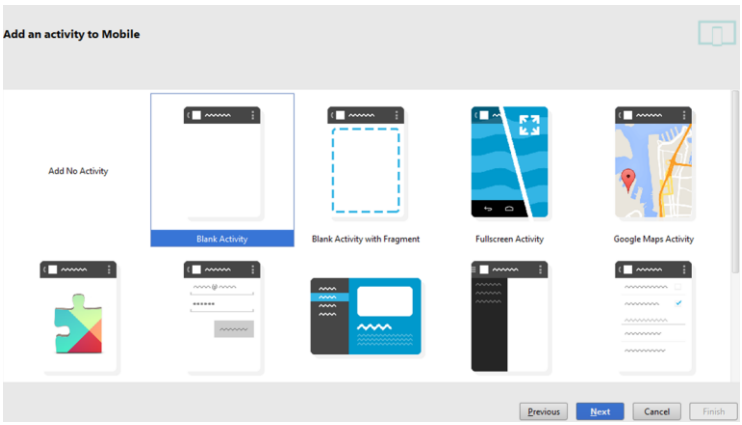


Εικόνα 21: Δήλωση ονόματος εφαρμογής

Στη συνέχεια καθορίζουμε τη μικρότερη έκδοση του SDK, στην οποία θα μπορεί να τρέχει η εφαρμογή. Έπειτα μας δίνεται η δυνατότητα να επιλέξουμε τον τύπο του πρωταρχικού Activity (βλ. κεφάλαιο 4) από το οποίο θα εκκινείτε η εφαρμογή μας.



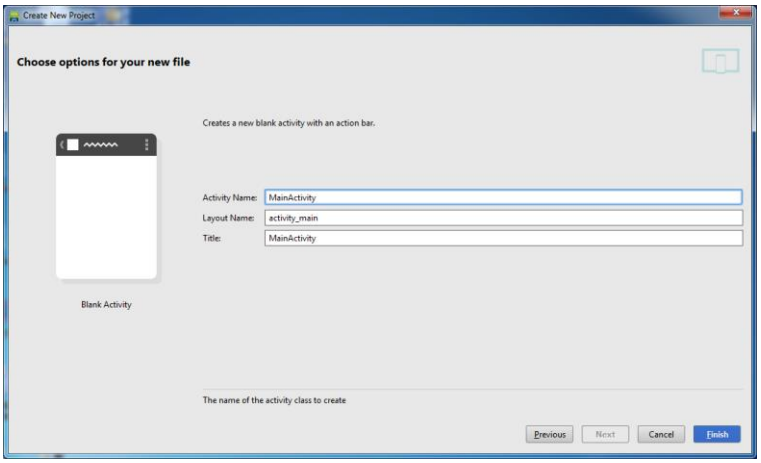
Εικόνα 22: Ορισμός έκδοσης SDK



Εικόνα 23: Παράθυρο επιλογής Activity

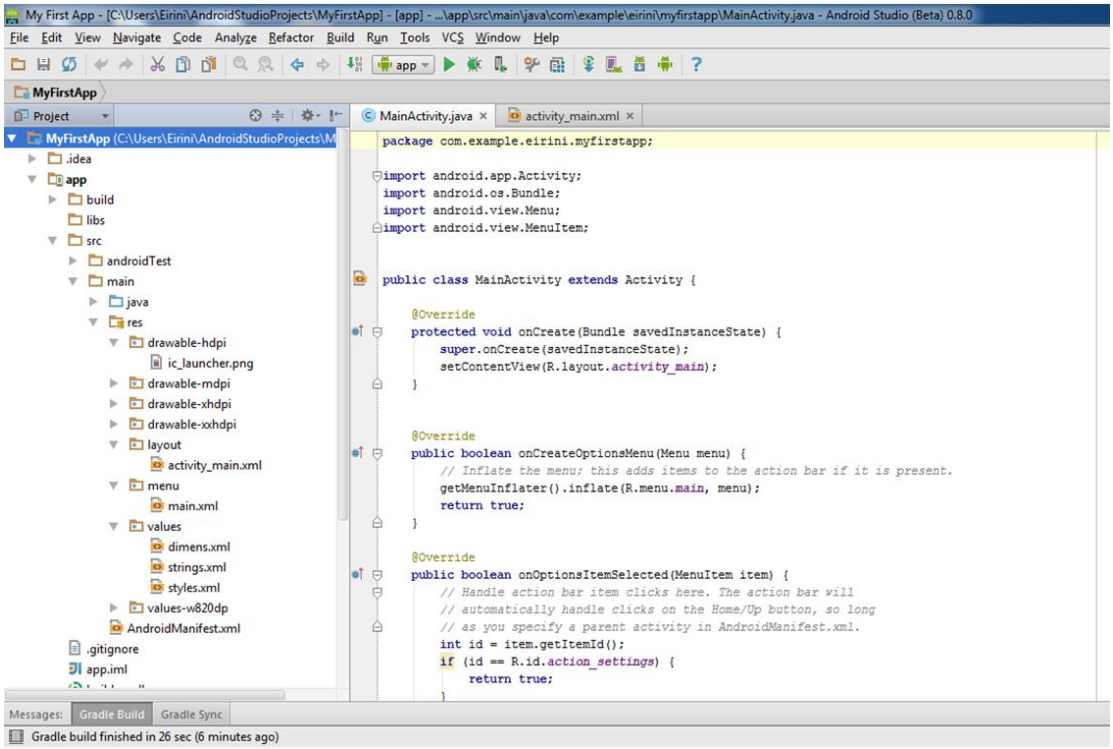
Τέλος, ορίζουμε την ονομασία του Activity, το όνομα του layout[1] που αντιπροσωπεύει το Activity αυτό, τον τίτλο που θα αναγράφεται στο Action Bar της εφαρμογής μας και πατάμε το κουμπί Finish.

[1] Διάταξη οθόνης



Εικόνα 24: Δήλωση ονόματος Activity

Αντικρίζοντας το περιβάλλον εργασίας του Android Studio, στα αριστερά βρίσκονται οι φάκελοι του project δομημένοι με ιεραρχική μορφή.



Εικόνα 25: Περιβάλλον εργασίας Android Studio

Μέσα στο φάκελο app συναντάμε τον υποφάκελο src ο οποίος περιέχει σημαντικούς καταλόγους για την υλοποίηση μίας εφαρμογής. Στο φάκελο main βρίσκονται τα κύρια συστατικά μιας εφαρμογής τα οποία είναι οι κατάλογοι java, res και το αρχείο AndroidManifest.xml.

Στον κατάλογο java βρίσκεται το αρχικό Activity, όπως το ονομάσαμε στο στάδιο δημιουργίας ενός καινούργιου project, που στην περίπτωσή μας είναι το MainActivity.java. Επίσης σε αυτόν το φάκελο θα αποθηκευτούν όλα τα Activity που θα δημιουργηθούν για τις ανάγκες της εφαρμογής.

Στον κατάλογο res βρίσκονται όλοι οι πόροι (δεδομένα) που δομούν την εφαρμογή. Υπάρχουν πολλοί τύποι πόρων και για αυτόν τον λόγο αποθηκεύονται σε διαφορετικούς υποφακέλους. Στον υποφάκελο drawable αποθηκεύονται όλα τα δεδομένα που σχετίζονται με γραφικά στοιχεία, όπως εικόνες. Ανάλογα με την ανάλυση της οθόνης έχουμε διαφορετικούς υποφακέλους για κάθε ανάλυση

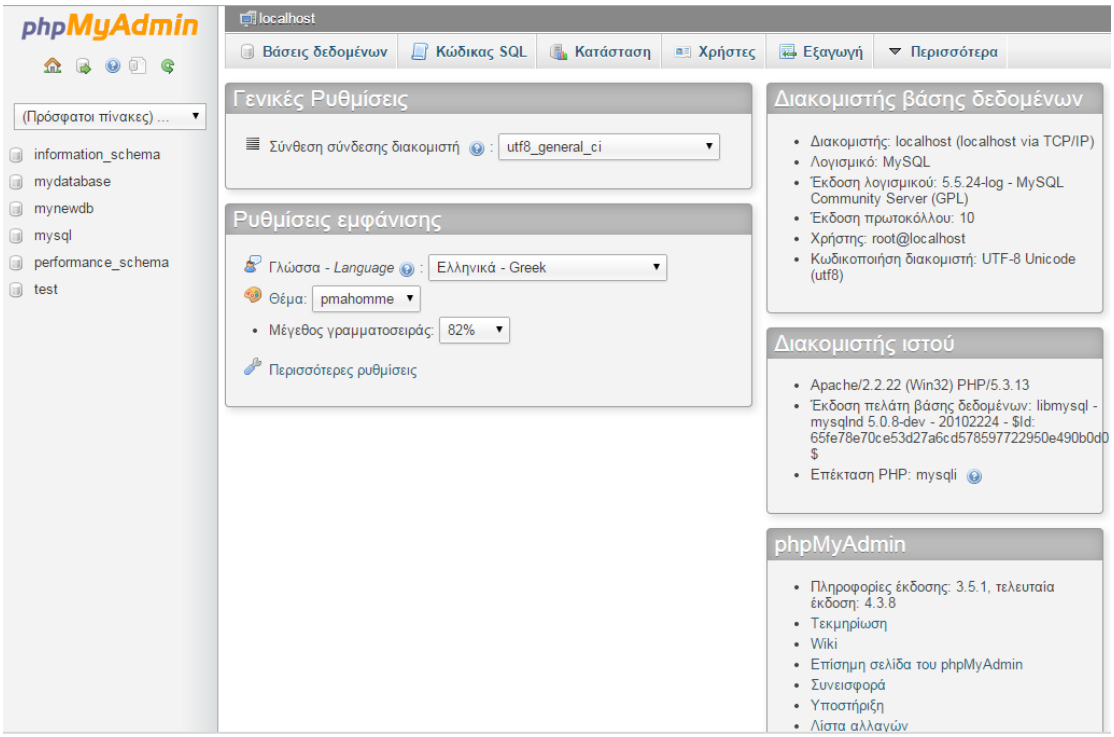
έτσι προκύπτουν οι: drawable-hdpi για εικόνες 72x72 pixels, drawable-mdpi για εικόνες 48x48, drawable-xhdpi για εικόνες 96x96 και drawable-xxhdpi για εικόνες 144x144. Στον υποφάκελο layout βρίσκονται οι πόροι διεπαφής χρήστη. Το κάθε Activity συνοδεύεται από ένα xml αρχείο το οποίο αναπαριστά στην οθόνη τα γραφικά στοιχεία του Activity. Σε αυτό το xml μπορούμε να προσθέσουμε διάφορα αντικείμενα όπως κουμπιά, εικόνες κ.α., είτε γράφοντας κώδικα σε μορφή xml, είτε κάνοντας drag and drop από τη διαθέσιμη παλέτα που υπάρχει στην καρτέλα Design. Στον υποφάκελο menu βρίσκονται τα αρχεία xml τα οποία περιγράφουν τα στοιχεία που έχουμε στο menu της Action Bar. Το κάθε στοιχείο στο αρχείο menu.xml αποτελεί ένα item. Στον υποφάκελο values συναντάμε τα αρχεία: dimens.xml στο οποίο μπορούμε να έχουμε αποθηκευμένες διαστάσεις, strings.xml το οποίο έχει αποθηκευμένες όλες τις συμβολοσειρές της εφαρμογής και styles.xml, το οποίο περιέχει διάφορες μορφοποιήσεις (στυλ, χρώματα, διαστάσεις) για αντικείμενα όπως ένα κουμπί. Τέλος βρίσκεται το αρχείο AndroidManifest.xml το οποίο περιλαμβάνεται σε κάθε εφαρμογή Android. Το αρχείο αυτό περιέχει σημαντικά στοιχεία που αφορούν την "προσωπικότητα" της εφαρμογής όπως ο τίτλος, το εικονίδιο που θα είναι ορατό στους χρήστες, πληροφορίες έκδοσης (SDK) καθώς και τα δικαιώματα που θα έχει η εφαρμογή όπως για παράδειγμα, πρόσβαση στο internet.

2.2 Εργαλεία ανάπτυξης βάσης δεδομένων

Για να δημιουργήσουμε τη βάση δεδομένων της εφαρμογής μας χρησιμοποιήσαμε το πρόγραμμα WAMP. Το πρόγραμμα αυτό είναι ένα πακέτο, που περιλαμβάνει χρήσιμα προγράμματα για την ανάπτυξη μίας βάσης δεδομένων. Αυτά τα προγράμματα είναι τα: Apache, MySQL και PHP. Εν συντομία, ο Apache είναι ένας εξυπηρετητής ιστού (web), η MySQL είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων και η PHP γλώσσα προγραμματισμού για δημιουργία σελίδων web. Το WAMP πήρε το όνομα του από τα αρχικά των παραπάνω προγραμμάτων, συμπεριλαμβανομένου και των Windows, το λειτουργικό που τρέχει το πρόγραμμα. Το WAMP εγκαθίσταται εύκολα και γρήγορα και διατίθεται δωρεάν στο σύνδεσμο: http://www.wampserver.com/en/.

Εικόνα 26: Λογότυπο WAMP

Μετά την επιτυχή εγκατάσταση του WAMP έχουμε πρόσβαση στην phpMyAdmin (εργαλείο λογισμικού γραμμένο σε PHP), η οποία διαχειρίζεται τη MySQL μέσω του διαδικτύου (web browser). Η ευκολία που παρέχει η phpMyAdmin είναι η δημιουργία βάσεων δεδομένων σε γραφικό περιβάλλον χωρίς όμως να χάνεται η δυνατότητα να εκτελέσουμε κώδικα SQL.



Εικόνα 27: Περιβάλλον εργασίας phpMyAdmin

3 Μεθοδολογία υλοποίησης

3.1 Γενική εικόνα

Η παρούσα εργασία έχει σαν στόχο την υλοποίηση μίας εφαρμογής στο λειτουργικό σύστημα Android, η οποία όχι μόνο παρέχει πληροφορίες αλλά δίνει και μία μεγάλη ποικιλία επιλογών στους χρήστες για ένα τουριστικό μέρος. Δημιουργήσαμε ένα σύστημα για μία πόλη με στοιχεία όπως: ξενοδοχεία, γραφεία ενοικιάσεων οχημάτων, εκδρομικά γραφεία, δρομολόγια αεροπορικών/ακτοπλοϊκών γραμμών και μουσεία/αξιοθέατα.

Στο χρήστη θέλουμε να δίνεται η δυνατότητα πρόσβασης στις καρτέλες Hotels, Rentals, Trips, Flights & Ferries και Attractions που αντιστοιχούν στα παραπάνω στοιχεία, παρέχοντάς του τη δυνατότητα για online κρατήσεις μέσω μίας βάσης δεδομένων που θα κατασκευάσουμε κατάλληλα. Οι καρτέλες αυτές θα τοποθετηθούν σε μία μπάρα ενεργειών (Action Bar). Για τη διευκόλυνση του χρήστη θα τοποθετούν στη μπάρα αυτή, εκτός από το εικονίδιο της εφαρμογής και τον τίτλος της, ένα καλάθι αγορών και μια ενέργεια αναζήτησης. Το καλάθι αγορών θα δίνει δικαίωμα στο χρήστη για ακυρώσεις κρατήσεων και η ενέργεια της αναζήτησης, δικαίωμα για μεμονωμένη ή μαζική αναζήτηση.

3.2 Λειτουργίες εφαρμογής

Συγκεκριμένα, η πρώτη καρτέλα θα περιέχει μία λίστα με τα ξενοδοχεία, ταξινομημένη βάση την κατηγορία τους. Πατώντας σε ένα από αυτά, ο χρήστης θα μπορεί να αντλήσει απλές πληροφορίες, όπως διεύθυνση και τηλέφωνο. Σε δεύτερο όμως στάδιο, θα του δίνεται η δυνατότητα να επιλέξει την ημερομηνία που επιθυμεί να βρεθεί στο ξενοδοχείο και να κάνει κράτηση σε ένα δωμάτιο. Στο επόμενο παράθυρο που θα ανοίγει, θα φαίνονται τα διαθέσιμα δωμάτια και οι αντίστοιχες τιμές τους. Όταν ο χρήστης επιλέξει ένα από αυτά τα δωμάτια, θα ανοίγει ένα παράθυρο διαλόγου στο οποίο θα μπορεί να επιβεβαιώσει την κράτησή του. Αυτόματα θα πρέπει να ενημερώνεται η βάση δεδομένων και το καλάθι αγορών. Στην οθόνη θα εμφανίζονται τα διαθέσιμα δωμάτια με την αντίστοιχη χρέωση ανά ημέρα. Εάν ο χρήστης επιλέξει να κάνει κράτηση παραπάνω από μία ημέρα, τότε στο καλάθι αγορών η χρέωση θα προκύπτει από των αριθμό των ημερών επί την τιμή ανά ημέρα.

Στην επόμενη καρτέλα ο χρήστης θα συναντά τα γραφεία ενοικιάσεων σε μορφή λίστας με τα ονόματα τους, τα τηλέφωνα και την περιοχή τους. Θα ακολουθήσουμε την ίδια διαδικασία όπως και προηγουμένως, για την επιλογή της ημερομηνίας, καταλήγοντας σε μία λίστα με τα διαθέσιμα οχήματα. Για την καλύτερη εξυπηρέτηση του χρήστη, θα τοποθετηθούν εικονίδια που αντιπροσωπεύουν τον τύπο του οχήματος (αυτοκίνητο ή μηχανή). Επιλέγοντας κάποιο από αυτά θα εμφανίζεται μία καινούργια οθόνη η οποία θα περιλαμβάνει μία φωτογραφία του επιλεγμένου οχήματος, καθώς και πληροφορίες όπως κυβικά, θέσεις επιβατών, τύπος οχήματος και φυσικά η τιμή ανά ημέρα. Σε αυτό το σημείο θα ενσωματώσουμε ένα είδος προσφοράς. Δηλαδή, αυξάνοντας τις ημέρες κρατήσεως ενός οχήματος θα γίνεται μία κλιμακωτή μείωση στην τιμή ανά ημέρα. Τέλος, μέσω ενός κουμπιού, η εφαρμογή θα τοποθετεί αυτό το προϊόν στο καλάθι αγορών και αυτόματα θα ενημερώνεται η βάση δεδομένων.

Στην επόμενη καρτέλα θα τοποθετηθούν τα εκδρομικά γραφεία της περιοχής. Αυτά θα εμφανίζονται σε μία λίστα που θα περιλαμβάνει τα ονόματα, τα τηλέφωνα και την περιοχή του κάθε γραφείου. Επιλέγοντας ένα από αυτά, ο χρήστης θα μπορεί να δει μία λίστα με τις εκδρομές (όνομα προορισμού) που διοργανώνει το κάθε γραφείο. Διαλέγοντας μία εκδρομή θα εμφανίζεται ένα νέο παράθυρο που θα περιέχει μία φωτογραφία του προορισμού και τις απαραίτητες πληροφορίες όπως την ώρα και το σημείο έναρξης της εκδρομής, την τιμή ανά άτομο, τη δυνατότητα επιλογής της ημερομηνίας που ο χρήστης θέλει να κάνει κράτηση, καθώς και τον αριθμό των ατόμων. Σε αυτό το σημείο θα ενσωματώσουμε το πακέτο προσφορών για πολλά άτομα, δηλαδή όσο αυξάνεται το σύνολο των ατόμων, τόσο μειώνεται η τιμή του κάθε ατόμου. Τέλος με το πάτημα ενός κουμπιού που θα προσθέσουμε στο τέλος αυτού του παραθύρου, η εφαρμογή θα τοποθετεί την εκδρομή στο καλάθι

αγορών. Η τιμή που θα αναγράφεται στο καλάθι αγορών, θα προκύπτει από το σύνολο των ατόμων επί την τελική τιμή του κάθε ατόμου.

Η καρτέλα "Flights & Ferries" θα εμφανίζει μία λίστα με τα ονόματα των αεροπορικών και ακτοπλοϊκών εταιριών. Διαλέγοντας μία από τις αεροπορικές εταιρίες θα εμφανίζεται ένα παράθυρο στο οποίο ο χρήστης θα μπορεί να επιλέξει αεροδρόμιο αναχώρησης καθώς και αεροδρόμιο προορισμού. Επιπλέον θα δίνεται η δυνατότητα επιλογής της ημερομηνίας αναχώρησης και της ημερομηνίας επιστροφής. Παρόλα αυτά ο χρήστης ίσως να μην επιλέξει να κλείσει εισιτήριο επιστροφής, έτσι το εισιτήριο θα είναι one way. Επίσης θα υλοποιήσουμε με κατάλληλο τρόπο την εφαρμογή ώστε να δέχεται πτήσεις μόνο από και προς Χανιά (το μέρος που επιλέξαμε). Στην συνέχεια πατώντας το κουμπί "Check availability and price", θα εμφανίζεται ένα παράθυρο που θα επιβεβαιώνει τα στοιχεία που επιλέχθηκαν και την αντίστοιχη τιμή του εισιτηρίου. Τέλος, πατώντας ένα κουμπί, θα ακολουθεί η ίδια διαδικασία όπως και στις προηγούμενες καρτέλες. Αντίθετα στις ακτοπλοϊκές γραμμές, ο χρήστης δεν θα έχει τη δυνατότητα επιλογής εισιτηρίου επιστροφής.

Τα "Attractions" θα είναι η τελευταία καρτέλα της εφαρμογής μας, στην οποία ο χρήστης θα συναντά μια λίστα που θα περιλαμβάνει διάφορα μουσεία και αξιοθέατα της περιοχής. Οποιαδήποτε και αν είναι η επιλογή του χρήστη, η εφαρμογή θα τον μεταφέρει σε ένα παράθυρο, το οποίο θα περιλαμβάνει μία εικόνα (με δυνατότητα εναλλαγής) και ένα κείμενο που θα αποτελεί την περιγραφή για το μουσείο/αξιοθέατο που επιλέχθηκε.

Όπως προαναφέραμε στο Action Bar θα ενσωματωθούν δύο λειτουργίες, το καλάθι αγορών και η αναζήτηση. Πατώντας τον μεγεθυντικό φακό (αναζήτηση), θα εμφανίζεται ένα νέο παράθυρο το οποίο θα έχει μία πληθώρα επιλογών. Αρχικά ο χρήστης θα μπορεί να κάνει αναζήτηση για διαθέσιμα δωμάτια, βάση της κατηγορίας και της περιοχής ενός ξενοδοχείου, να κάνει αναζήτηση για διαθέσιμο όχημα, βάση του τύπου και των αριθμό των θέσεων επιβατών, να κάνει αναζήτηση μίας εκδρομής ή να κάνει αναζήτηση όλων των παραπάνω μαζί. Αξίζει να αναφέρουμε ότι το Drop down menu που αφορά την περιοχή του ξενοδοχείου και το Drop down menu που αφορά τις θέσεις ενός οχήματος θα προσαρμόζεται ανάλογα με την κατηγορία του ξενοδοχείου και τον τύπο του οχήματος αντίστοιχα. Όταν ο χρήστης θα συμπληρώσει τα απαιτούμενα στοιχεία, θα εμφανίζεται ένα κουμπί με όνομα "Search". Τα αποτελέσματα της αναζήτησης θα εξαρτώνται άμεσα από τις ημερομηνίες (Check-in Date, Check-out Date), που θα ορίσει ο χρήστης πριν πατήσει το κουμπί.

Η αναζήτηση θα βγάζει τα αποτελέσματα που ικανοποιούν τις απαιτήσεις του χρήστη, σε ένα νέο παράθυρο. Το παράθυρο αυτό θα περιέχει μία λίστα με τα αποτελέσματα, καθένα από τα οποία θα συνοδεύεται με ένα εικονίδιο (προσθήκη στο καλάθι). Η αντίστοιχη τιμή που θα αναγράφεται σε καθένα στοιχείο της λίστας θα είναι άμεσα εξαρτώμενη από το σύνολο των ημερών που ο χρήστης θα θέλει να κάνει κράτηση. Πατώντας το αντίστοιχο εικονίδιο σε κάθε γραμμή της λίστας θα εμφανίζεται ένα παράθυρο διαλόγου και πατώντας την επιλογή "Add" θα γίνεται προσθήκη της κράτησης στο καλάθι.

3.3 Βάση δεδομένων

Για την υλοποίηση της εφαρμογής ήταν άμεση ανάγκη η δημιουργία μίας βάσης δεδομένων, η οποία θα έχει αποθηκευμένες όλες τις απαραίτητες πληροφορίες. Με τη χρήση του προγράμματος WAMP, δημιουργήσαμε έναν τοπικό server στον οποίο έχει πρόσβαση η εφαρμογή μας μέσω του internet. Χάρη στις τεχνολογίες Apache, MySQL και PHP οργανώσαμε έναν κύκλο επικοινωνίας στον οποίο η εφαρμογή στέλνει αιτήματα (request) και ο server απαντά στέλνοντας τα απαιτούμενα δεδομένα (response).

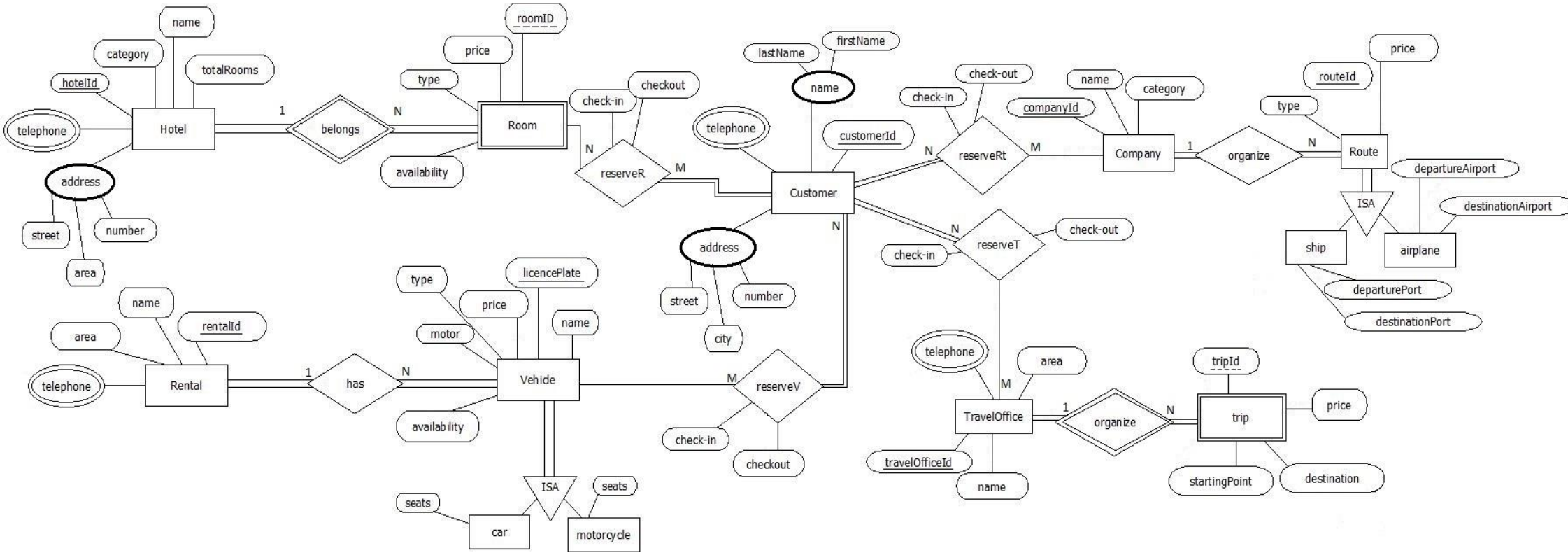
Βάση του πρωτοκόλλου Hypertext Transfer Protocol (HTTP), το οποίο είναι σχεδιασμένο για επικοινωνία μεταξύ Client και Server, έχουμε στη διάθεση μας τις μεθόδους GET και POST. Με τη μέθοδο GET μπορούμε να ανακτήσουμε πληροφορίες από μία βάση δεδομένων ενώ με τη μέθοδο POST μπορούμε να στείλουμε δεδομένα προς εγγραφή στη βάση μας. Σε συνδυασμό με τις τεχνολογίες PHP και MySQL μπορούμε να κάνουμε συγκεκριμένα ερωτήματα στη βάση καθώς και ενέργειες όπως δημιουργία, ενημέρωση, διαγραφή και ανάγνωση ενός πίνακα. Όλη αυτή η διαδικασία μεταξύ Client και Server που βασίζεται στο διαδίκτυο ονομάζεται Web Service.

Για να δημιουργήσουμε μία λειτουργική βάση δεδομένων για την εφαρμογή μας, έπρεπε πρώτα να μελετήσουμε τα βήματα για τη δημιουργία μίας βάσης δεδομένων από τη θεωρητική της

μεριά. Ξεκινήσαμε με το σχεδιασμό ενός μοντέλου δεδομένων βασισμένο σε αντικείμενα, συγκεκριμένα το Μοντέλο Οντοτήτων – Συσχετίσεων (Entity – Relationship Model). Το μοντέλο αυτό είναι μία διαγραμματική αναπαράσταση της δομής της βάσης. Για να το σχεδιάσουμε έπρεπε να συλλέξουμε όλα τα απαραίτητα δεδομένα και να τα οργανώσουμε κατάλληλα.

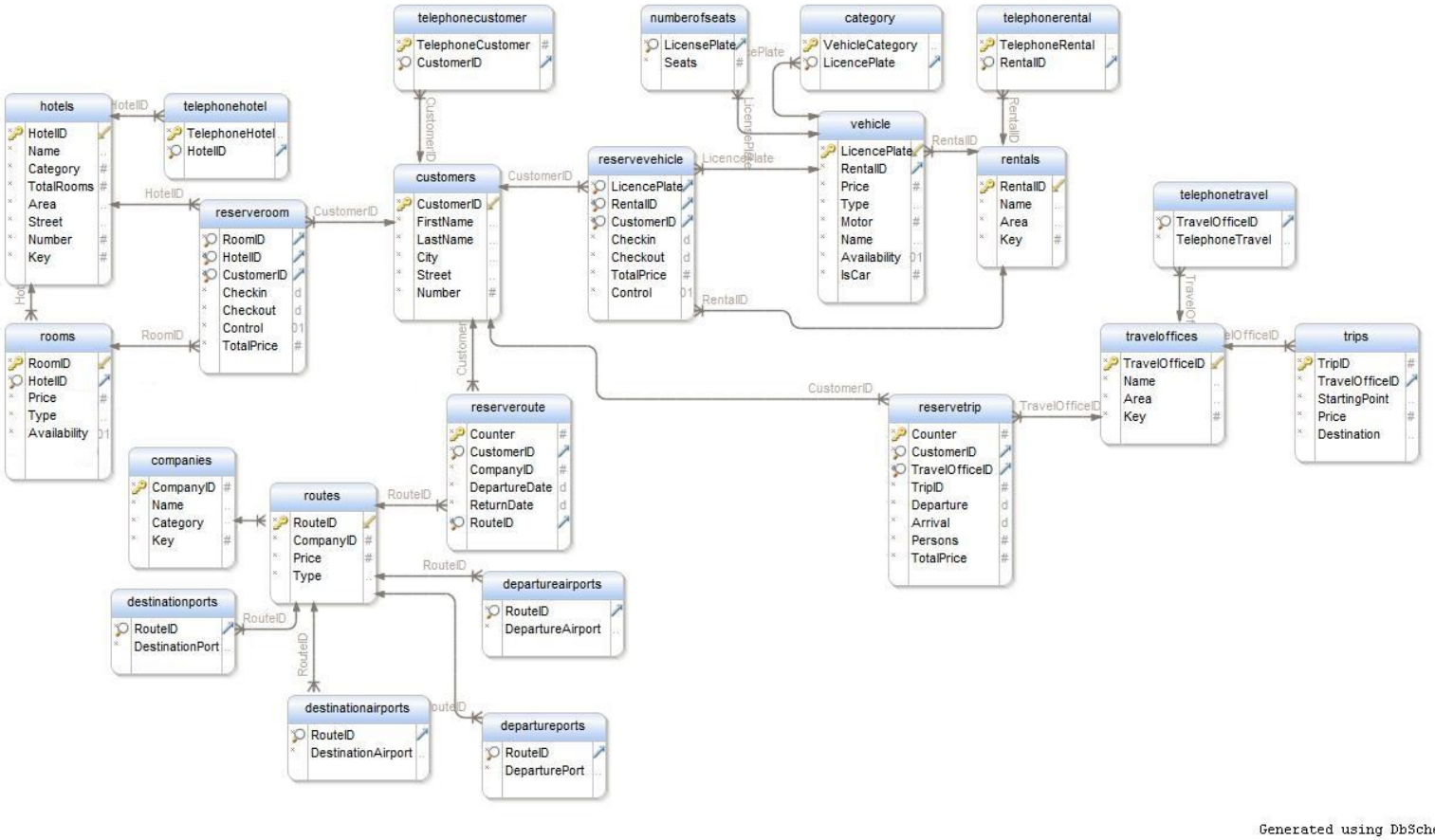
Σε πρώτο στάδιο έπρεπε να βρούμε ποια από τα δεδομένα μας θα ήταν οι οντότητες και ποια οι συσχετίσεις. Μία οντότητα μπορούμε να θεωρήσουμε ένα αντικείμενο, ένα γεγονός η μία κατάσταση, ενώ μία συσχέτιση τη σύνδεση μεταξύ οντοτήτων. Η κύρια οντότητα που δημιουργήσαμε είναι η οντότητα Customer, η οποία αντιπροσωπεύει τον χρήστη της συσκευής. Στη συνέχεια δημιουργήσαμε τις οντότητας Hotel, Room, Rental, Vehicle, Company, Route, TravelOffice και Trip. Η οντότητα Hotel αντιπροσωπεύει τα ξενοδοχεία της περιοχής που έχουμε επιλέξει ενώ η Room, τα δωμάτια του κάθε ξενοδοχείου. Η οντότητα Rental αντιστοιχεί στα γραφεία ενοικιάσεων και η οντότητα Vehicle αναφέρεται στο κάθε όχημα που ανήκει στο εκάστοτε γραφείο. Για τα διαθέσιμα δρομολόγια που μπορεί να βρει ο χρήστης δημιουργήσαμε την οντότητα Route, το κάθε στιγμιότυπο της οποίας ανήκει στην οντότητα Company, τις εταιρίες που ανήκουν τα δρομολόγια. Τέλος για τα γραφεία που διοργανώνουν εκδρομές δημιουργήθηκαν οι αντίστοιχες οντότητες TravelOffice και Trip.

Η κάθε οντότητα συνοδεύεται από ένα σύνολο ιδιοτήτων που την περιγράφουν, οι οποίες ονομάζονται γνωρίσματα. Για να γίνει κατανοητή η διαδικασία δημιουργίας του σχεσιακού μοντέλου θα εξηγήσουμε τον τρόπο σκέψης που ακολουθήσαμε. Η οντότητα Hotel συνδέεται με την οντότητα Room μέσω της συσχέτισης «belongs». Η πληθικότητα της σχέσης αυτής είναι 1:N διότι σε ένα ξενοδοχείο μπορούν να ανήκουν πολλά δωμάτια, αλλά κάθε δωμάτιο ανήκει σε ένα ξενοδοχείο. Ο Customer μπορεί να κάνει κράτηση μέσω της συσχέτισης «reserveR» σε ένα ή περισσότερα δωμάτια καταλήγοντας σε λόγο πληθικότητας N:M, διότι ο Customer μπορεί να κάνει κράτηση σε πολλά δωμάτια και σε κάθε δωμάτιο μπορεί να γίνει κράτηση από πολλούς πελάτες, υπό την προϋπόθεση η κράτηση να είναι για διαφορετική ημερομηνία. Το ίδιο συμβαίνει και με τις οντότητες Rental και Vehicle οι οποίες συνδέονται με τη σχέση «has» καθώς και με τις υπόλοιπες οντότητες όπως φαίνεται στο παρακάτω σχήμα.



Εικόνα 28: Μοντέλο Οντοτήτων-Συσχετίσεων

Σε δεύτερο στάδιο έπρεπε να μεταφέρουμε το παραπάνω σχεσιακό μοντέλο σε αντίστοιχους πίνακας μέσω της phpMyAdmin. Στο γραφικό περιβάλλον της phpMyAdmin αντιστοιχίσαμε τις οντότητες, τις συσχετίσεις αλλά και ορισμένα γνωρίσματα σε πίνακες ορίζοντας τα αντίστοιχα κλειδιά ώστε να επιτευχθεί η διασύνδεση μεταξύ των πινάκων. Κάνοντας εξαγωγή της βάσης μας μέσω του προγράμματος DbSchema έχουμε το παρακάτω σχεσιακό διάγραμμα E-R.



Εικόνα 29: Σχεσιακό διάγραμμα E-R βάσης δεδομένων

3.3.1 Availability & Control

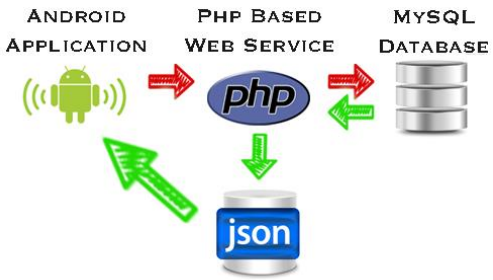
Όπως προαναφέραμε πρωταρχικό ρόλο στη βάση μας έχει ο πίνακας Customer. Όταν ο χρήστης (Customer) θέλει να κάνει κράτηση μίας υπηρεσίας τότε αυτή η εγγραφή γίνεται στον αντίστοιχο πίνακα. Για τον λόγο αυτό, συσχετίσεις όπως «reserveR» και «reserveV» μετατράπηκαν σε αντίστοιχους πίνακες. Όταν ο χρήστης κάνει κράτηση ενός δωματίου για μία συγκεκριμένη ημερομηνία τότε αυτή η εγγραφή προστίθεται στον πίνακα reserveroom. Το γνώρισμα Availability στον πίνακα rooms έχει τον ρόλο ενός ψηφίου ελέγχου. Όταν ο Customer θέλει να κάνει κράτηση ενός δωματίου, τότε με το αντίστοιχο ερώτημα sql η βάση αρχικά θα κάνει αναζήτηση ενός δωματίου πρώτα στον πίνακα rooms και θα επιστρέψει τα δωμάτια τα οποία έχουν στο Availability την τιμή 1. Για να έχει ένα δωμάτιο Availability ίσον με 1 συνεπάγεται στο γεγονός ότι δεν έχει γίνει ποτέ κράτηση σε αυτό το δωμάτιο. Στην περίπτωση που ένα δωμάτιο έχει Availability ίσον με 0, δηλαδή υπάρχει κάποια κράτηση για αυτό το δωμάτιο, τότε η βάση θα κάνει αναζήτηση στον πίνακα reserveroom βάση ημερομηνίας και θα επιστρέψει τα δωμάτια των οποίων η ημερομηνία κράτησης

δεν συγκρούεται με την ημερομηνία που έχει θέσει ο χρήστης. Με την ίδια λογική ακολουθήσαμε την διαδικασία αναζήτησης ενός διαθέσιμου οχήματος (vehicle).

Το γνώρισμα Control που υπάρχει στους πίνακες reserveroom και reservevehicle παίζει σημαντικό ρόλο τόσο στην αναζήτηση όσο και στην ορθή επιστροφή των διαθέσιμων δωματίων. Για να κατανοήσουμε τη λειτουργία του ας υποθέσουμε ότι ο χρήστης έχει κάνει κράτηση στο δωμάτιο 101 για τις ημερομηνίες 10/06/2014 έως 16/06/2014, τότε θα προστεθεί μία εγγραφή στον πίνακα reserveroom με τις αντίστοιχες ημερομηνίες. Στην περίπτωση που ο χρήστης κάνει και δεύτερη κράτηση στο δωμάτιο 101 για τις ημερομηνίες 20/06/2014 έως 23/06/2014 και θέλει να κάνει και τρίτη κράτηση για το δωμάτιο 101 τότε θα προέκυπτε η εξής διαδικασία. Εφόσον το Availability στον πίνακα rooms είναι 0 για το δωμάτιο 101 τότε η βάση μας θα έκανε έλεγχο για διαθεσιμότητα στον πίνακα reserveroom. Στον πίνακα κρατήσεων των δωματίων η βάση θα έκανε ερώτημα σε κάθε εγγραφή ξεχωριστά βάση ημερομηνίας. Όταν ο χρήστης ήθελε να κλείσει το δωμάτιο 101, για τρίτη φορά, για τις ημερομηνίες 14/06/2014 έως 17/06/2014 τότε θα προέκυπτε το εξής λάθος. Στην πρώτη εγγραφή το δωμάτιο 101 θα απορριπτόταν για το λόγο ότι συγκρούονται οι ημερομηνίες, αντίθετα στη δεύτερη εγγραφή, η ημερομηνία που επέλεξε ο χρήστης περνάει από τον έλεγχο και λανθασμένα η βάση επιστρέφει ως διαθέσιμο δωμάτιο το 101. Για την αντιμετώπιση του προβλήματος προστέθηκε το ψηφίο ελέγχου Control, το οποίο λειτουργεί ως εξής, αρχικά η βάση θα έκανε ερώτημα για μη διαθέσιμα δωμάτια και θα επέστρεφε το αντίστοιχο roomId του κάθε δωματίου. Έχοντας στη διάθεσή μας τα μη διαθέσιμα δωμάτια, στον πίνακα reserveroom θα μετατρέπαμε το Control από 1 σε 0. Έπειτα γίνεται ερώτημα sql για να μας φέρει όσα δωμάτια έχουν Control 1, δηλαδή τα έγκυρα πλέον δωμάτια.

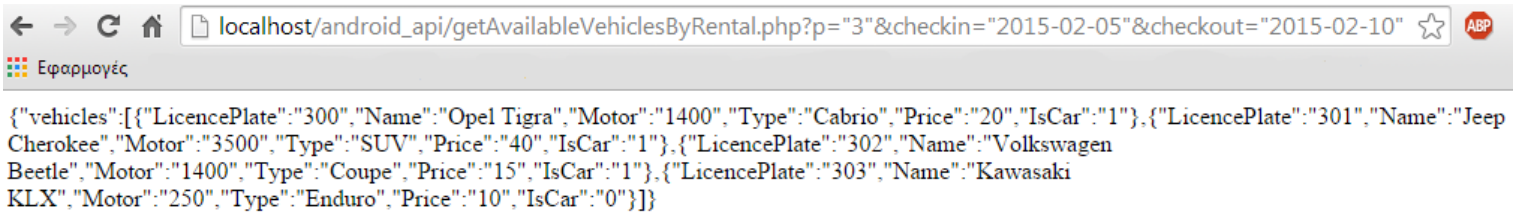
3.3.2 Ανταλλαγή δεδομένων μεταξύ Android & Database

Για να επικοινωνήσει το Android με τη βάση δεδομένων χρησιμοποιεί κατάλληλα ερωτήματα μέσω PHP αρχείων. Αρχικά η εφαρμογή στέλνει SQL queries στον web service. Αυτός με τη σειρά του διαβάζει τα ερωτήματα αυτά και τα αποκωδικοποιεί ώστε να εκτελεστούν στη βάση δεδομένων. Η βάση δεδομένων παράγει τα αποτελέσματα τα οποία διαβάζει ο web service και τα μετατρέπει σε JSON μορφή ώστε να μπορέσουν να διαβαστούν από το Android.



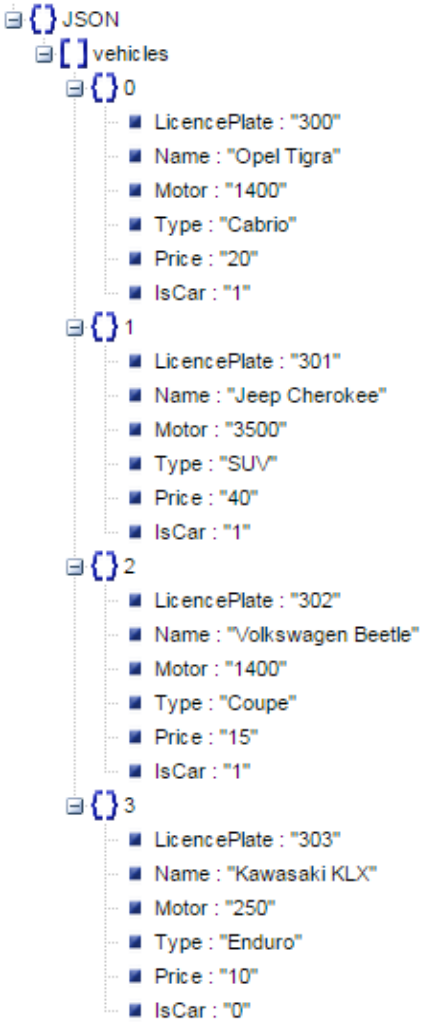
Εικόνα 30: Διαδικασία επικοινωνίας Android-Database

Το JSON (JavaScript Object Notation) είναι ένα πρότυπο ανταλλαγής δεδομένων το οποίο διευκολύνει τους ανθρώπους στην ανάγνωση και χρησιμοποιείται για μετάδοση δεδομένων μεταξύ Server και Client. Η δομή του JSON αποτελείται από ένα σύνολο ονομάτων και των αντίστοιχων τιμών (name/value). Το σύνολο των ονομάτων-τιμών που υπάρχουν στη δομή του JSON ονομάζεται Object και διακρίνεται ανάμεσα στα άγκιστρα ({…..}). Πολλά Object μαζί συνθέτουν ένα JSON Array. Στη συνέχεια ακολουθεί ένα παράδειγμα στο οποίο κάνουμε αίτημα (GET) για τα διαθέσιμα οχήματα της εφαρμογής μας και πως αυτά εμφανίζονται σε JSON μορφή.



Εικόνα 31: Μορφή JSON στον browser

Στον browser διακρίνουμε ένα JSON Array με την ονομασία vehicles το οποίο περιέχει τέσσερα JSON Objects κάθε ένα εκ των οποίων περιέχει έξι ζευγάρια ονόματος/τιμής. Ακολουθεί το ιεραρχικό διάγραμμα του JSON Array για να κατανοήσουμε καλύτερα την παραπάνω εικόνα.



Εικόνα 32: Απεικόνιση ενός JSON Array

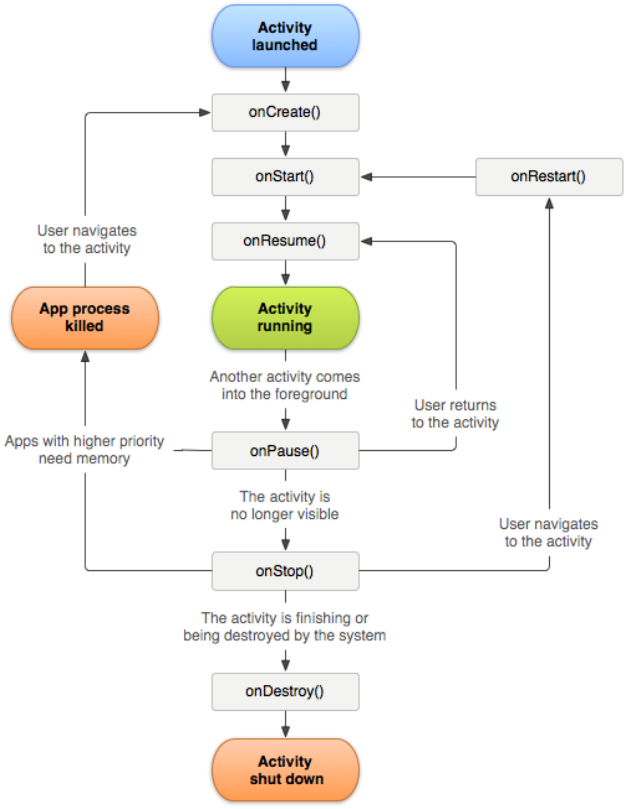
4 Προγραμματισμός στο Android

Σε αυτό το κεφάλαιο θα αναφερθούμε στην ανατομία της εφαρμογής και στον τρόπο με τον οποίο δημιουργήθηκε. Τα βασικά συστατικά της εφαρμογής είναι οι κλάσεις Java και τα xml αρχεία που συνθέτουν ένα Activity καθώς και η συγγραφή PHP αρχείων τα οποία περιέχουν τα SQL queries. Θα ξεκινήσουμε την αναλυτική περιγραφή όλων των παραπάνω αφού πρώτα κατανοήσουμε την έννοια, τη δομή και πως λειτουργεί ένα Activity.

4.1 Κύκλος ζωής ενός Activity

Η κάθε εφαρμογή Android αποτελείται από ένα σύνολο εργασιών στο οποίο κάθε εργασία ονομάζεται Activity. Κάθε Activity μέσα σε μία εφαρμογή είναι ένα στιγμιότυπο της οθόνης το οποίο έχει ένα μοναδικό σκοπό. Κάθε εφαρμογή Android περιλαμβάνει πολλά Activity και το λειτουργικό σύστημα Android επιτρέπει την εκτέλεση πολλών εφαρμογών ταυτόχρονα με την προϋπόθεση ότι υπάρχει η απαιτούμενη μνήμη και επεξεργαστική ισχύ. Κάθε χρονική στιγμή μόνο μία εργασία (Activity) είναι στο προσκήνιο!

Για να επιτευχθεί αυτό, το Android διαχειρίζεται τα Activity τοποθετώντας τα σε μία στοίβα. Η στοίβα λειτουργεί ως εξής. Όταν ξεκινάει ένα καινούργιο Activity, το Activity που ήταν μέχρι τώρα στο προσκήνιο διακόπτεται προσωρινά και το νέο Activity τοποθετείται στην κορυφή της στοίβας. Όταν ένα Activity σταματήσει τότε αφαιρείται από τη στοίβα και το προηγούμενο Activity παίρνει τη θέση του. Η παραπάνω διαδικασία ονομάζεται κύκλος ζωής ενός Activity και επιτυγχάνεται μέσω των μεθόδων onCreate(), onPause(), onStart(), onRestart(), onResume(), onStop() και onDestroy() ενός Activity.



Εικόνα 33: Κύκλος ζωής ενός Activity

Η onCreate() είναι η πιο σημαντική μέθοδος αφού είναι αυτή που καλείται όταν ένα Activity ξεκινά για πρώτη φορά. Αμέσως μετά ακολουθούν οι μέθοδοι onStart() και onResume() έως ότου το Activity είναι ορατό και διαθέσιμο στο χρήστη για αλληλεπίδραση. Στην περίπτωση που ένα άλλο Activity βρεθεί στην κορυφή της στοίβας τότε στο ήδη υπάρχον Activity καλείται η μέθοδος onPause() η οποία πρέπει να αποθηκεύσει πληροφορίες κατάστασης προτού το νέο Activity βρεθεί στο προσκήνιο. Εάν για κάποιο λόγο το προηγούμενο Activity πρέπει να βρεθεί ξανά στο προσκήνιο τότε θα εκτελεστεί η μέθοδος onResume(). Η μέθοδος onStop() καλείται όταν ένα Activity δεν είναι ορατό στην οθόνη, δηλαδή δεν είναι στην κορυφή της στοίβας. Εάν ένα Activity θέλει να βρεθεί στο προσκήνιο από την κατάσταση onStop(), τότε εκτελείται η μέθοδος onRestart() και έπειτα η onStart(). Τέλος η μέθοδος onDestroy() καλείται για να καταστραφεί ένα Activity, είτε επειδή το Activity ολοκλήρωσε τον κύκλο ζωής του, είτε επειδή το λειτουργικό σύστημα δεν έχει τους απαιτούμενους πόρους.

4.2 Δομή ενός Activity

Το κάθε Activity που δημιουργήσαμε αποτελείται από ένα αρχείο Java και το αντίστοιχο XML που το περιγράφει. Στα υποκεφάλαια αυτά θα περιγράψουμε τη δομή μίας κλάσης Java ενός Activity καθώς και τη δομή του XML αρχείου που συνδέεται με την κλάση, ώστε να μπορέσουμε στη συνέχεια να κατανοήσουμε τον κώδικα της εφαρμογής μας.

4.2.1 Δομή μίας κλάσης Java

Στην κορυφή κάθε κλάσης Java υπάρχει το όνομα του πακέτου στο οποίο βρίσκεται το αρχείο, όπως αυτό το ορίσαμε κατά τη δημιουργία του project μας. Ακριβώς από κάτω δηλώνονται όλες οι βιβλιοθήκες που χρησιμοποιούνται στο Activity οι οποίες εισάγονται με το χαρακτηριστικό «import». Στη συνέχεια ακολουθεί η δήλωση της κλάσης με τη μορφή «public class *Όνομα_κλάσης* extends Activity {…}». Μέσα στην κλάση δηλώνονται οι global μεταβλητές, δηλαδή οι μεταβλητές που θα είναι ορατές σε όλο το Activity καθώς και τα στοιχεία του xml αρχείου τα οποία θέλουμε να τα αντιστοιχήσουμε με αυτά που υπάρχουν στο αρχείο xml. Έπειτα ακολουθεί ο κυρίως κώδικας ο οποίος αποτελείται από μεθόδους. Σε όλα μας τα Activity η πρώτη μέθοδος που υλοποιείται αυτόματα είναι η onCreate(), η οποία καλείται κατά τη δημιουργία ενός Activity. Μέσα στη μέθοδο onCreate() καλείται ο constructor της super class και έπειτα καλείται η μέθοδος setContentView() η οποία "ζωγραφίζει" στην οθόνη το xml που αντιστοιχεί στο Activity. Όλα τα resources (layouts, Strings, images κ.α.) που έχουμε δημιουργήσει υπάρχουν καταχωρημένα στην κλάση R.Java η οποία δημιουργείται αυτόματα και περιέχει όλα τα resources δηλωμένα με IDs.

4.2.2 Σχεδίαση διεπαφής χρήστη

4.2.2.1 Δομή ενός Layout

Ένα Layout που περιγράφει τη διεπαφή χρήστη, είναι γραμμένο στη γλώσσα σήμανσης XML και ορίζει τη γραφική αναπαράσταση ενός Activity έχοντας συγκεκριμένη δομή. Στην πρώτη γραμμή ορίζεται η έκδοση και η γλώσσα που υποστηρίζεται.

```
<?xml version="1.0" encoding="utf-8"?>
```

Στη συνέχεια ακολουθεί η δήλωση της διάταξης στην οθόνη καθώς και η δήλωση των χαρακτηριστικών που θα έχει, όπως ύψος και πλάτος.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

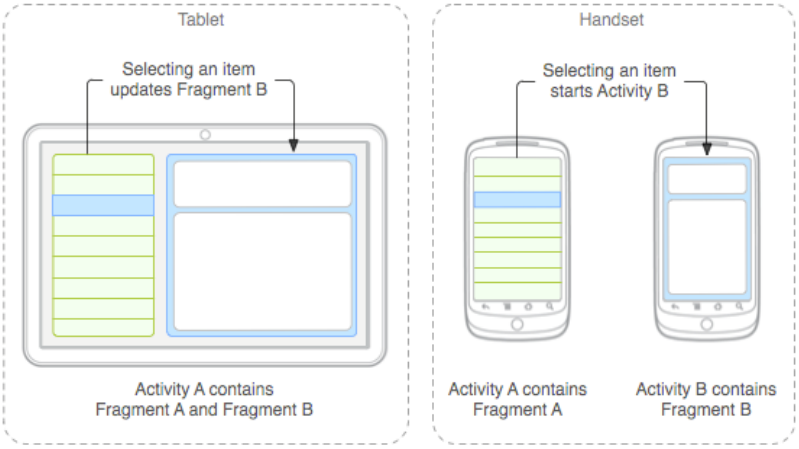
```
android:layout_height ="fill_parent"
android:layout_width ="fill_parent">
```

Από εκεί και κάτω μπορούμε να δηλώσουμε όλα τα δομικά στοιχεία (components) τα οποία θα περιέχει το Activity μας. Κάθε στοιχείο ξεκινάει με το είδος, αφού προηγηθεί το tag "<". Έπειτα μπορούμε να εισάγουμε διάφορα χαρακτηριστικά όπως το ύψος που θέλουμε να έχει, το αναγνωριστικό id, την τιμή που θα αναγράφει κ.α.. Η περιγραφή ενός στοιχείου τελειώνει πάντα με το tag "/>". Ακολουθεί ένα παράδειγμα ενός πεδίου κειμένου (TextView).

```
<TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Category: " />
```

4.2.2.2 Fragments

Ένα Fragment είναι ένα ανεξάρτητο component που παρέχει το Android και μπορεί να χρησιμοποιηθεί από ένα Activity. Ένα Fragment αντιπροσωπεύει ένα τμήμα της διεπαφής χρήστη σε ένα Activity και ένα Activity μπορεί να περιέχει πολλά Fragments. Ένα Fragment εκτελείται στο πλαίσιο μίας δραστηριότητας αλλά έχει το δικό του κύκλο ζωής και μπορεί να έχει τη δική του διεπαφή χρήστη. Το Android εισήγαγε τα Fragments στην έκδοση 3.0 (API level 11) για να υποστηρίξει μία πιο ευέλικτη και δυναμική διεπαφή χρήστη με στόχο την προσαρμογή της διεπαφής σε ένα μεγάλο εύρος οθονών συμπεριλαμβανομένων και των tablets.



Εικόνα 34: Αναπαράσταση διεπαφής με χρήση Fragment

Στην εφαρμογή μας κάνουμε χρήση των Fragments στις βασικές καρτέλες και υλοποιούμε τη μέθοδο onCreateView() η οποία καλείται για να σχεδιάσει το User Interface (UI) ενός Fragment όταν αυτό καλείται για πρώτη φορά. Πιο συγκεκριμένα κάθε μία από τις βασικές καρτέλες κάνει «extends» το ListFragment για το λόγο ότι κάθε Fragment θέλουμε να αποτελεί μία λίστα. Για να διαχειριστούμε κάποιο από τα στοιχεία μίας λίστας γίνεται χρήση της μεθόδου onListItemClick() η οποία αντιλαμβάνεται το event και ανάλογα με τη θέση του στοιχείο στη λίστα πράττει ανάλογα.

4.3 Συγγραφή κώδικα

4.3.1 MainActivity

Η πρώτη κλάση που δημιουργήσαμε είναι η MainActivity.java, η οποία είναι και το κύριο Activity της εφαρμογής μας. Για να μπορέσουμε να ενσωματώσουμε τις κύριες καρτέλες μας, η MainActivity κάνει «extends» το ActionBarActivity καθώς και «implements» το Interface, ActionBar.TabListener. Για την ενσωμάτωση των καρτελών στην Action Bar, συμβουλευτήκαμε το παρακάτω tutorial. http://sunil-android.blogspot.gr/2013/08/actionbar-tab-listfragment-in-android.html

```
public class MainActivity extends ActionBarActivity implements
ActionBar.TabListener
```

Το xml αρχείο που συνοδεύει το MainActivity ονομάζεται activity_main.xml και περιλαμβάνει τη βιβλιοθήκη ViewPager επιτρέποντας σε κάθε οθόνη που είναι στο προσκήνιο να είναι μία καρτέλα.

```
<android.support.v4.view.ViewPager
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Μέσα στην κλάση αυτή και συγκεκριμένα στη μέθοδο onCreate() ορίσαμε στο ActionBar την επιλογή NAVIGATION_MODE_TABS στη μέθοδο setNavigationMode, δίνοντάς μας την δυνατότητα να ενσωματώσουμε καρτέλες στην μπάρα ενεργειών.

```
final ActionBar actionBar = getActionBar();
actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
```

Έπειτα με τον παρακάτω κώδικα δημιουργήσαμε και δώσαμε τίτλο σε κάθε καρτέλα ξεχωριστά.

```
actionBar.addTab(actionBar.newTab().setText("Hotels").setTabListener(this)
);

actionBar.addTab(actionBar.newTab().setText("Rentals").setTabListener(this
));

actionBar.addTab(actionBar.newTab().setText("Trips").setTabListener(this))
;
        actionBar.addTab(actionBar.newTab().setText("Flights &
Ferries").setTabListener(this));

actionBar.addTab(actionBar.newTab().setText("Attractions").setTabListener(
this));
```

Μέσω των API's που διαθέτει το Android, χρησιμοποιήσαμε την βιβλιοθήκη android.support.v4.app.FragmentPagerAdapter για να διαχειριστούμε τη καρτέλα που θα είναι στο προσκήνιο. Έτσι λοιπόν δημιουργήσαμε μία νέα εμφωλευμένη κλάση, την AppSectionsPagerAdapter, μέσα στην MainActivity, η οποία κάνει «extends» το FragmentPagerAdapter και επιστρέφει την εκάστοτε καρτέλα που έχει επιλέξει ο χρήστης μέσω της μεθόδου getItem(int i). Μέσα στη μέθοδο getItem() δηλώσαμε ένα switch statement και ανάλογα με την τιμή που έχει η ακέραια μεταβλητή i καλείται ο αντίστοιχος constructor του Fragment. Αφού δημιουργηθεί το Fragment η getItem() είναι υπεύθυνη να επιστρέψει το Fragment ώστε να βρεθεί στο προσκήνιο.

```
  public static class AppSectionsPagerAdapter extends FragmentPagerAdapter
{

        public AppSectionsPagerAdapter(FragmentManager fm) {
            super(fm);
        }

        @Override
        public Fragment getItem(int i) {
            switch (i) {
                case 0:
                    HotelsFragment hotelsFragment=new HotelsFragment();
                    return hotelsFragment;

                case 1:
                    RentalsFragment rentalsFragment=new RentalsFragment();
                    return rentalsFragment;

                case 2:
                    TripsFragment tripsFragment=new TripsFragment();
                    return tripsFragment;

                case 3:
                    FlightsAndFerriesFragment
flightsAndFerriesFragment=new FlightsAndFerriesFragment();
                    return flightsAndFerriesFragment;

                case 4:
                     AttractionsFragment attractionsFragment=new
AttractionsFragment();
                     return attractionsFragment;

                default:
                    return null;
            }
        }
        @Override
        public int getCount(){
            return 5;
        }
        @Override
        public CharSequence getPageTitle(int position) {
        }
    }
```

Έπειτα για να εμφανιστεί η επιλεγμένη καρτέλα στο προσκήνιο, δηλώσαμε σαν global μεταβλητή ένα στιγμιότυπο της κλάσης AppSectionsPagerAdapter με όνομα appSectionsPagerAdapter και στη μέθοδο onCreate() καλέσαμε τον constructor της καινούργιας κλάσης, δίνοντας του σαν όρισμα το Fragment που πρέπει να είναι στο προσκήνιο κάθε φορά. Τον ρόλο αυτό τον αναλαμβάνει η μέθοδος getSupportFragmentManager().

```
appSectionsPagerAdapter = new
AppSectionsPagerAdapter(getSupportFragmentManager());
```

Στη συνέχεια δηλώσαμε σαν global μεταβλητή τη viewPager η οποία είναι τύπου ViewPager ώστε να μπορέσουμε να καλέσουμε τη μέθοδο setAdapter() και να της δώσουμε σαν όρισμα το Fragment που επέστρεψε ο constructor της κλάσης AppSectionPagerAdapter. Για να μπορέσουμε να κάνουμε την αντιστοιχία του viewPager με το αντίστοιχο στο xml, το οποίο ονομάζεται container, έπρεπε να καλέσουμε τη μέθοδο findViewById().

```
viewPager = (ViewPager) findViewById(R.id.container);
viewPager.setAdapter(appSectionsPagerAdapter);
```

Στο επόμενο βήμα υλοποιήσαμε τη μέθοδο onTabSelected() του Interface ActionBar.TabListener ώστε ανάλογα με τη θέση (position) που έχει η κάθε καρτέλα να μπορεί το στιγμιότυπο viewPager να εμφανίζει το αντίστοιχο περιεχόμενο.

```
   @Override
    public void onTabSelected(ActionBar.Tab tab, FragmentTransaction
fragmentTransaction) {
        viewPager.setCurrentItem(tab.getPosition());
}
```

Για να ενσωματώσουμε το καλάθι αγορών (shopping cart) και τη ενέργεια της αναζήτησης που θέλαμε να βρίσκονται στη μπάρα ενεργειών (ActionBar), υλοποιήσαμε τις δύο Override μεθόδους του ActionBarActivity, τις onCreateOptionMenu() και onOptionItemSelected(). Η πρώτη μέθοδος αναλαμβάνει να συμπεριλάβει το αντίστοιχο xml από το φάκελο menu, το οποίο περιέχει τις ενέργειες που θα έχει η ActionBar.

```
@Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.activity_main_actions, menu);
        return super.onCreateOptionsMenu(menu);
    }
```

Για να προσθέσουμε μία ενέργεια στο xml του menu πρέπει να αναφερθούμε με το tag "<item.. />". Από εκεί και έπειτα μπορούμε να του δώσουμε ιδιότητες όπως κάποιο χαρακτηριστικό id, το εικονίδιο που θα εμφανίζεται στην ActionBar καθώς και τον τίτλο. Ακολουθεί ο αντίστοιχος κώδικας του activity_main_actions.

```
<menu 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"
    tools:context=".MainActivity" >

    <item android:id="@+id/shopping_cart"
        android:icon="@drawable/shopping_cart_64"
        android:title="shopping_cart"
        app:showAsAction="ifRoom" />

    <item android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:title="action_search"
        app:showAsAction="ifRoom" />

    <item android:id="@+id/action_settings"
        android:title="@string/action_settings"
```

```
        android:orderInCategory="100"
        app:showAsAction="never"/>
</menu>
```

Σε αυτό το σημείο θα πρέπει να κατανοήσουμε τον τρόπο με τον οποίο εκκινείτε ένα καινούργιο Activity καθώς και τον τρόπο με τον οποίο μεταβιβάζονται δεδομένα από ένα Activity σε ένα άλλο. Το Android χρησιμοποιεί ένα μηχανισμό με προθέσεις ώστε να ξεκινήσει ένα καινούργιο Activity. Ένα intent αποτελεί μία πρόθεση και εστιάζει στην πραγματοποίηση μίας ενέργειας. Για να γίνει εκκίνηση ενός καινούργιου Activity πρέπει να γίνει κλήση της μεθόδου startActivity(), η οποία παίρνει σαν όρισμα ένα intent. Για να δημιουργήσουμε ένα στιγμιότυπο της κλάσης Intent πρέπει να δώσουμε σαν πρώτο όρισμα το context της εφαρμογής, το οποίο περιέχει όλες τις λειτουργίες της εφαρμογής μέσω της μεθόδου getApplicationContext() και σαν δεύτερο όρισμα τον τίτλο του Activity που θέλουμε να ξεκινήσει. Για να μπορέσουμε να μεταβιβάσουμε δεδομένα μεταξύ των Activity χρησιμοποιούμε τις μεθόδους που προσφέρει ένα στιγμιότυπο της κλάσης Intent. Κάνοντας χρήση της μεθόδου intent.putExtra() μπορούμε να στείλουμε δεδομένα από ένα Activity σε ένα άλλο. Το Activity που λαμβάνει τα δεδομένα για επεξεργασία, πρέπει να εκτελέσει την εντολή intent.getExtra().

Η δεύτερη μέθοδος onOptionsItemSelected είναι υπεύθυνη για να αναγνωρίσει την ενέργεια που επιλέχθηκε από το χρήστη και να πράξει ανάλογα. Με κύριο γνώμονα το αναγνωριστικό id που έχει κάθε δομικό στοιχείο (component) το οποίο είναι καταγεγραμμένο στην κλάση R.Java, το Android μπορεί να ξεχωρίσει ποια ενέργεια επιλέχθηκε. Σαν αποτέλεσμα όταν ο χρήστης επιλέξει το κουμπί της αναζήτησης το λειτουργικό σύστημα αντιλαμβάνεται την ενέργεια αυτή και εκκινεί το καινούργιο Activity το οποίο περιέχει τις λειτουργίες της αναζήτησης. Η ίδια διαδικασία ακολουθείται και στο κουμπί για το καλάθι αγορών.

```
@Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if (id == R.id.action_search) {
            Intent intent=new Intent(getApplicationContext(),
SearchActivity.class);
            startActivity(intent);
            return true;
        }
        else if(id==R.id.action_settings) {
              return true;
        }
        else if(id==R.id.shopping_cart) {
            Intent intent=new Intent(getApplicationContext(),
ShoppingCart.class);
            startActivity(intent);
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
```

4.3.2 Η καρτέλα HotelsFragment

Η συγγραφή του κώδικα που ακολουθήσαμε για τα Fragments των κυρίων καρτελών της εφαρμογής μας είναι η ίδια. Για αυτό το λόγο θα περιγράψουμε τον κώδικα της πρώτης καρτέλας Hotels. Για την καρτέλα Hotels δημιουργήσαμε το Activity HotelsFragment το οποίο κάνει «extends» το ListFragment δηλώνοντας έτσι, ότι το περιεχόμενο του Activity θα αποτελεί μία λίστα. Μέσα στη κλάση δηλώσαμε και αρχικοποιήσαμε ένα πίνακα συμβολοσειρών με την ονομασία "myHotels[]"

στον οποίο προσθέσαμε τα ονόματα των ξενοδοχείων. Στη συνέχεια δηλώσαμε και αρχικοποιήσαμε ένα πίνακα ακεραίων με την ονομασία "stars[]" στον οποίο προσθέσαμε τις εικόνες με τα αστέρια.

```
public class HotelsFragment extends ListFragment {

    String[] myHotels = new String[] {
            "Elotia Hotel","Nefeli Hotel","Kriti Hotel","El Greco Hotel",
            "Samaria Hotel","Minoa Hotel","Kydon Hotel","Akali Hotel",
            "Santa Marina Plaza","Porto Platanias Beach Resort & Spa"};

    int[] stars = new int[]{
            R.drawable.rate2,
            R.drawable.rate3,
            R.drawable.rate4,
            R.drawable.rate5,
    };
```

Μέσα στη μέθοδο onCreateView() δημιουργήσαμε μία λίστα που θα περιέχει πολλά HashMap. Ένα HashMap είναι μία δομή που επιτρέπει την αποθήκευση ζευγαριών κλειδιού/τιμής, στην περίπτωσή μας ξενοδοχείο/αστέρια. Υλοποιώντας τη μέθοδο επανάληψης for προσθέσαμε σε κάθε HashMap το όνομα του ξενοδοχείου με την αντίστοιχη εικόνα των αστεριών. Στη συνέχεια δημιουργήσαμε δύο πίνακες για να μπορέσουμε να αντιστοιχήσουμε τις τιμές που έχουμε δώσει στο HashMap με τα components που βρίσκονται στο αντίστοιχο xml αρχείο κάνοντας χρήση ενός στιγμιότυπου της κλάσης SimpleAdapter. Ο SimpleAdapter ανήκει στην οικογένεια των Adapter οι οποίοι λειτουργούν ως μία γέφυρα μεταξύ των δεδομένων και των δομικών στοιχείων και παρέχονται από τις βιβλιοθήκες του Android.

```
@Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,Bundle savedInstanceState) {
        List<HashMap<String,String>> aList = new
ArrayList<HashMap<String,String>>();

      for(int i=0;i<2;i++){
            HashMap<String, String> hm = new HashMap<String,String>();
            hm.put("hotelName", myHotels[i]);
            hm.put("stars",Integer.toString(stars[0]));
            aList.add(hm);
        }
        for(int i=2;i<5;i++){
            HashMap<String, String> hm = new HashMap<String,String>();
            hm.put("hotelName", myHotels[i]);
            hm.put("stars",Integer.toString(stars[1]));
            aList.add(hm);
        }
        for(int i=5;i<8;i++){
            HashMap<String, String> hm = new HashMap<String,String>();
            hm.put("hotelName", myHotels[i]);
            hm.put("stars",Integer.toString(stars[2]));
            aList.add(hm);
        }
        for(int i=8;i<10;i++){
            HashMap<String, String> hm = new HashMap<String,String>();
            hm.put("hotelName", myHotels[i]);
```

```
            hm.put("stars",Integer.toString(stars[3]));
            aList.add(hm);
        }

        String[] from = { "hotelName","stars" };

        int[] to = { R.id.hotelName,R.id.stars};

        SimpleAdapter adapter = new
SimpleAdapter(getActivity().getBaseContext(), aList,
R.layout.hotels_fragment_list, from, to);
        setListAdapter(adapter);
        return super.onCreateView(inflater, container,
savedInstanceState);
    }
```

Τέλος υλοποιήσαμε τη μέθοδο onListItemClick() δημιουργώντας ένα switch statement στο οποίο ανάλογα με τη θέση (position) που έγινε κλικ σε κάποιο στοιχείο της λίστας, να εκκινεί και το αντίστοιχο Activity.

```
@Override
    public void onListItemClick(ListView list, View v, int position, long
id) {

        Intent intent;

        switch (position) {
            case 0:
                intent=new Intent(v.getContext(),ElotiaHotel.class);
                startActivity(intent);
                break;
            case 1:
                intent=new Intent(v.getContext(),NefeliHotel.class);
                startActivity(intent);
                break;
            case 2:
                intent=new Intent(v.getContext(),KritiHotel.class);
                startActivity(intent);
                break;
            case 3:
                intent=new Intent(v.getContext(),ElGrecoHotel.class);
                startActivity(intent);
                break;
            case 4:
                intent=new Intent(v.getContext(),SamariaHotel.class);
                startActivity(intent);
                break;
            case 5:
                intent=new Intent(v.getContext(),MinoaHotel.class);
                startActivity(intent);
                break;
            case 6:
                intent=new Intent(v.getContext(),KydonHotel.class);
                startActivity(intent);
```

```
                break;
            case 7:
                intent=new Intent(v.getContext(),AkaliHotel.class);
                startActivity(intent);
                break;
            case 8:
                intent=new Intent(v.getContext(),SantaMarinaPlaza.class);
                startActivity(intent);
                break;
            case 9:
                intent=new Intent(v.getContext(),PortoPlatanias.class);
                startActivity(intent);
                break;
            default:
        }
    }
```

Οι βιβλιοθήκες που έγιναν import για την δημιουργία της παραπάνω κλάσης είναι οι ακόλουθες.

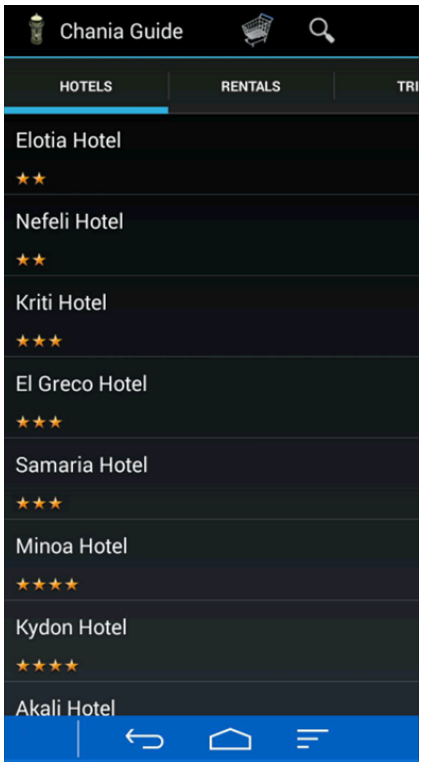
```
import android.content.Intent;
import android.os.Bundle;
import android.support.v4.app.ListFragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import com.example.mixalis.final_project.hotels.*;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
```

Το xml αρχείο που περιγράφει τη διεπαφή χρήστη του HotelsFragment ονομάζεται hotels_fragment_list.xml και περιέχει ένα πεδίο κειμένου και ένα πεδίο εικόνας.

```
<?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="match_parent"
    android:orientation="horizontal" >

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

        <TextView
            android:id="@+id/hotelName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="18dp"
            android:textColor="@android:color/white"
            android:paddingTop="10dp"
            android:paddingRight="10dp"
```

```
            android:paddingLeft="10dp"
            android:paddingBottom="10dp"/>
        <ImageView
            android:id="@+id/stars"
            android:layout_width="80dp"
            android:layout_height="25dp"
            android:paddingLeft="10dp"/>
    </LinearLayout>
</LinearLayout>
```

Το αποτέλεσμα του παραπάνω xml αρχείου σε συνδυασμό με τον κώδικα Java φαίνεται στην παρακάτω εικόνα και είναι αυτό που είναι ορατό στον χρήστη μέσω της Android συσκευής του.



Εικόνα 35: Διεπαφή χρήστη HotelsFragment

4.3.3 Book a room

Σε αυτό το υποκεφάλαιο θα εξηγήσουμε αναλυτικά τον κώδικα για τη διαδικασία κράτησης ενός δωματίου, αφού πρώτα εξηγήσουμε κάποιες βασικές κλάσεις που χρησιμοποιήθηκαν για την επικοινωνία με τη βάση δεδομένων, καθώς και την διαχείριση χρονοβόρων ενεργειών όπως για παράδειγμα το "parsing" δεδομένων από τη βάση μας.

Για να μπορέσουμε να τραβήξουμε ή να στείλουμε δεδομένα στη βάση μας έπρεπε να δημιουργήσουμε μία νέα κλάση την ServiceHandler η οποία έπρεπε να διαχειρίζεται τα HTTP αιτήματα GET και POST. Σκοπός της κλάσης αυτής είναι να αντιλαμβάνεται τα αιτήματα που στέλνει ο client και να επιστρέφει την απάντηση (response) από τον server. Για το λόγο αυτό υλοποιήσαμε τη μέθοδο makeServiceCall() βάση των βιβλιοθηκών που παρέχει το Android ώστε να μπορέσουμε να εξασφαλίσουμε την επικοινωνία με τη βάση δεδομένων. Η μέθοδος makeServiceCall() παίρνει σαν πρώτο όρισμα το URL και σαν δεύτερο όρισμα έναν ακέραιο αριθμό ο οποίος δηλώνει αν το αίτημα είναι GET ή POST και επιστρέφει την απάντηση από τον server σε μία μεταβλητή τύπου String. Τα αντίστοιχα PHP αρχεία που περιέχουν τα SQL queries μέσω της εντολής json_encode προβάλλουν

στον browser το response του server σε μορφή JSON. Παρακάτω εμφανίζεται ο κώδικας της κλάσης ServiceHandler.

```
package com.example.mixalis.final_project.support;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.util.List;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.utils.URLEncodedUtils;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.util.EntityUtils;

public class ServiceHandler {
    static String response = null;
    public final static int GET = 1;
    public final static int POST = 2;

    public ServiceHandler() {
    }

    public String makeServiceCall(String url, int method) {
        return this.makeServiceCall(url, method, null);
    }

    public String makeServiceCall(String url, int method,
                                  List<NameValuePair> params) {
        try {
            DefaultHttpClient httpClient = new DefaultHttpClient();
            HttpEntity httpEntity = null;
            HttpResponse httpResponse = null;

            if (method == POST) {
                HttpPost httpPost = new HttpPost(url);
                if (params != null) {
                    httpPost.setEntity(new UrlEncodedFormEntity(params));
                }

                httpResponse = httpClient.execute(httpPost);

            } else if (method == GET) {
                if (params != null) {
                    String paramString = URLEncodedUtils
                            .format(params, "utf-8");
                    url += "?" + paramString;
                }
                HttpGet httpGet = new HttpGet(url);

                httpResponse = httpClient.execute(httpGet);
```

```
            }
            httpEntity = httpResponse.getEntity();
            response = EntityUtils.toString(httpEntity);

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

        return response;

    }
}
```

Ορισμένες εφαρμογές απαιτούν μεγάλο χρόνο για να μπορέσουν να επεξεργαστούν τα δεδομένα ώστε να τα προβάλλουν στη διεπαφή χρήστη. Ένα τέτοιο παράδειγμα είναι η αίτηση δεδομένων από μία βάση. Το γεγονός αυτό σημαίνει ότι χρονοβόρες ενέργειες δεν πρέπει να επιβραδύνουν ή να μπλοκάρουν τη διεπαφή χρήστη (UI). Το Android παρέχει μία ειδική κλάση την AsyncTask η οποία επιτρέπει την επεξεργασία δεδομένων στο παρασκήνιο και προβολή του UI όταν η επεξεργασία τελειώσει. Μπορούμε να χρησιμοποιήσουμε την AsyncTask δημιουργώντας μία νέα εμφωλευμένη κλάση μέσα στην ήδη υπάρχουσα που κάνει «extends» την AsyncTask και υλοποιώντας τις μεθόδους onPreExecute(), doInBackground() και την onPostExecute(). Η μέθοδος onPreExecute() εκτελείται πριν ξεκινήσει η επεξεργασία δεδομένων στο παρασκήνιο. Η doInBackground() είναι υπεύθυνη για την επεξεργασία των δεδομένων στο παρασκήνιο και η μέθοδος onPostExecute() ενημερώνει τη διεπαφή χρήστη προσφέροντας τα δεδομένα που επεξεργάστηκαν. Τέλος για να χρησιμοποιήσουμε την κλάση που κάνει «extends» την AsyncTask πρέπει στην onCreate() της κύριας κλάσης να εκτελέσουμε την εντολή «new *Όνομα_κλάσης*.execute()». Ένα παράδειγμα δομής AsyncTask φαίνεται παρακάτω.

```
class Όνομα_κλάσης extends AsyncTask<String, String, String> {
   @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }
   @Override
        protected String doInBackground(String... args) {
      try{
         }
      catch (Exception e){
      Log.e("Failed")
                          }
        }
    @Override
       protected void onPostExecute(String file_url) {
            super.onPostExecute(file_url);
        }
}
```

Αφού περιγράψαμε τις κλάσεις ServiceHandler και AsyncTask μπορούμε τώρα να αναλύσουμε τον κώδικα που συγγράψαμε για τη διαδικασία online κράτησης ενός δωματίου από τα

διαθέσιμα ξενοδοχεία που βρίσκονται στην καρτέλα "Hotels". Θα δούμε τον κώδικα του Activity ElGrecoHotel.java ο οποίος ακολουθήθηκε ομοίως και για τα υπόλοιπα ξενοδοχεία.

Η λειτουργία του Activity ElGrecoHotel.java είναι να κάνει ένα αίτημα GET στη βάση δεδομένων και να επιστρέφει τις πληροφορίες του ξενοδοχείου. Έχοντας δημιουργήσει έναν τοπικό server με τη βοήθεια του WAMP, δώσαμε την στατική IP στον υπολογιστή 192.168.1.3 και όλα τα αρχεία PHP που περιέχουν τα SQL queries βρίσκονται στο σκληρό δίσκο, στο φάκελο android_api (C:\wamp\www\android_api). Μέσα στην κλάση του Activity δηλώσαμε, εκτός των άλλων, σαν global μεταβλητές το URL που θα χρησιμοποιηθεί για το αίτημα GET και ένα String hotelID το οποίο το προσθέσαμε σαν παράμετρο στο URL. Έπειτα δηλώσαμε ένα JSONArray με την ονομασία "hotel" και σαν στατικές μεταβλητές τα ονόματα των στοιχείων που περιέχονται σε κάθε objects του JSON που επιστρέφει ο server.

```
private Button findRoom;
private ImageView image,imageRate;
public String hotelID ="4";
String url =
"http://192.168.1.3/android_api/getHotelDetails.php"+"?"+"p="+ hotelID;
JSONArray hotel = null;
private TextView textView3,textView5,textView7;
private ProgressDialog pDialog;
private String category,totalRooms,area,street,number,telephone;
private static final String TAG_HOTEL = "hotels";
private static final String TAG_CATEGORY = "Category";
private static final String TAG_TOTAL_ROOMS= "TotalRooms";
private static final String TAG_AREA = "Area";
private static final String TAG_STREET = "Street";
private static final String TAG_NUMBER = "Number";
private static final String TAG_TELEPHONE = "TelephoneHotel";
```

Μέσα στη μέθοδο onCreate() δώσαμε σαν όρισμα στη μέθοδο setContentView() το hotel_details_layout το οποίο περιέχει τη διεπαφή χρήστη του Activity. Τα δομικά στοιχεία που περιέχει το hotel_details_layout.xml είναι ένα ImageView το οποίο θα έχει την εικόνα του ξενοδοχείου, ένα ImageView το οποίο θα έχει την εικόνα με την κατηγορία αστεριών, τα TextView στα οποία θα αναγράφονται οι πληροφορίες του ξενοδοχείου και ένα Button. Συνεχίζοντας μέσα στην onCreate() αντιστοιχίσαμε τα δομικά στοιχεία που δηλώσαμε στο Activity με τα αυτά του xml. Με τη μέθοδο setImageResource() "ζωγραφίσαμε" την εικόνα του ξενοδοχείου δίνοντας σαν όρισμα την εικόνα που βρίσκεται στο φάκελο drawable-mdpi. Έπειτα δηλώσαμε έναν Listener στο κουμπί "findRoom", με το πάτημα του οποίου θα μεταβαίνουμε στο επόμενο Activity. Σε αυτό το Activity στέλνουμε το id του ξενοδοχείου μέσω της μεθόδου putExtra(). Το τελευταίο πράγμα που πρέπει να κάνουμε στην onCreate() είναι να καλέσουμε την κλάση που χρησιμοποιεί τις μεθόδους του AsyncTask ώστε να λάβουμε τις πληροφορίες από τη βάση.

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.hotel_details_layout);
        getActionBar().setIcon(new
ColorDrawable(getResources().getColor(android.R.color.transparent)));

        image=(ImageView)findViewById(R.id.imageView);
        image.setImageResource(R.drawable.hotel_el_greco);

        imageRate=(ImageView)findViewById(R.id.imageView3);
```

```
        textView3=(TextView)findViewById(R.id.textView3);
        textView5=(TextView)findViewById(R.id.textView5);
        textView7=(TextView)findViewById(R.id.textView7);

        findRoom=(Button)findViewById(R.id.button);
        findRoom.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(v.getContext(),
FindRoomFromParameters.class);
                intent.putExtra("hotelID",hotelID);
                startActivity(intent);
            }
        });
        new loadHotelDetails().execute();
    }
```

Μέσα στην κύρια κλάση μας δημιουργήσαμε μία νέα εμφωλευμένη κλάση η οποία κάνει «extends» το AsyncTask. Μέσα στη μέθοδο onPreExecute() δημιουργήσαμε ένα ProgressDialog το οποίο εμφανίζεται κατά τη διάρκεια επεξεργασίας των δεδομένων. Στη συνέχεια στη μέθοδο doInBackground() δημιουργήσαμε ένα στιγμιότυπο της κλάσης ServiceHandler και καλέσαμε τη μέθοδο makeServiceCall(). Εάν το αποτέλεσμα που επέστρεψε η makeServiceCall() δεν είναι κενό τότε ακολουθεί η αποδόμηση του JSON μέσα σε try-catch. Αρχικά δημιουργήσαμε ένα καινούργιο JSONObject, το jsonResponse, στο οποίο δώσαμε σαν όρισμα το String που επέστρεψε ο server. Επόμενο βήμα ήταν να δώσουμε στο JSONArray hotel το όνομα που έχει το JSONArray όπως αυτό φαίνεται στον browser. Έχοντας στη διάθεσή μας το όνομα του κεντρικού κόμβου του JSON Array και μέσω της επανάληψης for αποθηκεύουμε σε global μεταβλητές τις τιμές που περιέχει κάθε JSON Object. Τέλος στη μέθοδο onPostExecute() καταργούμε το ProgressDialog και αποδίδουμε τις τιμές που επέστρεψε η βάση στα αντίστοιχα TextView. Ανάλογα με την κατηγορία του ξενοδοχείου εμφανίζεται και η αντίστοιχη εικόνα των αστεριών.

```
class loadHotelDetails extends AsyncTask<String, String, String> {
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            pDialog = new ProgressDialog(ElGrecoHotel.this);
            pDialog.setMessage("Please wait..");
            pDialog.setCancelable(false);
            pDialog.show();
        }
        @Override

        protected String doInBackground(String... args) {

            ServiceHandler sh = new ServiceHandler();
            String jsonStr = sh.makeServiceCall(url, ServiceHandler.GET);
            Log.d("Response: ", "> " + jsonStr);

            if (jsonStr != null) {
                try {
                    hotel=null;
                    JSONObject jsonResponse = new JSONObject(jsonStr);
                    hotel = jsonResponse.optJSONArray(TAG_HOTEL);
```

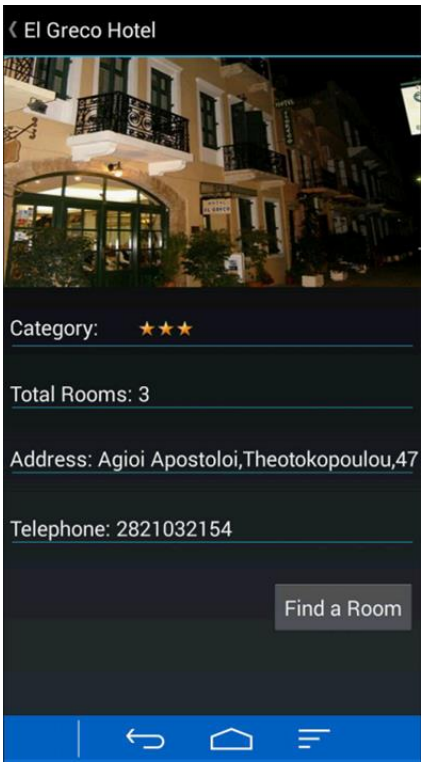
```
                    for (int i = 0; i < hotel.length(); i++) {
                        JSONObject c = hotel.getJSONObject(i);

                         category = c.optString(TAG_CATEGORY);
                         totalRooms = c.optString(TAG_TOTAL_ROOMS);
                         area = c.optString(TAG_AREA);
                         street = c.optString(TAG_STREET);
                         number = c.optString(TAG_NUMBER);
                         telephone = c.optString(TAG_TELEPHONE);

                    }

                } catch (JSONException e) {
                    e.printStackTrace();
                }
            } else {
                Log.e("ServiceHandler", "Couldn't get any data from the
url");
            }

            return null;
        }

        protected void onPostExecute(String file_url) {
            super.onPostExecute(file_url);
            if (pDialog.isShowing())
                pDialog.dismiss();
            textView3.setText(totalRooms);
            textView5.setText(area + "," + street + "," + number);
            textView7.setText(telephone);

            switch (Integer.parseInt(category)) {
                case 2:
                    imageRate.setImageResource(R.drawable.rate2);
                    break;
                case 3:
                    imageRate.setImageResource(R.drawable.rate3);
                    break;
                case 4:
                    imageRate.setImageResource(R.drawable.rate4);
                    break;
                case 5:
                    imageRate.setImageResource(R.drawable.rate5);
                    break;
            }
        }
    }
```

Το PHP αρχείο που περιέχει το SQL query για την ανάκτηση των πληροφοριών από τη βάση δεδομένων ενός ξενοδοχείου είναι το getHotelDetails.php και ο κώδικάς του φαίνεται παρακάτω. Μέσα σε σχόλια γίνεται επεξήγηση του κώδικα.

```
<?php
```

```
//Διαδικασία σύνδεσης με τη βάση δεδομένων.

$host='localhost';
$uname='root';
$pwd='';
$db='mydatabase';
$con = mysql_connect($host,$uname,$pwd) or die("connection failed");
      mysql_select_db($db,$con) or die("db selection failed");
//Παράκληση της μεταβλητής "p" η οποία θα δοθεί σαν όρισμα στο url και αποθήκευση της τιμής στη μεταβλητή $tag.
      if (isset($_REQUEST['p']) && $_REQUEST['p'] != '') {
    $tag = $_REQUEST['p'];
      }
//Ερώτημα SQL στο οποίο διαλέγουμε τα δεδομένα που θέλουμε προς ανάκτηση από τον πίνακα hotels, σε συνένωση με τον πίνακα telephonehotel. Προϋπόθεση το hotelID που βρίσκεται στους πίνακες hotels και telephonehotel να ισούται με $tag.
      $r=mysql_query("SELECT
hotels.Category,hotels.TotalRooms,hotels.Area,hotels.Street,hotels.Number,
telephonehotel.TelephoneHotel
      FROM hotels
      LEFT JOIN telephonehotel
      ON hotels.HotelID=telephonehotel.HotelID
      WHERE hotels.HotelID=$tag
      ",$con);
    $cls = array();
//Αποθήκευση των δεδομένων που επέστρεψε το SQL query στον πίνακα cls.
while($row=mysql_fetch_assoc($r))
      {
            $cls['hotels'][]=$row;

      }
//Προβολή των δεδομένων στον browser σε μορφή JSON.
      print(json_encode($cls));
      mysql_close($con);
?>
```

Η διεπαφή χρήστη του ElGrecoHotel Activity φαίνεται παρακάτω.



Εικόνα 36: Διεπαφή χρήστη ElGrecoHotel

Πατώντας το κουμπί "Find a Room" εκκινείτε το Activity FindRoomFromParameters.java το οποίο κάνει «extends» το ActionBarActivity και «implements» το Interface AdapterView.OnItemSelectedListener, ώστε να μπορέσουμε να υλοποιήσουμε τις μεθόδους για να συμπεριλάβουμε τα drop-down menu, τα λεγόμενα spinners.

```
public class FindRoomFromParameters extends ActionBarActivity implements
AdapterView.OnItemSelectedListener
```

Η λειτουργία του Activity είναι η επιλογή της επιθυμητής ημερομηνίας από τον χρήστη, δίνοντας του τη δυνατότητα να επιλέξει την ημερομηνία κράτησης του δωματίου (check-in, check-out). Ο χρήστης μπορεί να επιλέξει μία ημερομηνία είτε μέσα από την επιλογή των spinners είτε πατώντας το εικονίδιο με το ημερολόγιο. Για να μπορέσουμε να ενσωματώσουμε ένα spinner στο Activity, πρέπει να προσθέσουμε στο αντίστοιχο xml (hotel_room_parameters.xml) που περιγράφει τη διεπαφή χρήστη, τον ακόλουθο κώδικα.

```
<Spinner
android:id="@+id/spinner1"
android:layout_width="85dp"
android:layout_height="wrap_content"
android:layout_below="@+id/textView1"
android:layout_toRightOf="@+id/textView1"
        />
```

Για να μπορέσουμε να αξιοποιήσουμε ένα spinner, στον κώδικά μας, πρέπει να ενσωματώσουμε έναν ArrayAdapter στον οποίο θα δώσουμε σαν όρισμα έναν πίνακα ο οποίος θα περιέχει τις επιθυμητές επιλογές που θέλουμε να έχει το spinner. Στη συνέχεια αφού έχουμε αντιστοιχήσει τα components που έχουμε δηλώσει μέσα στο Activity με αυτά του xml, πρέπει να ορίσουμε στο spinner τον Adapter. Το Activity μας περιέχει τέσσερα spinners εκ των οποίων το πρώτο αναφέρεται για την ημέρα κράτησης, το δεύτερο για το μήνα κράτησης, το τρίτο για την ημέρα

αναχώρησης και το τέταρτο για το μήνα αναχώρησης. Οι παραπάνω λειτουργίες γίνονται μέσα στη μέθοδο onCreate().

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.hotel_room_parameters);
        intent3=getIntent();
        hotelID=intent3.getStringExtra("hotelID"); //Μέθοδος που παίρνει
τα δεδομένα από το προηγούμενο Activity και αποθήκευση της τιμής στη
μεταβλητή hotelID.
        checkinCalendar =(ImageButton)findViewById(R.id.imageButton1);
        checkoutCalendar =(ImageButton)findViewById(R.id.imageButton2);
        checkinDaySpinner = (Spinner) findViewById(R.id.spinner1);
        checkinDaySpinner.setOnItemSelectedListener(this);
        checkinMonthSpinner = (Spinner) findViewById(R.id.spinner2);
        checkinMonthSpinner.setOnItemSelectedListener(this);
        checkoutDaySpinner = (Spinner) findViewById(R.id.spinner3);
        checkoutDaySpinner.setOnItemSelectedListener(this);
        checkoutMonthSpinner = (Spinner) findViewById(R.id.spinner4);
        checkoutMonthSpinner.setOnItemSelectedListener(this);
        dayAdapter = ArrayAdapter.createFromResource(this, R.array.days,
android.R.layout.simple_spinner_item);
dayAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdow
n_item);
        checkinDaySpinner.setAdapter(dayAdapter);
        checkoutDaySpinner.setAdapter(dayAdapter);
        monthAdapter = ArrayAdapter.createFromResource(this,
R.array.months, android.R.layout.simple_spinner_item);
monthAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropd
own_item);
        checkinMonthSpinner.setAdapter(monthAdapter);
        checkoutMonthSpinner.setAdapter(monthAdapter);
}
```

Οι πίνακες με τις ημέρες ενός μήνα και οι μήνες βρίσκονται στο αρχείο strings.xml. Όταν ο χρήστης επιλέξει τις ημερομηνίες από τα spinners, υλοποιείται η μέθοδος onItemSelected(), η οποία παίρνει την επιλογή που έχει το spinner και την αποθηκεύει σε global μεταβλητές, ώστε να μπορέσουμε να συνθέσουμε την ημερομηνία για check-in Date και check-out Date.

```
@Override
    public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {
     switch (parent.getId()){
         case R.id.spinner1:

finalCheckinDay=checkinDaySpinner.getSelectedItem().toString();
              day1= checkinDaySpinner.getPositionForView(view)+1;
              break;
         case R.id.spinner2:

finalCheckinMonth=checkinMonthSpinner.getSelectedItem().toString();
              month1= checkinMonthSpinner.getPositionForView(view)+1;
              break;
```

```
         case R.id.spinner3:

finalCheckoutDay=checkoutDaySpinner.getSelectedItem().toString();
             day2= checkoutDaySpinner.getPositionForView(view)+1;
             checkinDate=year+"-"+convertNumber(month1)+"-
"+convertNumber(day1);
             checkoutDate=year+"-"+convertNumber(month2)+"-
"+convertNumber(day2);
             break;
         case R.id.spinner4:

finalCheckoutMonth=checkoutMonthSpinner.getSelectedItem().toString();
             month2= checkoutMonthSpinner.getPositionForView(view)+1;
             checkinDate=year+"-"+convertNumber(month1)+"-
"+convertNumber(day1);
             checkoutDate=year+"-"+convertNumber(month2)+"-
"+convertNumber(day2);
             break;
     }
}
```

Ο δεύτερος τρόπος επιλογής ημερομηνίας είναι το εικονίδιο με το ημερολόγιο. Όταν ο χρήστης πατήσει το εικονίδιο εκκινείτε ένα καινούργιο Activity που ονομάζεται Calendar.java. Η κλάση Calendar παρέχεται από τις βιβλιοθήκες του Android και τη συμπεριλάβαμε κάνοντας «import java.util.Calendar;». Το Android μας παρέχει τη μέθοδο startActivityForResult(), η οποία επιστρέφει το αποτέλεσμα ενός Activity που τελείωσε στο Activity που έκανε κλήση της μεθόδου. Για να ξεκινήσει το Activity Calendar δημιουργήσαμε ένα στιγμιότυπο Intent και το περάσαμε σαν όρισμα στη μέθοδο startActivityForResult() μαζί με έναν ακέραιο αριθμό.

```
        checkinCalendar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                intent1 = new Intent(getApplicationContext(),
com.example.mixalis.final_project.support.Calendar.class);
                requestCode1 = 1;
                startActivityForResult(intent1, requestCode1);
            }
        });
```

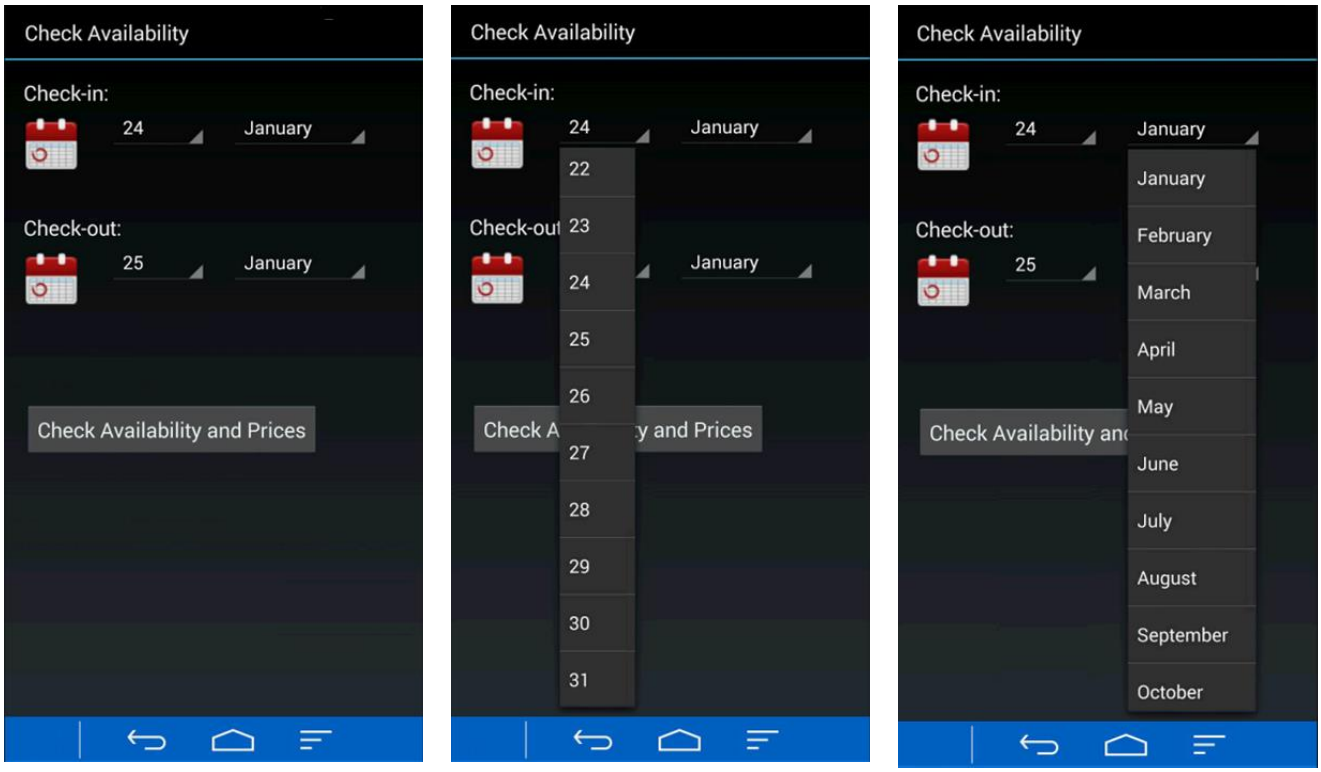
Όταν εκτελείται το Activity Calendar οι ημερομηνίες που επέλεξε ο χρήστης αποθηκεύονται σε global μεταβλητές και πατώντας το κουμπί "check" το Activity επιστρέφει τις ημερομηνίες στο Activity FindRoomFromParameters, μαζί με το συνθηματικό RESULT_OK και έπειτα τερματίζει.

```
       imageButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent returnIntent = new Intent();
                returnIntent.putExtra("DAY",selectedDay);
                returnIntent.putExtra("MONTH",selectedMonth);
                setResult(RESULT_OK, returnIntent);
                finish();
            }
        });
```

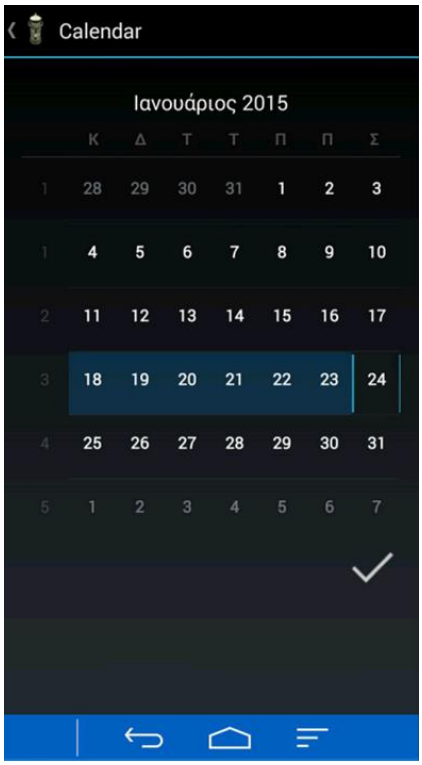
Επιστρέφοντας στο FindRoomFromParameters.java εκτελείται η μέθοδος onActivityResult() για να λάβει την ημερομηνία και μέσω της μεθόδου getExtra() αποθηκεύεται σε global μεταβλητές. Ο ρόλος του ακεραίου requestCode είναι η διαφοροποίηση μεταξύ των εικονιδίων του ημερολογίου ώστε να γνωρίζει αν πρόκειται για check-in Date ή check-out Date.

```
    @Override
    protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
        if (requestCode == 1) {
            if(resultCode == RESULT_OK){
                checkinDay = data.getIntExtra("DAY", checkinDay);
                checkinMonth = data.getIntExtra("MONTH", checkinMonth);
checkinDaySpinner.setSelection(dayAdapter.getPosition(Integer.toString(che
ckinDay)));
checkinMonthSpinner.setSelection(dayAdapter.getPosition(Integer.toString(c
heckinMonth)));
            }
        }
        else if(requestCode == 2){
            if(resultCode == RESULT_OK){
                checkoutDay = data.getIntExtra("DAY", checkoutDay);
                checkoutMonth = data.getIntExtra("MONTH", checkoutMonth);

checkoutDaySpinner.setSelection(dayAdapter.getPosition(Integer.toString(ch
eckoutDay)));
checkoutMonthSpinner.setSelection(dayAdapter.getPosition(Integer.toString(
checkoutMonth)));
            }
        }
    }
```

Η διεπαφή χρήστη του Activity FindRoomFromParameters καθώς και του Calendar φαίνονται παρακάτω.



Εικόνα 37: Διεπαφές χρήστη κατά την επιλογή ημερομηνίας



Εικόνα 38: Διεπαφή χρήστη Calendar

Τέλος πατώντας το κουμπί "Check Availability and Prices" εκκινείτε ένα καινούργιο Activity με τα διαθέσιμα δωμάτια στο οποίο στέλνουμε το hotelID καθώς και τις ημερομηνίες που επέλεξε ο χρήστης.

```
        checkAvailability.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(v.getContext(),
AvailableRoomList.class);
                intent.putExtra("hotelID",hotelID);
                intent.putExtra("checkinDate",checkinDate);
                intent.putExtra("checkoutDate",checkoutDate);
                intent.putExtra("checkinDay",day1);
                intent.putExtra("checkinMonth",month1);
                intent.putExtra("checkoutDay",day2);
                intent.putExtra("checkoutMonth",month2);
                intent.putExtra("currentDay",currentDay);
                intent.putExtra("currentMonth",currentMonth);
                startActivity(intent);
            }
        });
```

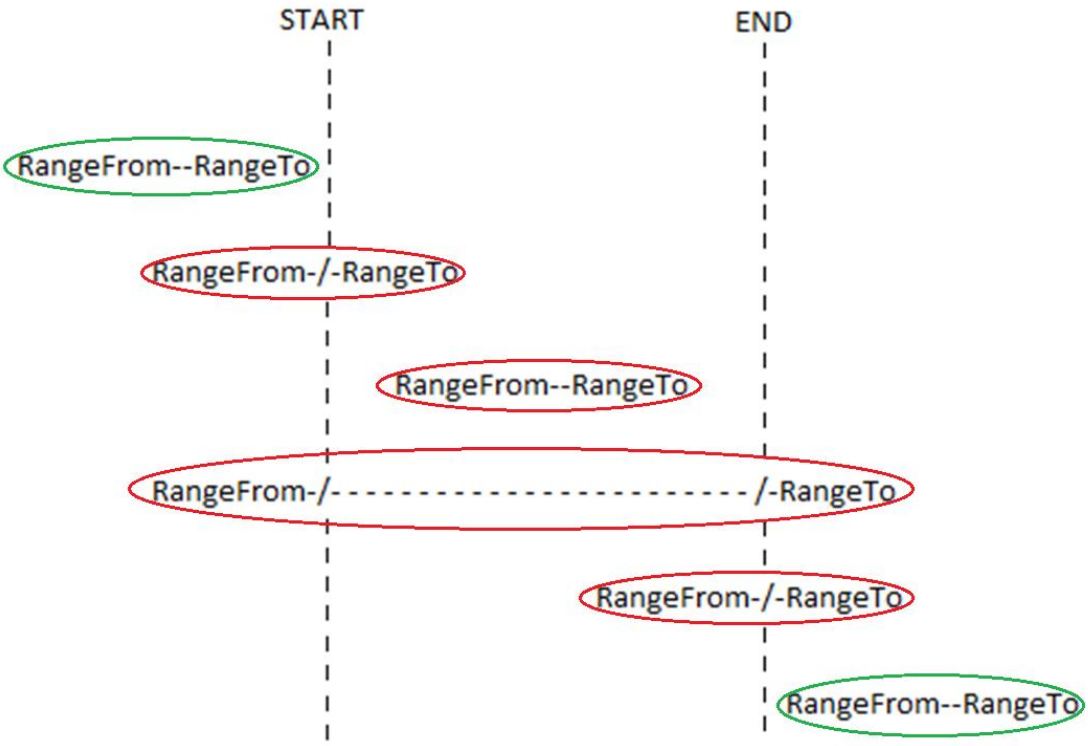
Το Activity που είναι υπεύθυνο για την εμφάνιση των διαθέσιμων δωματίων ονομάζεται AvailableRoomList.java και κάνει «extends» το ListActivity. Το xml αρχείο που περιγράφει τη διεπαφή χρήστη του Activity ονομάζεται hotel_room_list.xml και περιέχει ένα ListView.

```
<ListView
 android:id="@android:id/list"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_alignParentTop="true"
 android:layout_alignParentLeft="true"
 android:layout_alignParentStart="true"/>
```

Μέσα στην onCreate() φροντίσαμε να αποθηκεύσουμε τα δεδομένα που έστειλε το προηγούμενο Activity σε global μεταβλητές, μέσω των μεθόδων που προσφέρει ένα Intent. Για τη διαδικασία εύρεσης των διαθέσιμων δωματίων, καλούνται μέσα στην onCreate(), τέσσερις κλάσεις που χρησιμοποιούν τη μεθοδολογία του AsyncTask.

```
new FindNotAvailableRooms().execute();
new Control().execute();
new GetAvailableRooms().execute();
new UndoControl().execute();
```

Η πρώτη κλάση FindNotAvailbleRooms.java κάνει ένα αίτημα GET στη βάση δεδομένων και επιστρέφει το roomID των μη διαθέσιμων δωματίων. Το PHP αρχείο το οποίο περιέχει το SQL query ονομάζεται getNotAvailableRooms.php και παίρνει σαν παραμέτρους στο URL τις ημερομηνίες check-in και check-out. Για να μπορέσουμε να κάνουμε έλεγχο μεταξύ ημερομηνιών έπρεπε πρώτα να μετατρέψουμε τις ημερομηνίες σε κατάλληλο format πριν τις δώσουμε σαν όρισμα στο URL. Το format που ακολουθήσαμε είναι τύπου έτος-μήνας-ημέρα. Πριν δούμε τον κώδικα, παρουσιάζεται ένα σχήμα το οποίο μας δείχνει τους ελέγχους που πρέπει να ακολουθήσουμε για να βρούμε αν ένα δωμάτιο είναι διαθέσιμο ή όχι. Τα Start και End αντιστοιχούν σε μία ημερομηνία κράτησης ενός δωματίου και το RangeFrom—RangeTo, σε μία καινούργια ημερομηνία που θέλουμε να κάνουμε κράτηση σε αυτό το δωμάτιο. Μέσα σε πράσινο κύκλο βρίσκονται οι ημερομηνίες που μπορεί να γίνει κράτηση ενώ σε κόκκινο κύκλο βρίσκονται οι ημερομηνίες που συγκρούονται με μία ήδη υπάρχουσα κράτηση για αυτό το δωμάτιο.



Εικόνα 39: Βοηθητικό διάγραμμα ελέγχου ημερομηνιών

```
    private class FindNotAvailableRooms extends AsyncTask<Void, Void,
Void> {

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }

        @Override
        protected Void doInBackground(Void... params) {

            ServiceHandler sh = new ServiceHandler();

            sdf = new SimpleDateFormat("yyyy-MM-dd");
            firstDate = null;
            secondDate = null;

            try {
                firstDate = sdf.parse(checkinDate);
                secondDate = sdf.parse(checkoutDate);
            } catch (ParseException e) {
                e.printStackTrace();
            }
            url=
"http://192.168.1.3/android_api/getNotAvailableRooms.php"+"?"+"p="+hotelID
+"&"+"checkin=%27"+sdf.format(firstDate)+"%27"+"&"+"checkout=%27"+sdf.form
at(secondDate)+"%27";
```

```
            String jsonStr = sh.makeServiceCall(url, ServiceHandler.GET);
            Log.d("Response: ", "> " + jsonStr);
            if (jsonStr != null) {
                try {
                    JSONObject jsonObj = new JSONObject(jsonStr);

                    notAvailableRooms =
jsonObj.getJSONArray(TAG_NOT_AVAILABLE_ROOMS);

                     for (int i = 0; i < notAvailableRooms.length(); i++)
{
                        JSONObject c = notAvailableRooms.getJSONObject(i);
                        roomId = c.getString(TAG_ROOMID);

                         notAvailableRoomList.add(roomId);
                    }
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            } else {
                Log.e("ServiceHandler", "Couldn't get any data from the
url");
            }

            return null;
        }

        @Override
        protected void onPostExecute(Void result) {
            super.onPostExecute(result);

        }

    }
```

Το SQL query που επιστρέφει το roomID των μη διαθέσιμων δωματίων είναι το παρακάτω.

```
if (isset($_REQUEST['p']) && $_REQUEST['p'] != '') {
    $tag = $_REQUEST['p'];
      $date1=$_REQUEST['checkin'];
      $date2=$_REQUEST['checkout'];
      }
      $r=mysql_query("SELECT reserveroom.RoomID FROM reserveroom WHERE
reserveroom.HotelID=$tag AND
      ((reserveroom.Checkin BETWEEN $date1 AND $date2) OR
(reserveroom.Checkout BETWEEN $date1 AND
$date2)OR(reserveroom.Checkin<$date1 AND
reserveroom.Checkout>$date2))ORDER BY RoomID",$con);
```

Έχοντας πλέον στη διάθεσή μας αποθηκευμένα όλα τα roomID των μη διαθέσιμων δωματίων σε ένα ArrayList, μέσα στη κλάση Control κάνουμε ένα αίτημα POST στη βάση δεδομένων ώστε να αλλάξει το πεδίο control στον πίνακα reserveroom από 1 σε 0. Μέσω της επανάληψης for παίρνουμε ένα-ένα τα στοιχεία που βρίσκονται στο ArrayList με τα μη διαθέσιμα δωμάτια και τα δίνουμε σαν όρισμα στο URL.

```
    private class Control extends AsyncTask<Void, Void, Void> {
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }
        @Override
        protected Void doInBackground(Void... params) {

            String roomID;
            ServiceHandler sh = new ServiceHandler();

            for(int i=0;i<notAvailableRoomList.size();i++){
                roomID=notAvailableRoomList.get(i);
                url=
"http://192.168.1.3/android_api/setControlToZeroAtReserveRoom.php"+"?"+"p=
"+hotelID+"&"+"a="+roomID;
                String jsonStr2 = sh.makeServiceCall(url,
ServiceHandler.POST);
            }
            return null;
        }
        @Override
        protected void onPostExecute(Void result) {
            super.onPostExecute(result);

        }

    }
```

Το SQL query που περιέχεται στο setControlToZeroAtReserveRoom.php φαίνεται παρακάτω.

```
if (isset($_REQUEST['p']) && $_REQUEST['p'] != '') {
    $tag = $_REQUEST['p'];
      $roomid = $_REQUEST['a'];
      }
      $r=mysql_query("UPDATE reserveroom
      SET Control='0'
      WHERE HotelID=$tag AND RoomID=$roomid ",$con);
```

Η επόμενη κλάση που εκτελείται ονομάζεται GetAvailableRooms και είναι υπεύθυνη για την επιστροφή των έγκυρων δωματίων, μαζί με τις πληροφορίες κάθε δωματίου. Το PHP αρχείο που επιστρέφει τα διαθέσιμα δωμάτια ονομάζεται getAvailableRoomsByHotel.php και το SQL query που καλείται είναι το ακόλουθο.

```
if (isset($_REQUEST['p']) && $_REQUEST['p'] != '') {
        $tag = $_REQUEST['p'];
      $date1=$_REQUEST['checkin'];
      $date2=$_REQUEST['checkout'];
      }
      $r=mysql_query("SELECT rooms.RoomID,rooms.Type,rooms.Price FROM rooms
WHERE rooms.HotelID=$tag AND rooms.Availability='1'
      UNION ALL
      (SELECT DISTINCT reserveroom.RoomID,rooms.Type,rooms.Price
```

```
      FROM reserveroom
      LEFT JOIN rooms
      ON reserveroom.RoomID=rooms.RoomID
      WHERE reserveroom.HotelID=$tag AND (reserveroom.Control='1') AND
      ((((reserveroom.Checkin NOT BETWEEN $date1 AND $date2) AND
(reserveroom.Checkout NOT BETWEEN $date1 AND $date2)) AND
(reserveroom.Checkout<=$date1 OR reserveroom.Checkin>=$date2))))ORDER BY
RoomID",$con);
```

Για να κριθεί ένα δωμάτιο ως διαθέσιμο, το παραπάνω query θα ψάξει πρώτα στον πίνακα rooms για όσα δωμάτια έχουν Availability 1 και στη συνέχεια στον πίνακα reserveroom, για όσα δωμάτια έχουν Control 1 και δεν συγκρούονται οι ημερομηνίες. Για κάθε δωμάτιο που θα βρει επιστρέφει το roomId, τον τύπο του δωματίου και την τιμή. Αναγκαία προϋπόθεση ήταν να δώσουμε στο URL σαν ορίσματα τις ημερομηνίες check-in και check-out. Στη συνέχεια αποθηκεύονται τα δεδομένα κάθε δωματίου που επέστρεψε η βάση σε global μεταβλητές και με τη χρήση του HashMap συνθέσαμε κάθε γραμμή της λίστας, η οποία περιέχει το roomId, τον τύπο του δωματίου και την τιμή. Έπειτα στη μέθοδο onPostExecute() με τη χρήση ενός SimpleAdapter φροντίσαμε να είναι ορατή η λίστα.

```
    private class GetAvailableRooms extends AsyncTask<Void, Void, Void> {
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            pDialog = new ProgressDialog(AvailableRoomList.this);
            pDialog.setMessage("Please wait...");
            pDialog.setCancelable(false);
            pDialog.show();
        }
        @Override
        protected Void doInBackground(Void... params) {
            ServiceHandler sh = new ServiceHandler();
             sdf = new SimpleDateFormat("yyyy-MM-dd");
             firstDate = null;
             secondDate = null;
            try {
                firstDate = sdf.parse(checkinDate);
                secondDate = sdf.parse(checkoutDate);
            } catch (ParseException e) {
                e.printStackTrace();
            }
            url=
"http://192.168.1.3/android_api/getAvailableRoomsByHotel.php"+"?"+"p="+hot
elID+"&"+"checkin=%27"+sdf.format(firstDate)+"%27"+"&"+"checkout=%27"+sdf.
format(secondDate)+"%27";
            String jsonStr = sh.makeServiceCall(url, ServiceHandler.GET);
            Log.d("Response: ", "> " + jsonStr);
            if (jsonStr != null) {
                try {
                     JSONObject jsonObj = new JSONObject(jsonStr);
                     rooms = jsonObj.getJSONArray(TAG_ROOMS);
                      for (int i = 0; i < rooms.length(); i++) {
                        JSONObject c = rooms.getJSONObject(i);
                               roomId = c.getString(TAG_ROOMID);
                               type = c.getString(TAG_TYPE);
```

```
                        String price = c.getString(TAG_PRICE)+"€";
                        String pricePerDay = c.getString(TAG_PRICE);
                         totalPrice=totalDays*
Integer.parseInt(pricePerDay);
                        HashMap<String, String> eachRoom = new
HashMap<String, String>();
                        eachRoom.put(TAG_ROOMID, roomId);
                        eachRoom.put(TAG_TYPE, type);
                        eachRoom.put(TAG_PRICE, price);

eachRoom.put("TotalPrice",String.valueOf(totalPrice));
                        roomList.add(eachRoom);
                    }
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            } else {
                Log.e("ServiceHandler", "Couldn't get any data from the
url");
            }
            return null;
        }
        @Override
        protected void onPostExecute(Void result) {
            super.onPostExecute(result);
            if (pDialog.isShowing())
                pDialog.dismiss();
            ListAdapter adapter = new SimpleAdapter(
AvailableRoomList.this, roomList, R.layout.room_item, new String[] {
TAG_ROOMID,TAG_TYPE, TAG_PRICE ,"TotalPrice"}, new int[] {
R.id.roomID,R.id.type,R.id.price,R.id.totalPrice});
            setListAdapter(adapter);
        }
}
```

Αφού έχουμε στη διάθεσή μας τα διαθέσιμα δωμάτια φροντίζουμε να επαναφέρουμε το πεδίο Control στον πίνακα reserveroom στην αρχική του τιμή 1 ώστε να μην υπάρξει πρόβλημα σε μελλοντική εύρεση των διαθέσιμων δωματίων.

```
    private class UndoControl extends AsyncTask<Void, Void, Void> {
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }
        @Override
        protected Void doInBackground(Void... params) {
            ServiceHandler sh = new ServiceHandler();
                url=
"http://192.168.1.3/android_api/setControlToDefaultAtReserveRoom.php"+"?"+
"p="+hotelID;
                String jsonStr2 = sh.makeServiceCall(url,
ServiceHandler.POST);
            return null;
        }
```

```
        @Override
        protected void onPostExecute(Void result) {
            super.onPostExecute(result);
        }
    }
```

Το SQL query του παραπάνω PHP αρχείου είναι το ακόλουθο.

```
  if (isset($_REQUEST['p']) && $_REQUEST['p'] != '') {
$tag = $_REQUEST['p'];
  }
  $r=mysql_query("UPDATE reserveroom
  SET Control='1'
  WHERE HotelID=$tag",$con);
```

Έχοντας πλέον τελειώσει τη διαδικασία εύρεσης των διαθέσιμων δωματίων, έπρεπε να εισάγουμε κάποια αλληλεπίδραση μεταξύ δωματίου και χρήστη. Για το λόγο αυτό προσθέσαμε ένα Listener σε κάθε στοιχείο της λίστας και όταν ο χρήστης πατήσει σε κάποιο δωμάτιο, ανοίγει ένα DialogFragment. Στο Activity DialogFragment με την ονομασία RoomDialog.java, περάσαμε τα δεδομένα που αφορούν ένα δωμάτιο μέσω ενός στιγμιότυπου Bundle. Ο κώδικας του Listener βρίσκεται μέσα στην onCreate() της κλάσης AvailableRoomList.java και φαίνεται παρακάτω.

```
ListView lv = getListView();
         lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            public void onItemClick(AdapterView parent, View view, int
position, long id) {
                String pid = ((TextView)
view.findViewById(R.id.roomID)).getText().toString();
                String totalPrice = ((TextView)
view.findViewById(R.id.totalPrice)).getText().toString();
                Bundle args=new Bundle();
                args.putString("hotelID",hotelID);
                args.putString("roomID",pid);
                args.putString("type",type);
                args.putString("totalPrice",totalPrice);
                args.putString("checkin",sdf.format(firstDate));
                args.putString("checkout",sdf.format(secondDate));
                RoomDialog roomDialog=new RoomDialog();
                roomDialog.setArguments(args);
                roomDialog.show(getFragmentManager(),"My Dialog");
                }
        });
```

Κατά την εκτέλεση του RoomDialog.java, μέσα στη μέθοδο onCreateDialog(), αποθηκεύουμε τα δεδομένα που έστειλε το προηγούμενο Activity σε global μεταβλητές. Στη συνέχεια υλοποιήσαμε τις μεθόδους που αφορούν τα κουμπιά που περιέχει το RoomDialog. Όταν ο χρήστης πατήσει το κουμπί "Add" εκτελούνται δύο κλάσεις που κάνουν «extends» AsyncTask, ενώ όταν πατήσει "Cancel" κλείνει το παράθυρο διαλόγου.

```
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        AlertDialog.Builder alertDialog = new
AlertDialog.Builder(getActivity());
```

```
        alertDialog.setTitle("Book a Room");
        alertDialog.setMessage("Add this room to your cart?");

        hotelID=getArguments().getString("hotelID");
        roomId=getArguments().getString("roomID");
        checkinDate=getArguments().getString("checkin");
        checkoutDate=getArguments().getString("checkout");
        totalPrice=getArguments().getString("totalPrice");
        alertDialog.setPositiveButton("Add", new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog,int which) {
                new sendToReserveroom().execute();
                new setRoomAvailability().execute();
            }
        });
        alertDialog.setNegativeButton("Cancel", new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                                dialog.cancel();
            }
        });
        return alertDialog.show();

    }
```

Η πρώτη κλάση που καλείται όταν πατηθεί η επιλογή "Add", ονομάζεται sendToReserveroom η οποία στέλνει ένα αίτημα POST στη βάση δεδομένων, για να εισάγει στον πίνακα reserveroom μία εγγραφή.

```
    private class sendToReserveroom extends AsyncTask<Void, Void, Void> {
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }
        @Override
        protected Void doInBackground(Void... params) {
            ServiceHandler sh = new ServiceHandler();
            sdf = new SimpleDateFormat("yyyy-MM-dd");
            firstDate = null;
            secondDate = null;
            try {
                firstDate = sdf.parse(checkinDate);
                secondDate = sdf.parse(checkoutDate);
            } catch (ParseException e) {
                e.printStackTrace();
            }
            url1=
"http://192.168.1.3/android_api/setBookingAtReserveRoom.php"+"?"+"p="+hote
lID+"&"+"checkin=%27"+sdf.format(firstDate)+"%27"+"&"+"checkout=%27"+sdf.f
ormat(secondDate)+"%27"+"&"+"a="+roomId+"&price="+totalPrice;
            String jsonStr = sh.makeServiceCall(url1,
ServiceHandler.POST);
            return null;
        }
```

```
        @Override
        protected void onPostExecute(Void result) {
            super.onPostExecute(result);
        }
    }
```

Το SQL query που αναλαμβάνει αυτή τη δουλειά βρίσκεται στο setBookingAtReserveRoom.php και φαίνεται παρακάτω.

```
if (isset($_REQUEST['p']) && $_REQUEST['p'] != '') {
    $tag = $_REQUEST['p'];
      $date1=$_REQUEST['checkin'];
      $date2=$_REQUEST['checkout'];
      $roomid= $_REQUEST['a'];
      $price= $_REQUEST['price'];
      }
      $r=mysql_query("INSERT INTO
reserveroom(RoomID,HotelID,CustomerID,Checkin,Checkout,Control,TotalPrice)
      VALUES($roomid,$tag,'1',$date1,$date2,'1',$price)",$con);
```

Η δεύτερη κλάση που καλείται ονομάζεται setRoomAvailability η οποία αλλάζει το πεδίο Availability στον πίνακα rooms από 1 σε 0 για το δωμάτιο που έγινε κράτηση μέσω ενός POST αιτήματος.

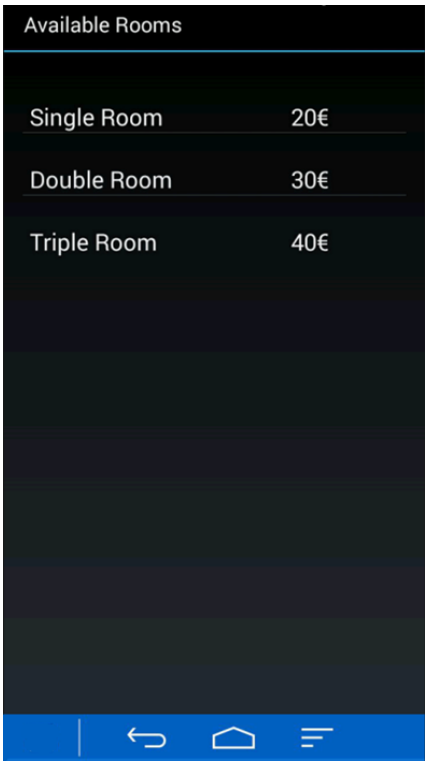
```
    private class setRoomAvailability extends AsyncTask<Void, Void, Void>
{

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }
        @Override
        protected Void doInBackground(Void... params) {
            ServiceHandler sh = new ServiceHandler();

url2="http://192.168.1.3/android_api/setRoomAvailabilty.php"+"?"+"p="+hote
lID+"&"+"a="+roomId;
            String jsonStr2 = sh.makeServiceCall(url2,
ServiceHandler.POST);
            return null;
        }

        @Override
        protected void onPostExecute(Void result) {
            super.onPostExecute(result);
        }
    }
```

Ακολουθεί το SQL query του setRoomAvailabilty.php.

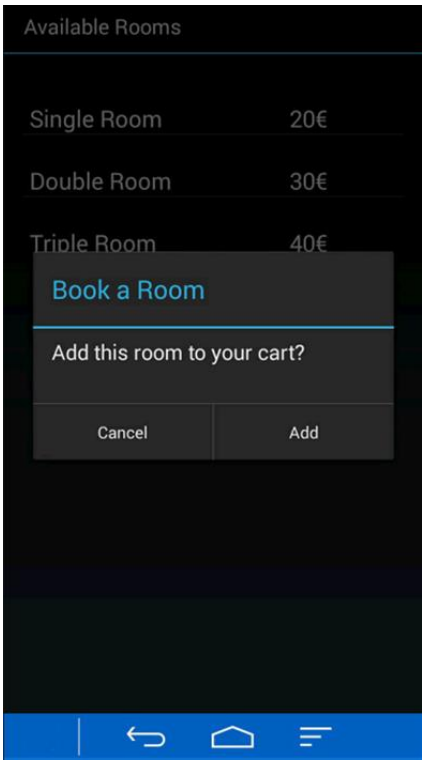
```
if (isset($_REQUEST['p']) && $_REQUEST['p'] != '') {
    $tag = $_REQUEST['p'];
      $roomid = $_REQUEST['a'];
      }
```

```
$r=mysql_query("UPDATE rooms
SET Availability='0'
WHERE HotelID=$tag AND RoomID=$roomid ",$con);
```

Η διεπαφή χρήστη με τα διαθέσιμα δωμάτια καθώς και το Dialog message παρουσιάζονται παρακάτω.



Εικόνα 40: Διεπαφή χρήστη AvailableRoomList



Εικόνα 41: Διεπαφή χρήστη RoomDialog

4.3.4 Shopping Cart

Για να δώσουμε τη δυνατότητα στο χρήστη να μπορεί να βλέπει οτιδήποτε έχει κάνει κράτηση καθώς και τη δυνατότητα ακύρωσης μίας κράτησης στο εικονίδιο καλάθι αγορών υλοποιήσαμε το Activity ShoppingCart.java. Στόχος μας ήταν η εμφάνιση των στοιχείων αποτελούμενα από τις πληροφορίες τους, τοποθετημένα σε μία λίστα και η δυνατότητα διαγραφής της κράτησης.

Για να δούμε τι έχει κάνει κράτηση ένας χρήστης έπρεπε να δούμε τι εγγραφές υπάρχουν στους πίνακες reserveroom, reservevehicle, reservetrip και reserveroute. Το γεγονός ότι μία κράτηση περιέχει πολλές πληροφορίες (roomId, hotelId, check-in, check-out) καθώς και το γεγονός ότι έπρεπε να υλοποιήσουμε ένα κουμπί για ακύρωση μίας κράτησης, οδήγησαν στο συμπέρασμα ότι έπρεπε να δημιουργήσουμε ένα δικό μας Adapter για τη λίστα των κρατήσεων. Αρχικά δημιουργήσαμε μία νέα κλάση, την ShoppingCartRow.java, η οποία αντιπροσωπεύει κάθε γραμμή της λίστας και περιέχει δεκαπέντε στοιχεία τύπου String. Η κλάση του Adapter που δημιουργήσαμε ονομάζεται CustomAdapter.java και μέσα σε αυτήν δηλώσαμε μία λίστα, της οποίας το κάθε στοιχείο θα αποτελεί ένα στιγμιότυπο της κλάσης ShoppingCartRow. Ακόμα μέσα στην CustomAdapter.java υλοποιήσαμε τον Listener που θα έχει το κουμπί της διαγραφής με σκοπό όταν αυτό πατηθεί να εμφανίζεται ένα DialogFragment με την ονομασία DeleteDialog. Σε αυτό το DeleteDialog στέλνουμε κάποιες πληροφορίες, οι οποίες βοηθούν στη διαγραφή του στοιχείου της λίστας που επέλεξε ο χρήστης προς ακύρωση, από τους αντίστοιχους πίνακες reserve (reserveroom, reservevehicle, reservetrip, reserveroute). Ο τρόπος υλοποίησης ενός Custom Adapter παρέχεται από την Android και όποια τυχόν τροποποίηση, γίνεται στη μέθοδο getView(). Για περισσότερες πληροφορίες που αφορούν την κατασκευή ενός Custom Adapter, ανατρέξτε εδώ.
http://developer.android.com/reference/android/widget/ArrayAdapter.html ,

https://github.com/codepath/android_guides/wiki/Using-an-ArrayAdapter-with-ListView

```
public class CustomAdapter extends ArrayAdapter<ShoppingCartRow> {
    private final List<ShoppingCartRow> list;
    private final Activity context;
    public CustomAdapter(Activity context, List<ShoppingCartRow> list) {
        super(context, R.layout.shopping_cart_item , list);
        this.context = context;
        this.list = list;
    }
static class ViewHolder {
        protected ImageButton imageButton;
}
    @Override
    public View getView(final int position, View convertView, ViewGroup
parent) {
..
..
viewHolder.imageButton = (ImageButton)
view.findViewById(R.id.imageButton1);
            view.setTag(viewHolder);
            viewHolder.imageButton.setTag(list.get(position));
     viewHolder.imageButton.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {

                    Integer pos=(Integer)v.getTag();
Bundle args=new Bundle();
                    args.putString("item1",list.get(pos).item1);
                    args.putString("item2",list.get(pos).item2);
..
..
                    DialogFragment deleteDialog=new DeleteDialog();
                    deleteDialog.setArguments(args);
                    deleteDialog.show(context.getFragmentManager(),"");
                }
            });
}
```

Το Activity ShoppingCart.java είναι υπεύθυνο για να μας επιστρέψει όλες τις εγγραφές που υπάρχουν στους πίνακες reserve. Αρχικά δηλώσαμε ένα στιγμιότυπο του CustomAdapter και ένα ArrayList με την ονομασία "itemArray" κάθε στοιχείο του οποίου θα είναι ένα στιγμιότυπο της κλάσης ShoppingCartRow.java. Μέσα στην onCreate() καλούνται οι κλάσεις που είναι υπεύθυνες για την ανάκτηση των δεδομένων από τους πίνακες reserve.

```
public class ShoppingCart extends Activity {
..
ListView shoppingCartList;
CustomAdapter customAdapter;
   ArrayList<ShoppingCartRow> itemArray = new
ArrayList<ShoppingCartRow>();
..
protected void onCreate(Bundle savedInstanceState) {
..
```

```
new GetStuffFromReserveRoom().execute();
new GetStuffFromReserveVehicle().execute();
new GetStuffFromReserveTrip().execute();
new GetStuffFromReserveRouteOnlyFlights().execute();
new GetStuffFromReserveRouteOnlyFerries().execute();

}
..
    }
```

Τα PHP αρχεία που χρησιμοποιούνται στις παραπάνω κλάσεις για τα αιτήματα GET είναι:

- getStuffFromReserveRoom.php
- getStuffFromReserveVehicle.php
- getStuffFromReserveTrip.php
- getStuffFromReserveRouteOnlyFlights.php
- getStuffFromReserveRouteOnlyFerries.php

Θα αναφερθούμε συνοπτικά στην πρώτη κλάση GetStuffFromReserveRoom, ο κώδικας της οποίας είναι πανομοιότυπος και με των υπολοίπων κλάσεων. Η κλάση κάνει «extends» το AsyncTask και μέσα στη doInBackground() γίνεται το "parsing" των δεδομένων από τη βάση, με τον τρόπο που έχουμε αναλύσει στην αρχή του κεφαλαίου 4. Αφού έχουμε αποθηκεύσει τα δεδομένα σε μεταβλητές, προσθέτουμε στο itemArray ένα στιγμιότυπο της κλάσης ShoppingCartRow, στον constructor της οποίας περνάμε τα δεδομένα που λάβαμε από τη βάση.

```
   @Override
        protected Void doInBackground(Void... params) {
..
..
url1= "http://192.168.1.3/android_api/getStuffFromReserveRoom.php";
..
..
itemArray.add(new ShoppingCartRow(key,roomID,"Hotel name: ",name,"Room
type: ",type,"Checkin: ",checkin,"Checkout: ",checkout,"Price:
",roomTotalPrice," "," ","",""));
..
}
```

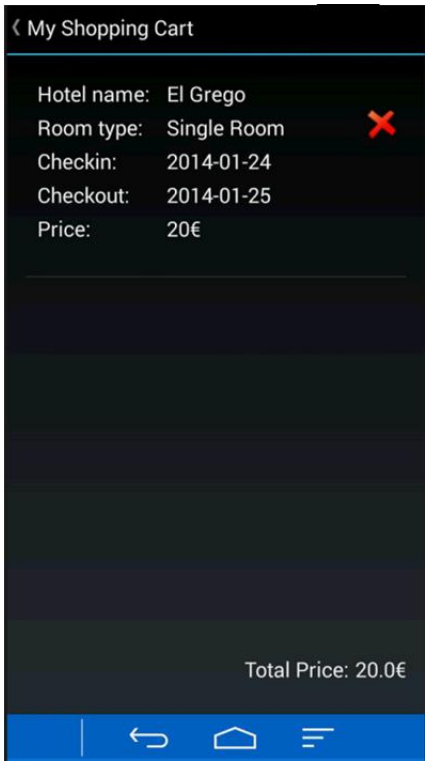
Τέλος στη μέθοδο onPostExecute() δημιουργούμε ένα στιγμιότυπο του CustomAdapter, στον Constructor του οποίου, δίνουμε σαν όρισμα το itemArray.

```
@Override
        protected void onPostExecute(Void result) {
..
..
customAdapter=new CustomAdapter(ShoppingCart.this,itemArray);
            shoppingCartList.setAdapter(customAdapter);

        }
```

Το SQL query που χρησιμοποιείται για την ανάκτηση των πληροφοριών των κλεισμένων δωματίων φαίνεται παρακάτω.

```
$r=mysql_query("SELECT DISTINCT
rooms.RoomID,hotels.Name,hotels.Key,rooms.Type,rooms.Price,reserveroom.Che
ckin,reserveroom.Checkout,reserveroom.TotalPrice
      FROM hotels
      LEFT JOIN rooms
      ON hotels.HotelID=rooms.HotelID
      LEFT JOIN reserveroom
      ON reserveroom.RoomID=rooms.RoomID
      WHERE reserveroom.CustomerID='1'",$con);
```

Η διεπαφή χρήστη του ShoppingCart εμφανίζεται παρακάτω.



Εικόνα 42: Διεπαφή χρήστη ShoppingCart

Όταν πατηθεί το κουμπί της διαγραφής ενός στοιχείου από τη λίστα του ShoppingCart, τότε εμφανίζεται ένα παράθυρο διαλόγου, το DeleteDialog. Μέσα στο Activity DeleteDialog.java και πιο συγκεκριμένα στη μέθοδο onCreateDialog(), αποθηκεύουμε τα δεδομένα που έστειλε ο CustomAdapter. Πλέον το Activity DeleteDialog.java έχει όλες τις απαραίτητες πληροφορίες για τη διαγραφή ενός στοιχείου (key, roomId, hotelName, roomType, check-in, check-out). Στη βάση δεδομένων έχουμε προσθέσει στους κύριους πίνακες (hotels, rentals, travelOffices, companies) το πεδίο key, ώστε να γνωρίζουμε εάν πρόκειται για διαγραφή κράτησης δωματίου, οχήματος κ.τ.λ.. Στον Listener που έχει το κουμπί "Remove" του DeleteDialog δημιουργήσαμε ένα switch statement με κριτήριο την τιμή που έχει το key. Έτσι εάν το key έχει την τιμή 1, πρόκειται για διαγραφή δωματίου. Πατώντας το κουμπί "Remove" εκτελείται η κλάση deleteFromReserveroom η οποία χρησιμοποιώντας τη μεθοδολογία του AsyncTask στέλνει ένα αίτημα POST στον πίνακα reserveroom. Στην κλάση deleteFromReserveroom και συγκεκριμένα στη doInBackground() χρησιμοποιούμε το PHP αρχείο deleteFromReserveroom.php, το οποίο διαγράφει την εγγραφή από τον πίνακα reserveroom της βάσης.

```
      if (isset($_REQUEST['p']) && $_REQUEST['p'] != '') {
      $tag= $_REQUEST['p'];
```

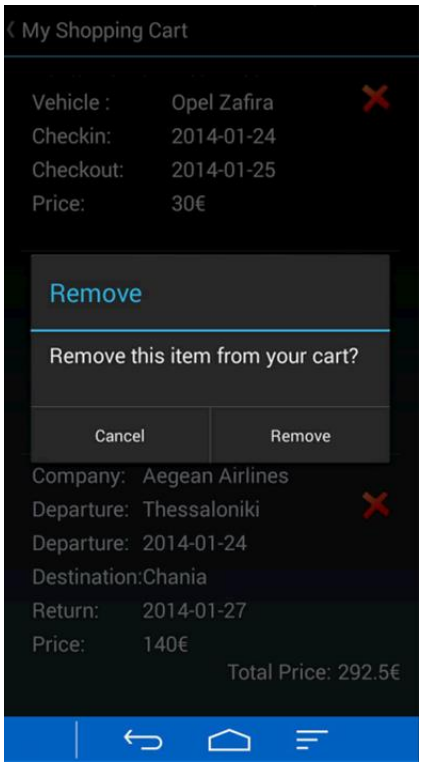
```
      $date1=$_REQUEST['checkin'];
      $date2=$_REQUEST['checkout'];

      }
      $r=mysql_query("DELETE FROM reserveroom
      WHERE reserveroom.RoomID=$tag AND reserveroom.CustomerID='1' AND
reserveroom.Checkin=$date1 AND reserveroom.Checkout=$date2",$con);
```

Ακολουθεί ο κώδικας της κλάσης deleteFromReserveroom.

```
   private class deleteFromReserveroom extends AsyncTask<Void, Void, Void>
{

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }
        @Override
        protected Void doInBackground(Void... params) {
            ServiceHandler sh = new ServiceHandler();
            sdf = new SimpleDateFormat("yyyy-MM-dd");
            firstDate = null;
            secondDate = null;
            try {
                firstDate = sdf.parse(item8);
                secondDate = sdf.parse(item10);
            } catch (ParseException e) {
                e.printStackTrace();
            }
            url1=
"http://192.168.1.3/android_api/deleteFromReserveRoom.php"+"?"+"p="+item2+
"&"+"checkin=%27"+sdf.format(firstDate)+"%27"+"&"+"checkout=%27"+sdf.forma
t(secondDate)+"%27";
            String jsonStr = sh.makeServiceCall(url1,
ServiceHandler.POST);
            return null;
        }
        @Override
        protected void onPostExecute(Void result) {
            super.onPostExecute(result);
        }
    }
```

Τέλος εφόσον διαγράψαμε ένα δωμάτιο από τον πίνακα reserveroom φροντίσαμε να αλλάξουμε το πεδίο Availability από τον πίνακα rooms, ώστε το δωμάτιο να είναι ξανά διαθέσιμο.

Η διεπαφή χρήστη του DeleteDialog φαίνεται παρακάτω.



Εικόνα 43: Διεπαφή χρήστη DeleteDialog

4.3.5 Search process

Για τη λειτουργία της αναζήτησης προσθέσαμε στην ActionBar το εικονίδιο με τον μεγεθυντικό φακό. Όταν ο χρήστης πατήσει αυτό το εικονίδιο, εκκινείτε ένα νέο Activity με την ονομασία SearchActivity.java. Το Activity αυτό περιέχει spinners τα οποία δίνουν τη δυνατότητα στο χρήστη να επιλέξει τα κριτήρια της αναζήτησης. Ο χρήστης μπορεί να κάνει μεμονωμένη αναζήτηση για κάποιο ξενοδοχείο βάση περιοχής και κατηγορίας, αναζήτηση για κάποιο όχημα βάση τύπου και θέσεων, αναζήτηση εκδρομής βάση προορισμού ή όλα τα παραπάνω μαζί. Βασικό κριτήριο για τον έλεγχο διαθεσιμότητας είναι η επιλογή της ημερομηνίας από τα αντίστοιχα εικονίδια. Όταν ο χρήστης επιλέξει τα κριτήρια προς αναζήτηση, εμφανίζεται το κουμπί "Search". Αξίζει να σημειωθεί ότι εάν ο χρήστης δεν συμπληρώσει τα απαιτούμενα κριτήρια της αναζήτησης, το κουμπί Search δεν εμφανίζεται. Ένα παράδειγμα κώδικα για τα κριτήρια του ξενοδοχείου φαίνεται παρακάτω.

```
public void checkButtonIfCanBePressed(){
       if(!hotelCategory.equals("") && !hotelArea.equals("")&&
vehicleType.equals("")&&vehicleSeats.equals("")){
           button.setVisibility(View.VISIBLE);
       } //εντολή equals: σύγκριση συμβολοσειρών
..
..
}
```

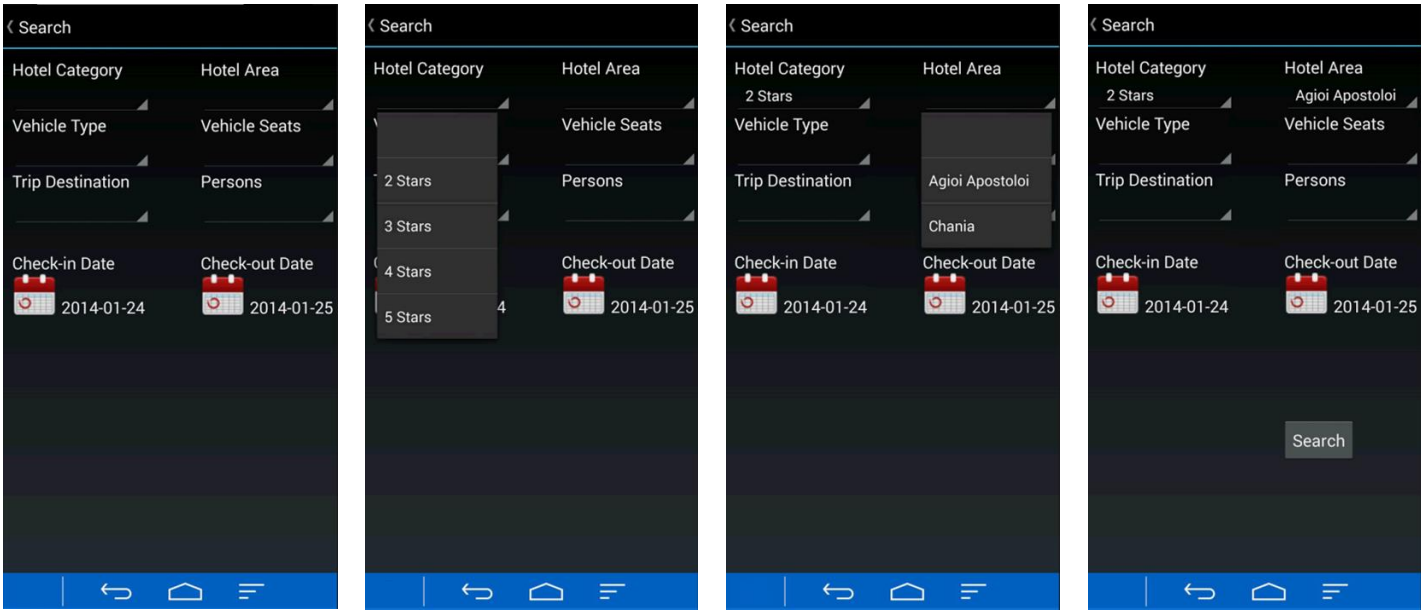
Πατώντας το κουμπί "Search" ξεκινάει ένα καινούργιο Activity, με την ονομασία SearchResultList.java, στο οποίο αποστέλλονται όλα τα κριτήρια που συμπλήρωσε ο χρήστης, μέσω ενός στιγμιότυπου Intent, στη μέθοδο onClick, που έχει ο Listener του κουμπιού.

```
    button.setOnClickListener(new View.OnClickListener() {
           @Override
           public void onClick(View v) {
```

```
                Intent intent = new Intent(v.getContext(),
SearchResultList.class);

intent.putExtra("hotelCategory",convertString(hotelCategory));
                intent.putExtra("hotelArea",convertString(hotelArea));
                intent.putExtra("vehicleType",convertString(vehicleType));

intent.putExtra("vehicleSeats",convertString(vehicleSeats));

intent.putExtra("tripDestination",convertString(tripDestination));
                intent.putExtra("persons",convertString(persons));

intent.putExtra("checkin",textView11.getText().toString());

intent.putExtra("checkout",textView12.getText().toString());
                intent.putExtra("checkinDay",checkinDay);
                intent.putExtra("checkinMonth",checkinMonth);
                intent.putExtra("checkoutDay",checkoutDay);
                intent.putExtra("checkoutMonth",checkoutMonth);
                intent.putExtra("currentDay",currentDay);
                intent.putExtra("currentMonth",currentMonth);
                startActivity(intent);
            }
        });
```

// Για τις περιπτώσεις των συμβολοσειρών που είχαν κενά π.χ.: Agioi Apostoloi φτιάξαμε τη μέθοδο convertString() η οποία παίρνει σαν όρισμα τη συμβολοσειρά και την επιστρέφει με τη μορφή Agioi%20Apostoloi ώστε να μην υπάρχουν προβλήματα στο URL.

Παρακάτω φαίνεται η διεπαφή χρήστη του SearchActivity.



Εικόνα 44: Διεπαφές χρήστη στο SearchActivity

Η SearchResultList.java είναι υπεύθυνη για την λειτουργία της αναζήτησης και την εμφάνιση των αποτελεσμάτων που επέστρεψε, τοποθετημένα σε μία λίστα. Κάθε στοιχείο που θα εμφανίζεται στη λίστα περιέχει όλες τις απαραίτητες πληροφορίες που χρειάζεται να γνωρίζει ο χρήστης για να

κάνει μία κράτηση, καθώς και ένα κουμπί προσθήκης, με σκοπό την ενσωμάτωση της κράτησης στο καλάθι αγορών.

Κατά την εκκίνηση του SearchResultList.java και συγκεκριμένα στη μέθοδο onCreate(), αποθηκεύουμε τα δεδομένα που έστειλε το προηγούμενο Activity σε global μεταβλητές. Στη συνέχεια, με τη χρήση if statement, ελέγχουμε τις συμβολοσειρές που έλαβε το Activity, για να διαπιστώσουμε εάν είναι κενές ή όχι. Για παράδειγμα, εάν η κατηγορία και η περιοχή ξενοδοχείου δεν είναι κενές ενώ όλες οι υπόλοιπες συμβολοσειρές είναι κενές, συνεπάγεται ότι πρέπει να γίνει αναζήτηση μόνο για δωμάτιο.

```
      if(!hotelCategory.equals("") && !hotelArea.equals("") &&
vehicleType.equals("") && vehicleSeats.equals("") &&
tripDestination.equals("") && persons.equals("")){
            new findHotelBasedCategoryAndArea().execute();
            new FindNotAvailableRooms().execute();
            new Control().execute();
            new GetAvailableRooms().execute();
            new UndoControl().execute();
        }
```

Εφόσον ο χρήστης επέλεξε την αναζήτηση ενός δωματίου, μέσα στην if εκτελούνται σειριακά οι πέντε παραπάνω κλάσεις, οι οποίες ακολουθούν τη μεθοδολογία του AsyncTask. Αρχικά εκτελείται η findHotelBasedCategoryAndArea η οποία επιστρέφει το hotelID που ικανοποιεί τα κριτήρια περιοχής και κατηγορίας ξενοδοχείου καθώς και τις υπόλοιπες πληροφορίες που αφορούν το ξενοδοχείο. Το SQL query του αρχείου findHotelByGategoryAndArea.php που χρησιμοποιείται για το αίτημα GET στο URL, είναι το ακόλουθο.

```
      if (isset($_REQUEST['p']) && $_REQUEST['p'] != '') {
    $category = $_REQUEST['p'];
      $area=$_REQUEST['area'];
      }
      $r=mysql_query("SELECT
hotels.Key,hotels.HotelID,hotels.Name,hotels.Category,hotels.Area FROM
hotels
      WHERE (hotels.Category=$category) AND (hotels.Area=$area)",$con);
```

Έχοντας πλέον στη διάθεσή μας το hotelID, στη συνέχεια εκτελούνται οι κλάσεις FindNotAvailableRooms, Control, GetAvailableRooms και UndoControl. Για τις τέσσερις αυτές κλάσεις έχουμε αναφερθεί αναλυτικά στο υποκεφάλαιο Book a room με τη μόνη διαφορά ότι στη κλάση GetAvailableRooms προσθέσαμε τη μέθοδο για εύρεση εναλλακτικών λύσεων.

Η κλάση αυτή φέρνει τα διαθέσιμα δωμάτια με κριτήριο την τιμή του Control όπως έχουμε πει παραπάνω. Τα δωμάτια που επέστρεψε η βάση τοποθετούνται σε ένα ArrayList και στη συνέχεια εμφανίζονται στη μέθοδο onPostExecute() μέσω ενός δεύτερου Custom Adapter που φτιάξαμε. Για την εμφάνιση των εναλλακτικών λύσεων έπρεπε να γνωρίζουμε εάν το JSON που επέστρεψε ο server είναι κενό ή όχι. Όταν λοιπόν το JSON είναι κενό σημαίνει ότι δεν υπάρχει κάποιο διαθέσιμο δωμάτιο στο ξενοδοχείο που ικανοποιεί τις επιλογές του χρήστη. Για να γνωρίζουμε εάν το JSON είναι κενό ή όχι, προσθέσαμε μία ακέραια μεταβλητή, με όνομα "bit", την οποία αρχικοποιήσαμε με την τιμή 0 στην αρχή της κλάσης. Μέσα στην doInBackground() της GetAvailableRooms, στη συνθήκη if (jsonStr != null) {…}, αλλάζουμε την τιμή του "bit", από 0 σε 1. Έτσι εάν το "bit" έχει την τιμή 1, σημαίνει ότι η βάση επέστρεψε κάποιο διαθέσιμο δωμάτιο, ενώ με την τιμή 0, πρέπει να προβούμε σε εύρεση εναλλακτικών λύσεων.

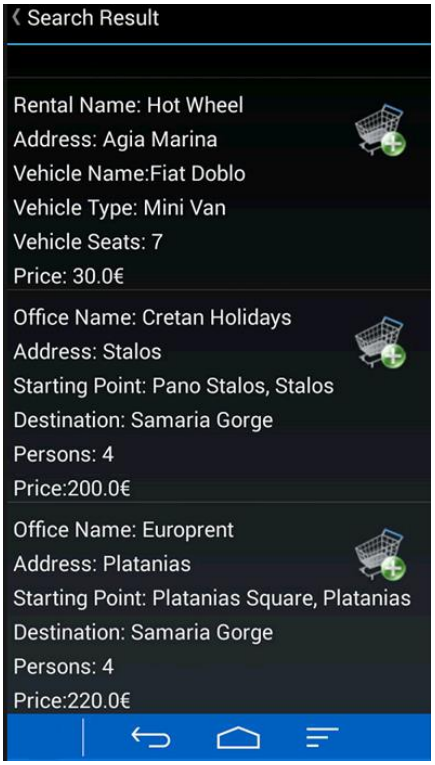
Στην onPostExecute() δημιουργήσαμε ένα if statement στο οποίο εάν το "bit" έχει την τιμή 0, αρχικά καλείται η μέθοδος changeHotelID(). Στη βάση δεδομένων τα ξενοδοχεία είναι ταξινομημένα βάση κατηγορίας. Έτσι λοιπον η changeHotelID() αλλάζει το hotelID κάθε φορά με το επόμενο hotelID. Για παράδειγμα αν στο ξενοδοχείο με hotelID 2 δεν υπάρχει κάποιο διαθέσιμο δωμάτιο, θα γίνει αναζήτηση στο ξενοδοχείο με hotelID 3 κ.ο.κ.. Επόμενο βήμα είναι η διαδικασία εύρεσης

διαθέσιμου δωματίου με το καινούργιο hotelID. Συμπέρασμα όλων αυτών είναι, ότι η διαδικασία θα επαναλαμβάνεται έως ότου βρεθεί κάποιο διαθέσιμο δωμάτιο. Όταν βρεθεί κάποιο δωμάτιο, φροντίζουμε μέσω του Adapter να εμφανιστεί στη λίστα.

```
private class GetAvailableRooms extends AsyncTask<Void, Void, Void> {
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }
        @Override
        protected Void doInBackground(Void... params) {
            ServiceHandler sh = new ServiceHandler();
            sdf = new SimpleDateFormat("yyyy-MM-dd");
            firstDate = null;
            secondDate = null;
            try {
                firstDate = sdf.parse(checkinDate);
                secondDate = sdf.parse(checkoutDate);
            } catch (ParseException e) {
                e.printStackTrace();
            }
            url=
"http://192.168.1.3/android_api/getAvailableRoomsByHotel.php"+"?"+"p="+hot
elID+"&"+"checkin=%27"+sdf.format(firstDate)+"%27"+"&"+"checkout=%27"+sdf.
format(secondDate)+"%27";
            String jsonStr = sh.makeServiceCall(url, ServiceHandler.GET);
            Log.d("Response: ", "> " + jsonStr);
            if (jsonStr != null) {
                try {
                    JSONObject jsonObj = new JSONObject(jsonStr);
                    bit=1;
                    rooms = jsonObj.getJSONArray(TAG_ROOMS);

                    for (int i = 0; i < rooms.length(); i++) {
                        JSONObject c = rooms.getJSONObject(i);
                        roomId = c.getString(TAG_ROOMID);
                        roomType = c.getString(TAG_TYPE);
                        roomPrice = c.getString(TAG_PRICE);
                        double
roomFinalPrice=Integer.parseInt(roomPrice)*totalDays;
                        itemArray.add(new
SearchResultRow(hotelKey,roomId,hotelID,checkinDate,checkoutDate,String.va
lueOf(roomFinalPrice),"","Hotel Name: "+hotelName(hotelID),"Hotel
Category: "+hotelCategory+" Stars","Address: "+hotelAddress(hotelID),"Room
Type: "+roomType,"Price: "+roomFinalPrice+"€",""));
                    }
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            } else {
                Log.e("ServiceHandler", "Couldn't get any data from the
url");
            }
            return null;
```

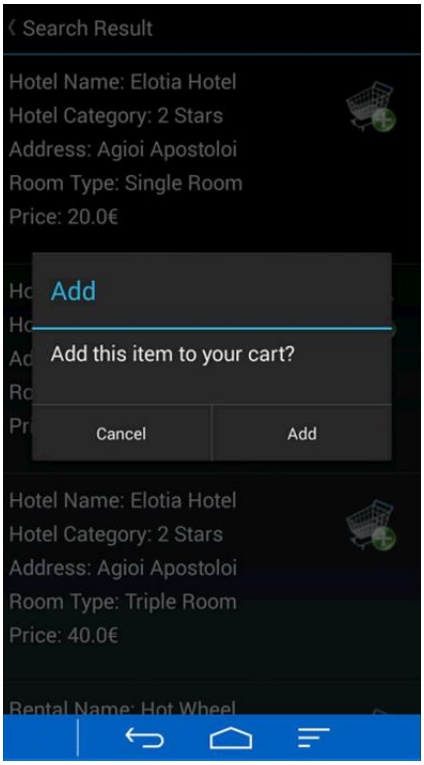
```
        }
        @Override
        protected void onPostExecute(Void result) {
            super.onPostExecute(result);
            if(bit==0){
                changeHotelID(hotelID);
                new FindNotAvailableRooms().execute();
                new Control().execute();
                new GetAvailableRooms().execute();
                new UndoControl().execute();
            }
            searchResultAdapter=new
SearchResultAdapter(SearchResultList.this,itemArray);
            searchResultList.setAdapter(searchResultAdapter);
        }
    }
```

Ένα παράδειγμα της διεπαφής χρήστη μετά από την αναζήτηση για διαθέσιμο όχημα και εκδρομή με προορισμό Samaria Gorge, φαίνεται παρακάτω.



Εικόνα 45: Διεπαφή χρήστη SearchResultList

Έχοντας ο χρήστης στη διάθεση του μία λίστα με δυνατές επιλογές, μπορεί να κάνει κράτηση πατώντας στο κουμπί της προσθήκης. Όταν ο χρήστης κάνει αυτήν την ενέργεια εμφανίζεται ένα παράθυρο διαλόγου και πατώντας την επιλογή "Add" προστίθεται η κράτηση στον αντίστοιχο πίνακα reserve της βάσης. Η διαδικασία εμφάνισης του παραθύρου διαλόγου καθώς και η αλληλεπίδραση με την επιλογή "Add" είναι η ίδια με αυτήν που ακολουθήθηκε για το RoomDialog στο υποκεφάλαιο Book a room.



Εικόνα 46: Διεπαφή χρήστη AddDialog

4.3.6 Αρχείο AndroidManifest

Φτάνοντας στο τέλος της ανάλυσης του κώδικα για την δημιουργία της εφαρμογής μας, ας ρίξουμε μία ματιά στο αρχείο AndroidManifest.xml. Έχοντας αναφέρει τη χρησιμότητα του αρχείου αυτού στην αρχή της πτυχιακής μας, θα δείξουμε ένα μέρος του κώδικα που βρίσκεται στο αρχείο.

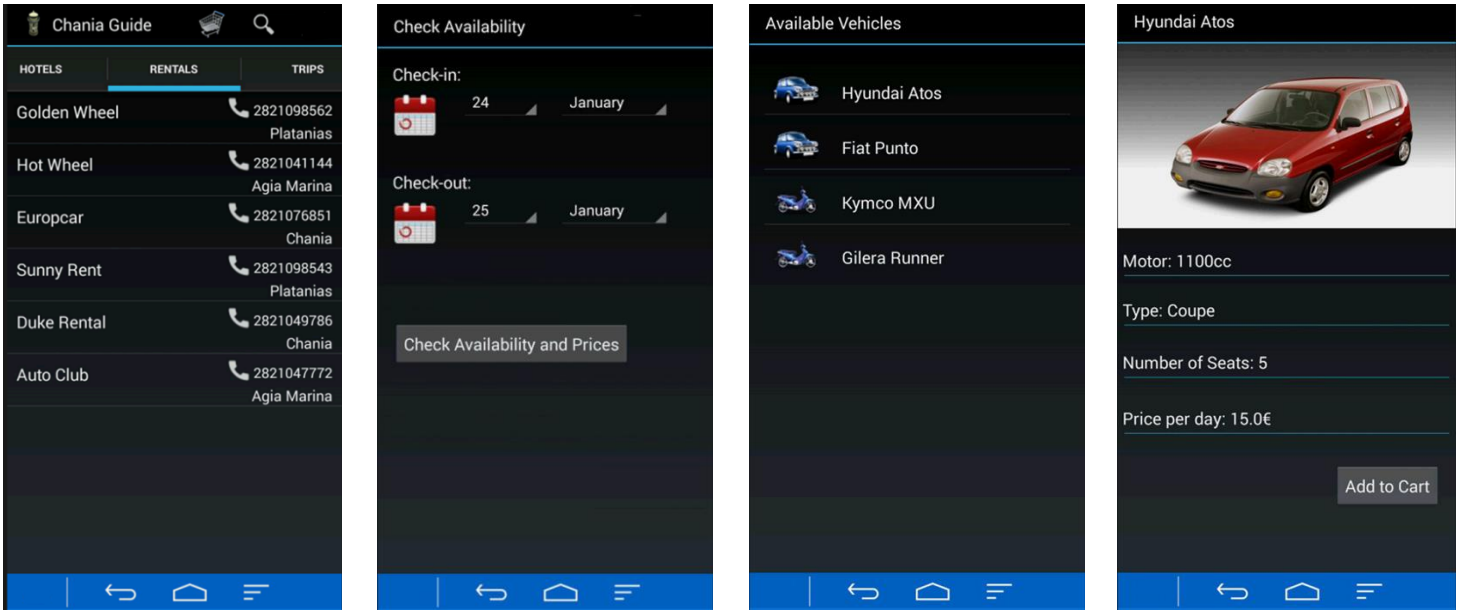
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.mixalis.final_project" >
    <uses-permission android:name="android.permission.INTERNET" /> //παροχή
δικαιώματος για πρόσβαση στο internet.
    <application
        android:allowBackup="true"
        android:icon="@drawable/lighthouse_64" //εικονίδιο εφαρμογής στη συσκευή.
        android:label="@string/app_name"
        android:theme="@style/Theme.Base.AppCompat" >
        <activity
            android:name=".MainActivity" //δήλωση ενός Activity
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
        <activity
            android:name=".HotelsFragment"
```

```
            android:label="@string/title_activity_hotels_fragment" >
        </activity>
        <activity
            android:name=".AttractionsFragment"
            android:label="@string/title_activity_attractions_fragment" >
        </activity>
        <activity
            android:name=".RentalsFragment"
            android:label="@string/title_activity_rentals_fragment" >
        </activity>
        <activity
            android:name=".TripsFragment"
            android:label="@string/title_activity_trips_fragment" >
        </activity>
        <activity
            android:name=".FlightsAndFerriesFragment"

android:label="@string/title_activity_flights_and_ferries_fragment" >
        </activity>
..
..
    </application>

</manifest>
```

4.4 Παρουσίαση λειτουργιών εφαρμογής

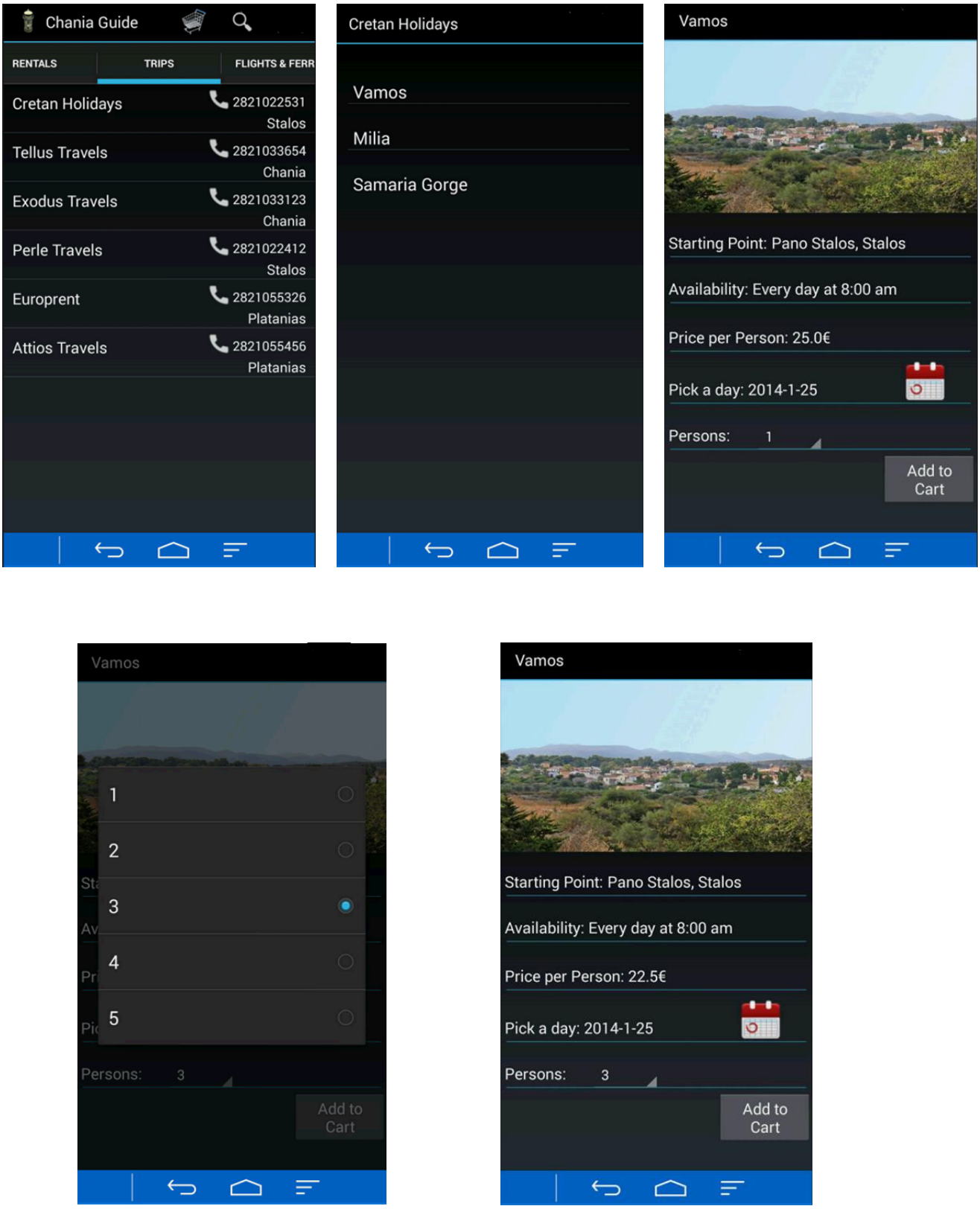
Έχουμε ήδη δείξει παραπάνω τις διεπαφές χρήστη για τις κύριες καρτέλες της εφαρμογής μας, τις διεπαφές που γίνονται ορατές στο χρήστη κατά τη διαδικασία κράτησης ενός δωματίου, καθώς και τις διεπαφές χρήστη που εμφανίζονται στις ενέργειες του Action Bar. Έτσι λοιπόν θα παρουσιάσουμε και τις υπόλοιπες διεπαφές χρήστη όπως αυτές εμφανίζονται στη συσκευή.

Οι διεπαφές που βλέπει ο χρήστης όταν θελήσει να κάνει κράτηση για ένα όχημα εμφανίζονται παρακάτω.



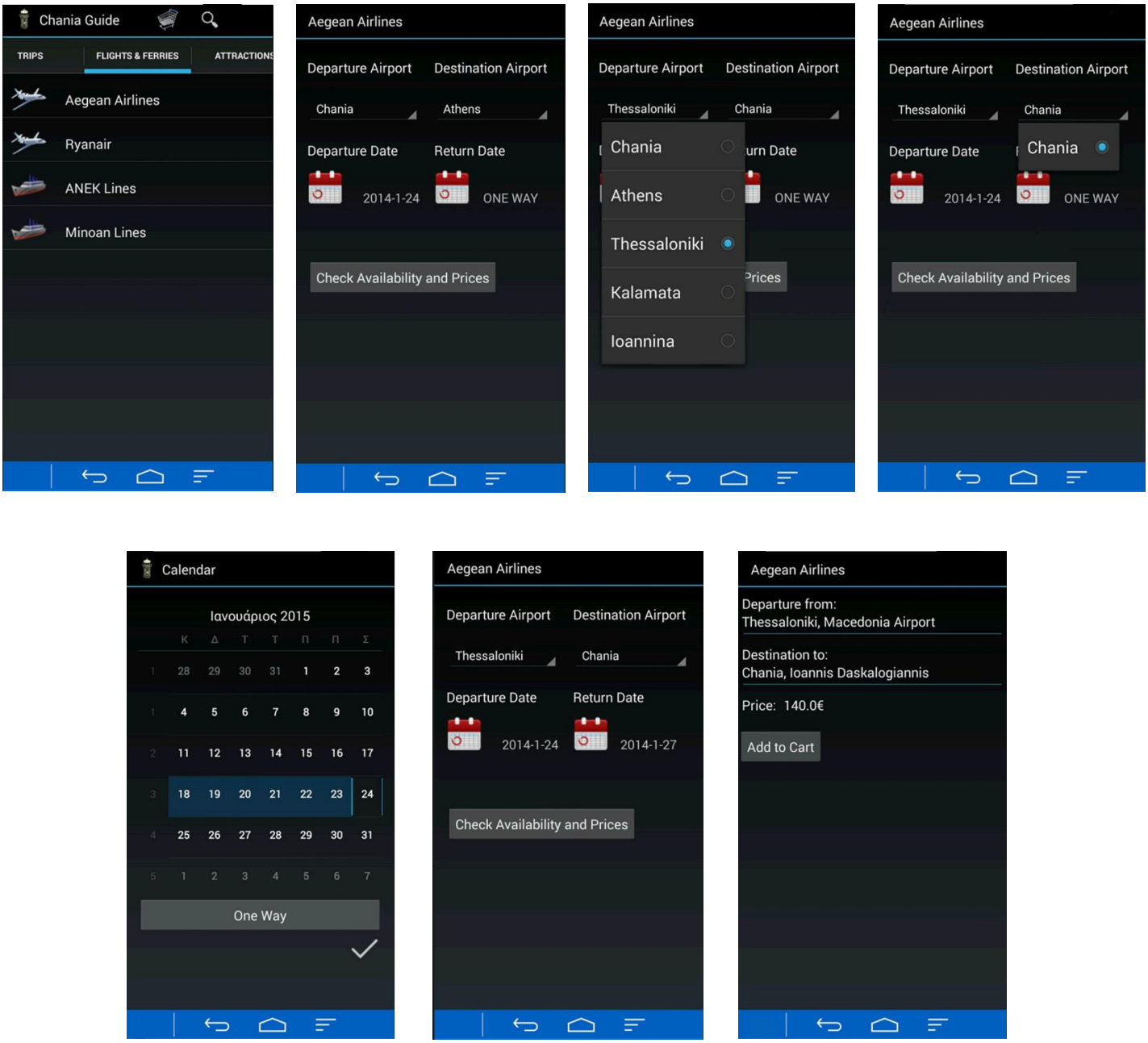
Εικόνα 47: Διεπαφές χρήστη για κράτηση οχήματος

Ακολουθούν οι διεπαφές που βλέπει ο χρήστης όταν θελήσει να κάνει κράτηση για μία εκδρομή.



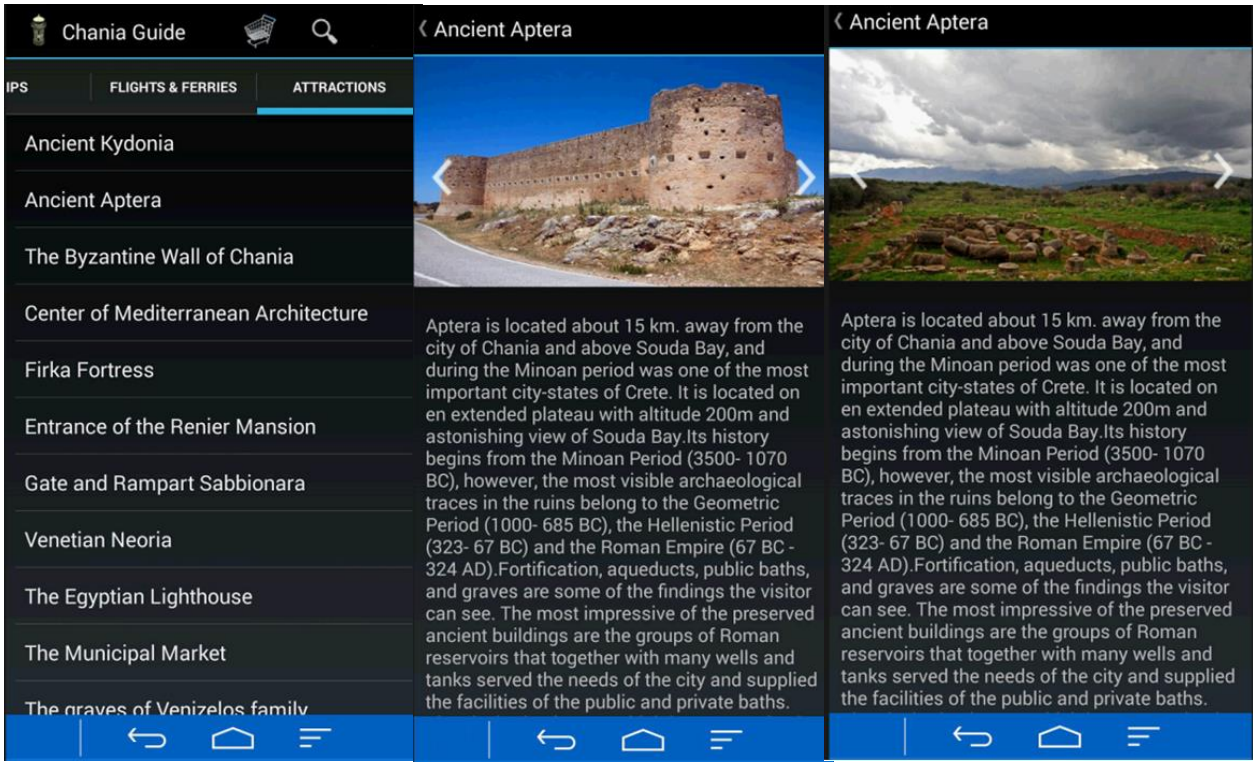
Εικόνα 48: Διεπαφές χρήστη για κράτηση εκδρομής

Στη συνέχεια παρουσιάζονται οι διεπαφές χρήστη όταν αυτός θελήσει να κάνει κράτηση για ένα δρομολόγιο ακτοπλοϊκών ή αεροπορικών γραμμών.



Εικόνα 49: Διεπαφές χρήστη για κράτηση εισητηρίου

Τέλος ακολουθούν οι διεπαφές που βλέπει ο χρήστης όταν επιλέξει να περιηγηθεί στα αξιοθέατα της πόλης μας.



Εικόνα 50: Διεπαφές χρήστη για μουσεία/αξιοθέατα

5 Βιβλιογραφία

[1]. Επίσημο site του Android:
http://developer.android.com/index.html

[2]. Βοήθημα για εκμάθηση PHP και SQL:
http://www.w3schools.com/

[3]. Ιστοσελίδες με δείγματα κώδικα και οδηγών (turorial) για ανάπτυξη εφαρμογών Android:
http://www.androidhive.info/
http://www.vogella.com/tutorials/android.html
http://sunil-android.blogspot.gr/2013/08/actionbar-tab-listfragment-in-android.html
https://github.com/codepath/android_guides/wiki/Using-an-ArrayAdapter-with-ListView
http://wptrafficanalyzer.in/blog/
http://sampleprogramz.com/android/

[4]. Πηγή εύρεσης ορισμών, εννοιών και ιστορικών αναδρομών:
http://en.wikipedia.org/

[5]. Οδηγός εγκατάστασης του JDK:
http://www.oracle.com/technetwork/java/javase/downloads/index.html

[6]. Οδηγός εγκατάστασης του WAMP:
http://www.wampserver.com/en/

[7]. Πρόγραμμα γραφικής αναπαράστασης βάσης δεδομένων:
http://www.dbschema.com/
http://dia-installer.de/

[8]. Ιστοσελίδα παρουσίασης της δομής ενός JSON:
http://jsonviewer.stack.hu/

[9]. Διάσημη ιστοσελίδα επίλυσης προβλημάτων:
http://stackoverflow.com/

[10]. Ιστοσελίδες εύρεσης εικονιδίων που χρησιμοποιήθηκαν:
http://www.iconspedia.com/
http://www.iconarchive.com/

[11]. Πληροφορίες για τα αξιοθέατα/μουσεία:
http://www.chaniatourism.com/see-do.html

[12]. Βιβλία που μελετήθηκαν:
Conder and Darcey, "*Ανάπτυξη Εφαρμογών με το Android*™", 2η έκδοση, 2011, εκδόσεις Μ.Γκιούρδας

Silberschratz, Korth and Sudarsham, "*Συστήματα Βάσεων Δεδομένων*", 4η έκδοση, εκδόσεις Μ.Γκιούρδας

Cadenhead and Lemay, "*Πλήρες Εγχειρίδιο της JAVA™ 6*", 5η έκδοση, εκδόσεις Μ.Γκιούρδας