ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη εφαρμογής προμήθειας εισιτηρίων σε κινηματογράφο με χρήση της πλατφόρμας Android

Των φοιτητών :

Κουκουνάκης Νικόλαος,
Κωνσταντίνου Άγγελος.

Αρ. Μητρώου: 063089, 063051

Επιβλέπων καθηγητής

Κος Κλεφτούρης Δημήτριος

Θεσσαλονίκη 2014

ΠΡΟΛΟΓΟΣ

Το θέμα της παρούσας πτυχιακής εργασίας είναι η ανάπτυξη εφαρμογής προμήθειας εισιτηρίων σε κινηματογράφο με χρήση της πλατφόρμας Android. Σε μια εποχή όπου το μεγαλύτερο μέρος του πληθυσμού χρησιμοποιεί φορητές συσκευές και επιθυμεί επέκταση σε αυτές των λειτουργιών που είναι διαθέσιμες στους συμβατικούς υπολογιστές, το λειτουργικό σύστημα Android αποτελεί πρωτοπόρο και οδηγό, έχοντας από το 2013 το μεγαλύτερο μερίδιο της αγοράς σε λειτουργικά συστήματα για φορητές συσκευές. Από τον Ιούνιο του 2014 οι ενεργοί χρήστες του Android ξεπερνούν το ένα δισεκατομμύριο, καθιστώντας την ανάπτυξη επιχειρηματικής δραστηριότητας στην αγορά των εφαρμογών σε Android πολλά υποσχόμενη.

Ξεκινώντας την διαδικασία δημιουργίας μιας εφαρμογής σε περιβάλλον Android θα γίνει μια σύντομη περιγραφή του λειτουργικού συστήματος του Android, όπως επίσης και των απαραίτητων εργαλείων που θα πρέπει να έχει ο προγραμματιστής προκειμένου να αναπτύξει μια εφαρμογή. Στη συνέχεια θα μελετήσουμε τις τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής, καθώς και τον τρόπο με τον οποίο εργαστήκαμε τόσο στο σχεδιαστικό κομμάτι, όσο και στην χρήση του κώδικα. Τέλος θα μιλήσουμε για τα συμπεράσματα που αποκομίσαμε όπως επίσης και θα δώσουμε ιδέες για περαιτέρω ανάπτυξη

ΠΕΡΙΕΧΟΜΕΝΑ

Εισαγωγή

Όπως αναφέραμε στον πρόλογο σκοπός της παρούσας πτυχιακής εργασίας είναι η ανάπτυξη μιας Android εφαρμογής και πιο συγκεκριμένα μιας εφαρμογής για την κράτηση εισιτηρίων σε κινηματογράφο. Πρωτού όμως περάσουμε στο προγραμματιστικό κομμάτι της ανάπτυξης καλό θα ήταν να αναφερθούμε στις συσκευές στις οποίες απευθύνεται η εφαρμογή, στο περιβάλλον του Android και τους λόγους για τους οποίους επιλέξαμε την συγκεκριμένη πλατφόρμα για την εφαρμογή μας.

Εδώ και δεκαετίες οι κινητές συσκευές τηλεφωνίας έχουν μπει για τα καλά στη ζωή μας. Από το 1973 που πρωτοεμφανίστηκε το κινητό τηλέφωνο μέχρι σήμερα η ανάπτυξη ήταν ραγδαία σε όλους τους τομείς, είτε αναφερόμαστε στο υλικό – *το πρώτο κινητό τηλέφωνο ζύγιζε δύο κιλά ενώ πλέον τα σημερινά smartphones ζυγίζουν κάτω από 200 γραμμάρια έχοντας παράλληλα επεξεργαστική ισχύ συγκρίσιμη με έναν σταθερό υπολογιστή* – είτε στο λειτουργικό – *τα πρώτα κινητά τηλέφωνα είχαν απλά μια ένδειξη για τον αριθμό που καλούσε ο χρήστης, ενώ σήμερα έχουμε τεράστιες δυνατότητες, τόσο για εργασία όσο και για διασκέδαση* – με αποκορύφωμα την εμφάνιση των «έξυπνων τηλεφώνων» μέρος των οποίων είναι και οι συσκευές που χρησιμοποιούν την πλατφόρμα Android.

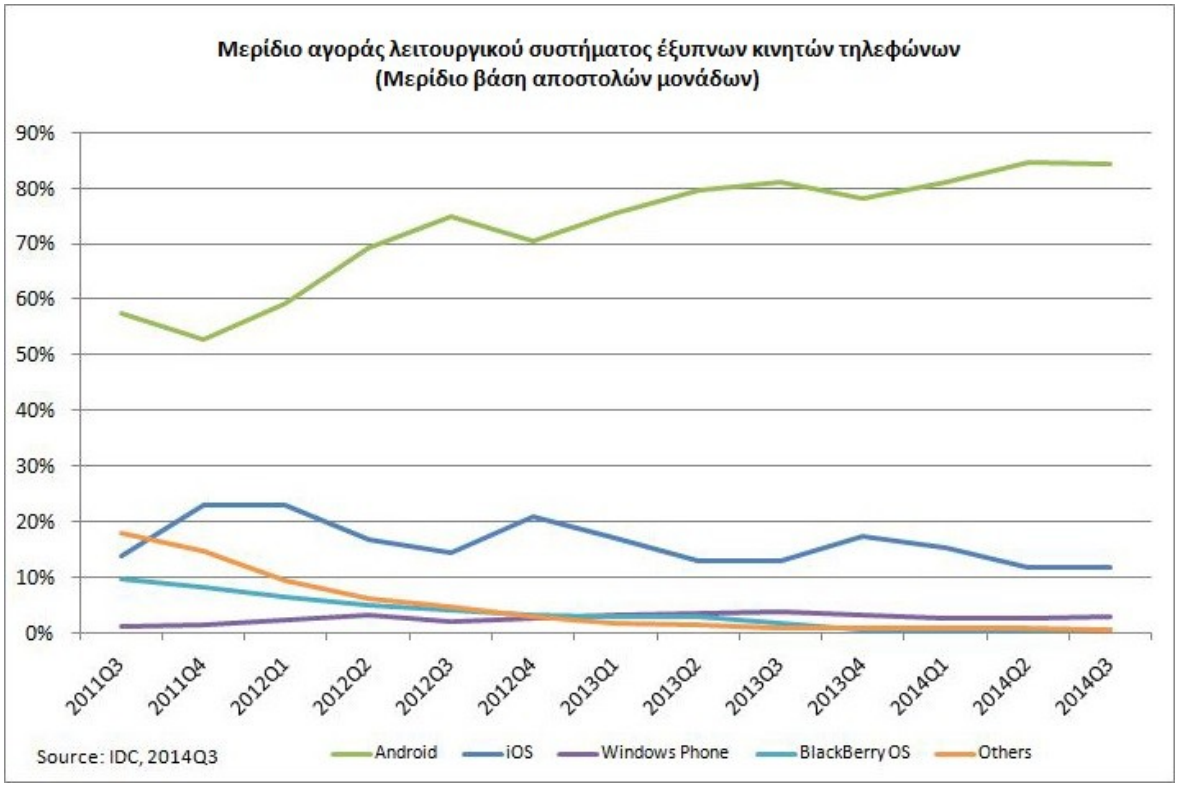
Η πλατφόρμα Android είναι ένα φορητό δωρεάν λειτουργικό σύστημα βασισμένο στον πυρήνα του Linux το οποίο αναπτύσσεται απο την Google. Συναντάται κυρίως σε ενσωματωμένα συστήματα όπως τα κινητά τηλέφωνα και τα tablet και πιο πρόσφατα τα smartwatch που αποτελούν την νέα περιοχή ανάπτυξης του Android. Το πρόγραμμα ανάπτυξης του Android ξεκίνησε το 2005 θέλοντας να κάνει δυνατή την ανάπτυξη ενός λειτουργικού συστήματος ανοιχτής πλατφόρμας, δίνοντας έτσι τη δυνατότητα στους προγραμματιστές να κάνουν δικές τους εφαρμογές. Έχοντας σαφείς περιορισμούς όπως π.χ. την κατανάλωση ενέργειας η ανάπτυξη εφαρμογών αποτελεί μια πρόκληση για τους προγραμματιστές που καλούνται να είναι προσεκτικοί τόσο με τους πόρους που χρησιμοποιούν οι εφαρμογές, όσο και με τον χρόνο για τον οποίο τους χρησιμοποιούν. Αυτό έχει ως αποτέλεσμα την ανάπτυξη καλύτερα συγκροτημένων προγραμμάτων και αναγκάζει τους προγραμματιστές να εξελίσονται διαρκώς προκειμένου να συμβαδίζουν με τις καινοτομίες που παρουσιάζει η πλατφόρμα αυξάνοντας διαρκώς τις ήδη πολλές δυνατότητες της.

1) Η πλατφόρμα Android

Καλό θα ήταν προτού ασχοληθούμε με το προγραμματιστικό κομμάτι της εργασίας να μάθουμε μερικές πληροφορίες για το λειτουργικό σύστημα του Android καθώς και να δούμε τους λόγους για τους οποίους θα προτιμούσε ένας προγραμματιστής να χρησιμοποιήσει την συγκεκριμένη πλατφόρμα για την ανάπτυξη μιας εφαρμογής.

Το λειτουργικό σύστημα του Android αυξάνει διαρκώς το μερίδιο του στο σύνολο των συσκευών που χρησιμοποιούν το διαδίκτυο. Πλέον ένα ποσοστό της τάξεως του είκοσι τοις εκατό του συνολικού αριθμού συσκευών που συνδέονται στο διαδίκτυο αποτελείται από συσκευές που χρησιμοποιούν την πλατφόρμα του android. Με τον αριθμό των συσκευών που ενεργοποιούνται καθημερινά να αγγίζει το ενάμιση εκατομμύριο και τον συνολικό αριθμό των συσκευών που την χρησιμοποιούν να ξεπερνά το ένα δισεκατομμύριο μπορούμε εύκολα να καταλήξουμε στο συμπέρασμα πως οι κινητές συσκευές θα αποτελούν σύντομα τον κύριο τρόπο πρόσβασης στο διαδίκτυο. Αναπτύσσοντας λοιπόν εφαρμογές στο προγραμματιστικό περιβάλλον του Android έχει κανείς την ευκαιρία να είναι μέρος αυτής της εξέλιξης.

Ρίχνοντας μια ματιά στις στατιστικές μπορούμε πολύ εύκολα να δούμε γιατί η ενασχόληση με την ανάπτυξη εφαρμογών στο περιβάλλον του Android μπορεί να είναι ελκυστική στους προγραμματιστές.



Όπως μπορεί να δει κανείς η πλατφόρμα του Android αυξάνει το μερίδιο της έναντι των άλλων παραγωγών λειτουργικών συστημάτων για κινητές συσκευές φτάνοντας το 3ο τετράμηνο του 2014 να κατέχει το ογδόντα τέσσερα τοις εκατό του συνόλου της αγοράς έναντι δώδεκα και τρία τοις εκατό για το iOS και τα Windows Phones αντίστοιχα που αποτελούν και τους κύριους ανταγωνιστές του.

Ο πυρήνας του Android στηρίζεται στον πυρήνα του Linux, ο οποίος είναι σταθερός και δοκιμασμένος και παρέχει στο Android το αφαιρετικό επίπεδο υλικού, επιτρέποντας του να μπορεί να χρησιμοποιηθεί σε μεγάλη ποικιλία πλατφορμών στο μέλλον. Κληρονομεί τόσο τις διαχειριστικές τεχνικές μνήμης και επεξεργαστή που διαθέτει, όσο και τα χαρακτηριστικά ασφαλείας του καθιστώντας το αρκετά αξιόπιστο. Τέλος χρησιμοποιώντας τον πυρήνα του Linux το Android κληρονόμησε και την διαθεσιμότητα του προς όλους. Ο καθένας μπορεί να το χρησιμοποιήσει όπως επιθυμεί, προσαρμόζοντας το στις δικές του ανάγκες και στο δικό του υλικό, προσόν που αξιοποιούν όλες οι εταιρίες που παράγουν συσκευές που χρησιμοποιούν το λειτουργικό. Η ιδιότητα αυτή του Android, το ότι είναι δηλαδή λογισμικό ανοιχτού κώδικα, είναι ο κύριος παράγοντας που συνετέλεσε στην ραγδαία ανάπτυξη και διάδοση του. Επίσης παρέχεται στους προγραμματιστές η δυνατότητα να επέμβουν και στο εσωτερικό των ανώτερων επιπέδων του λειτουργικού καθώς και να χτίσουν πάνω σε αυτό. Δίνοντας βάση σε αυτό το χαρακτηριστικό του το Android παρέχει στους προγραμματιστές δωρεάν ένα σύνολο εργαλείων ανάπτυξης εφαρμογών. Ο καθένας λοιπόν μπορεί να δημιουργήσει δικές του εφαρμογές με μοναδικό κόστος τον χρόνο που θα διαθέσει. Ένα πολύ θετικό υποπροϊόν της μεγάλης διάδοσης του Android είναι η ανάπτυξη μιας τεράστιας βάσης γνώσης αναφορικά με την ανάπτυξη εφαρμογών στο

συγκεκριμένο περιβάλλον. Έτσι πολλά προβλήματα που μπορεί να συναντήσει κανείς στην πρώτη του επαφή, και όχι μόνο, με το Android μπορούν να λυθούν με μεγάλη ευκολία.

Τελειώνοντας με τους λόγους για τους οποίους θα επέλεγε κανείς να χρησιμοποιήσει την πλατφόρμα αυτή για την ανάπτυξη εφαρμογών θα πρέπει να αναφερθούμε και στο κέρδος που μπορεί να βγάλει κανείς αναπτύσσοντας εφαρμογές για Android. Με τις εγκατεστημένες εφαρμογές στο σύνολο των συσκευών των χρηστών να υπερβαίνει το ένα δισεκατομμύριο γίνεται εύκολα κατανοητό ότι τόσο το κοινό στο οποίο απευθύνονται όσο και η ζήτηση για εφαρμογές είναι τεράστια. Επιπλέον η Google προσφέρει τόσο τα προγραμματιστικά εργαλεία, όσο και τα εργαλεία διαφήμισης για τους προγραμματιστές καθώς και εβδομήντα τοις εκατό επί της πληρωμής της εφαρμογής.

Φυσικά κανείς δεν μπορεί να εγγυηθεί πως η επιτυχία είναι εγγυημένη αναπτύσσοντας εφαρμογές για το Android. Υπάρχουν και αρνητικά στοιχεία όπως η δυσκολία που συναντούν οι προγραμματιστές να δημιουργήσουν εφαρμογές που να λειτουργούν σε όλες τις συσκευές που χρησιμοποιούν την πλατφόρμα, κυρίως λόγω της τεράστιας ποικιλίας των συσκευών που την χρησιμοποιούν, με διαφορετικά χαρακτηριστικά και δυνατότητες. Ακόμη ένα αρνητικό χαρακτηριστικό είναι η σχετική απροθυμία των χρηστών να πληρώσουν για τις εφαρμογές που χρησιμοποιούν. Οι μελέτες δείχνουν πως σε αυτό τον τομέα το Android βρίσκεται ακόμη πίσω απο τους ανταγωνιστές του και κυρίως το iOS το οποίο παρότι έχει αρκετά μικρότερο αγοραστικό κοινό έχει περισσότερους χρήστες διατεθειμένους να πληρώσουν για τις εφαρμογές που χρησιμοποιούν.

Αυτό που θα πρέπει κανείς να λάβει σοβαρά υπ όψιν είναι ότι η πλατφόρμα του Android είναι διαρκώς αναπτυσσόμενη με θετικά και αρνητικά στοιχεία η οποία όμως προσφέρει τεράστιες δυνατότητες

Η εξέλιξη του Android

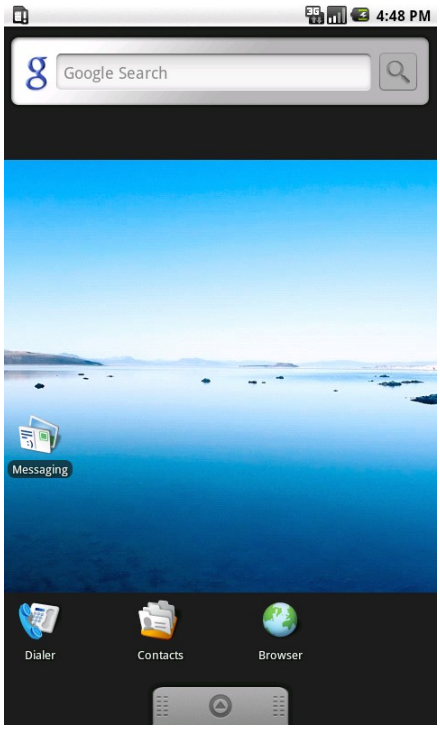
Από το Νοέμβριο του 2007 όταν και πρωτοπαρουσιάστηκε η πλατφόρμα του Android έχουν περάσει πάνω από επτά χρόνια στα οποία η εξέλιξη του λειτουργικού συστήματος ήταν ραγδαία. Εμφανίστηκε σωρεία νέων τεχνολογιών και αναβαθμίσεων, τις κυριότερες των οποίων θα προσπαθήσουμε να παρουσιάσουμε συνοπτικά παρακάτω.

1.0 - 1.1 Alpha – Beta

Η έκδοση αυτή, δεν εμφανίστηκε στο κοινό, αξίζει όμως να την αναφέρουμε καθώς από εκεί ξεκίνησε το android . Η alpha έκδοση ξεκίνησε το 2003, προτού

αγοραστεί η εταιρία από την Google ενώ η beta πρωτοεμφανίστηκε στο κοινό το Νοέμβρη του 2007 ακολουθούμενη από τα πρώτα SDK μέχρι και τον Σεπτέμβρη του επόμενου έτους, όταν και εμφανίστηκε στο κοινό με το κινητό Dream της εταιρίας HTC . Μερικά από τα χαρακτηριστικά του πρώτου κινητού τηλεφώνου με λειτουργικό Android ήταν α) Android Market, β) Web Browser, γ) Camera, δ) Media Player, ε) Wi-Fi – Bluetooth, στ) Φάκελοι, ζ) Google Mail, η) Google Maps, θ) Google Search, ι) Sms – Mms, ια) Voice Dialer, ιβ) YouTube Player, ιγ) Notifications Bar.

1.5 - 1.6 Cupcake – Donut



Η έκδοση 1.5 (Απρίλιος 2009) έφερε μια σειρά αλλαγών συμπεριλαμβανομένης και της ονομασίας της, καθώς είναι η πρώτη έκδοση που πήρε το όνομά της από ένα γλυκό (Cupcake) κάτι που ακολούθησαν και όλες οι υπόλοιπες εκδόσεις μέχρι και σήμερα. Οι αλλαγές σε αυτή την έκδοση περιλαμβάνουν την εμφάνιση των γραφικών στοιχείων (Widgets) που μπορούσαν να εισαχθούν σε άλλες εφαρμογές, όπως για παράδειγμα την αρχική οθόνη, την υποστήριξη πληκτρολογίων από άλλους προγραμματιστές με δυνατότητα πρόβλεψης λέξεων κατά την πληκτρολόγηση, την δημιουργία λεξικού με λέξεις από τον χρήστη, την δυνατότητα ανεβάσματος βίντεο και φωτογραφιών στο ίντερνετ, την αυτόματη προσαρμογή της συσκευής ανάλογα με την κλίση της, την εισαγωγή λίστας κλήσεων, όπου αναγράφονταν η ώρα και η ημερομηνία των εισερχομένων και εξερχομένων κλήσεων και την δυνατότητα εισαγωγής εικόνων στις επαφές.

Android 1.5

Η έκδοση 1.6 (Σεπτέμβριος 2009) με την ονομασία Donut συνέχισε τις αλλαγές της προηγούμενης παρέχοντας όμως κυρίως βελτιώσεις επί των ήδη

υπαρχόντων δυνατοτήτων. Συνοπτικά, στην έκδοση αυτή είχαμε αλλαγές στην δυνατότητα αναζήτησης, βελτίωση του Android Market, βελτίωση της συλλογής φωτογραφιών (Gallery), βελτίωση της αναζήτησης, υποστήριξη για WVGA αναλύσεις οθόνης και εισαγωγή νέου εκτεταμένου Gesture Framework και νέου εργαλείου GestureBuilder .

2.0 – 2.0.1 – 2.1 Eclair

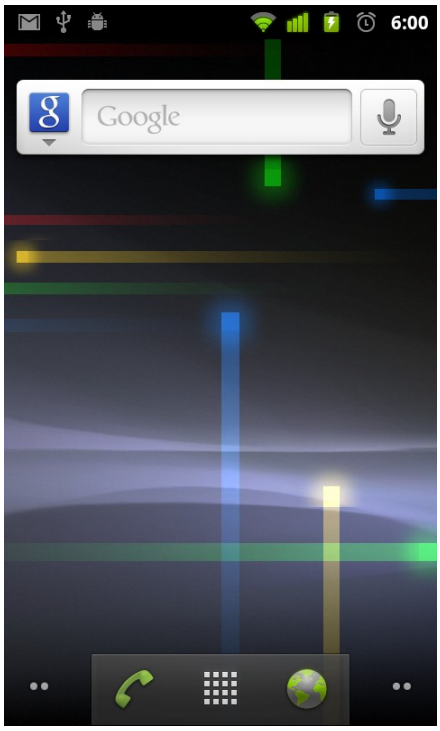
Η έκδοση 2.0 (Οκτώβριος 2009) εισήγαγε ορισμένα νέα χαρακτηριστικά. Πιο αναλυτικά, στην κάμερα έχουμε εισαγωγή υποστήριξης flash, ψηφιακό zoom, εφέ χρώματος, ρύθμιση ισορροπίας λευκού και δυνατότητα macro focus. Επιπλέον είχαμε βελτιώσεις στον Android Browser με την εισαγωγή υποστήριξης HTML 5 και double-tap zoom, βελτίωση του ημερολογίου, του Google Maps, εισαγωγή για πρώτη φορά του multi – touch, εισαγωγή ενεργών ταπετσαριών στην επιφάνεια εργασίας, δυνατότητα αναζήτησης στα μηνύματα του χρήστη, δυνατότητα εισαγωγής πολλαπλών λογαριασμών που να μπορούν να συγχρονίζονται μεταξύ τους, υποστήριξη του Microsoft exchange e-mail server, του πρωτοκόλλου 2,0 του Bluetooth και τέλος βελτιώσεις του λειτουργικού για την επίτευξη καλύτερων επιδόσεων. Οι επόμενες δυο εκδόσεις διατήρησαν το όνομα και δεν έφεραν μεγάλες αλλαγές στο λειτουργικό σύστημα παρά μόνον διορθώσεις.

2.2 – 2.2.3 Froyo

Η έκδοση 2.2 (Μάιος 2010) ήταν για χρόνια η δημοφιλέστερη έκδοση του λειτουργικού, ενώ μέχρι και σήμερα εκατοντάδες εκατομμύρια συσκευές την χρησιμοποιούν ακόμη. Με την έκδοση αυτή το Android έκανε ένα μεγάλο άλμα, προσφέροντας μεγάλες βελτιώσεις σε ταχύτητα και επιδόσεις, τόσο σε επίπεδο λειτουργικού, όσο και σε επίπεδο εφαρμογών. Επίσης οι αλλαγές επηρέασαν τόσο τους χρήστες όσο και τους προγραμματιστές. Οι χρήστες μπορούν πλέον να διαμορφώσουν την επιφάνεια εργασίας τους όπως επιθυμούν καθώς και να προσθέσουν περισσότερες από μια, έχουμε αλλαγές στην ασφάλεια με τους χρήστες να μπορούν να εισάγουν κωδικό κλειδώματος της συσκευής και επιπλέον με την εισαγωγή δυνατότητας απομακρυσμένης διαγραφής των περιεχομένων του τηλεφώνου σε περίπτωση κλοπής η απώλειας. Άλλαξε επίσης η κάμερα με την εισαγωγή νέου περιβάλλοντος χρήστη και την δυνατότητα χρήσης φλας κατά την λήψη βίντεο. Επιπλέον συγκεκριμένες συσκευές όπως το Nexus One μπορούν να χρησιμοποιηθούν ως φορητά σημεία πρόσβασης με την δυνατότητα προσφοράς ίντερνετ μέχρι και σε 8 συσκευές. Τέλος πλέον οι χρήστες μπορούν να εισάγουν και να επιλέξουν μεταξύ πολλαπλών επιλογών γλώσσας στο πληκτρολόγιο χρησιμοποιώντας το swipe στο κουμπί space του πληκτρολογίου. Πολλές ήταν και οι αλλαγές που επηρέασαν τους προγραμματιστές. Στην έκδοση αυτή είχαμε εισαγωγή του Android Could to Device Messaging , με τις εφαρμογές να μπορούν να χρησιμοποιήσουν το android cloud για να ανταλλάσουν μηνύματα με το cloud, να λαμβάνουν δεδομένα κ.α. Ακόμη έχουμε πλέον και Android Application Error Reports , για της εφαρμογές του Android Market μέσω του οποίου οι προγραμματιστές μπορούν να λαμβάνουν αναφορές σφαλμάτων από τους χρήστες. Τέλος είχαμε την εισαγωγή αρκετών νέων Developer APIs(Media

Framework, Camera API, Graphics API, Data Backup, Device policy manager, UI Framework). Για τα API(Application Programming Interfaces) του λειτουργικού Android θα αναφερθούμε αναλυτικότερα παρακάτω.

2.3 – 2.3.2 – 2.3.3 – 2.3.7 Gingerbread



Android 2.3

Στην έκδοση αυτή (Δεκέμβριος 2010) είχαμε αναβαθμίσεις τόσο σχεδιαστικά όσο και λειτουργικά. Στο σχεδιαστικό κομμάτι είχαμε την επανασχεδίαση του εικονικού πληκτρολογίου προκειμένου να προσφέρει χρηστικότητα και ταχύτητα μέσω της απλότητας. Στο λειτουργικό κομμάτι είχαμε πλήθος αλλαγών και εισαγωγή νέων χαρακτηριστικών. Τα νέα χαρακτηριστική περιελάμβαναν την εισαγωγή νέου Download manager μέσω του οποίου οι χρήστες μπορούσαν να δουν τα αρχεία που έχουν κατεβάσει και νέων εφέ ήχου και τέλος υποστήριξη για NFC (Near Field Communication) που επέτρεπαν στους χρήστες να διαβάσουν ετικέτες NFC σε πόστερ, αυτοκόλλητα η διαφημιστικά . Οι αλλαγές των υπαρχόντων λειτουργιών αφορούσαν την υποστήριξη περισσότερων από μιας καμερών στις συσκευές, την υποστήριξη επιπλέον τύπων αναπαραγωγής βίντεο και ήχου και την αναβάθμιση λειτουργιών όπως η αντιγραφή και η επικόλληση. Επιπλέον είχαμε αναβαθμίσεις που στόχευαν στην καλύτερη διαχείριση ενέργειας με το λειτουργικό πλέον να έχει το δικαίωμα να τερματίσει οποιαδήποτε εφαρμογή τρέχει στο παρασκήνιο και καταναλώνει μεγάλα ποσά ενέργειας. Τέλος είχαμε αλλαγές που στόχευαν στην διευκόλυνση των προγραμματιστών, όπως αναβαθμίσεις στους οδηγούς γραφικών και ήχων και τις ταυτόχρονη «συλλογή σκουπιδιών» μέσω του Dalvik VM με στόχο την ελαχιστοποίηση κολλημάτων στις εφαρμογές και την αύξηση της ανταπόκρισης, βελτιστοποίηση της κατανομής των γεγονότων (events) που είχε ως αποτέλεσμα την καλύτερη χρησιμοποίηση της CPU. Έτσι αυξήθηκε η αποδοτικότητα όλων των εφαρμογών. Τέλος είχαμε

εισαγωγή API για διάφορους νέους τύπους αισθητήρων, καθώς και νέου ανοιχτού API για την διαχείριση και επεξεργασία ήχου.

3.0 – 3.1 – 3.2 Honeycomb

Η έκδοση Honeycomb (Φεβρουάριος 2011) είχε την ιδιαιτερότητα πως απευθύνονταν αποκλειστικά σε ταμπλέτες. Οι διαφορές με τις προηγούμενες εκδόσεις αφορούσαν ως επί το πλείστον την διεπαφή χρήστη η οποία σχεδιάστηκε με βασικό στόχο την ευχρηστία σε μεγαλύτερες οθόνες. Επιπλέον προστέθηκαν χαρακτηριστικά όπως η μπάρα συστήματος(System Bar) στο κάτω μέρος της οθόνης για γρήγορη πρόσβαση σε ειδοποιήσεις και σε διάφορες συχνά χρησιμοποιούμενες λειτουργίες, η μπάρα ενεργειών-εφαρμογών(Action Bar) προσφέροντας διάφορες επιλογές πλοήγησης, όπως εμφάνιση των εφαρμογών της συσκευής, φωνητική αναζήτηση κ.α. Είχαμε ακόμη αλλαγές στο multitasking. Πλέον οι χρήστες μπορούσαν να κοιτάξουν όλες τις ταυτόχρονα εκτελούμενες εφαρμογές και να μεταβούν σε οποιαδήποτε από αυτές η να τις τερματίσουν. Έχουμε ακόμη την δυνατότητα σύνδεσης εξωτερικών συσκευών όπως πληκτρολόγια, επανασχεδίαση της προβολής e-mail και επαφών με χρήση δύο καρτελών για το καθένα με στόχο την διευκόλυνση του χρήστη να οργανώνει τις επαφές του και να βλέπει τα διάφορα e-mails, επανασχεδίαση της διεπαφής της κάμερας με προσθήκη νέων επιλογών, αλλαγές στον περιηγητή με προσθήκη καρτελών και ανώνυμης περιήγησης και τέλος επανασχεδίαση του πληκτρολογίου με γνώμονα το νέο μεγαλύτερο μέγεθος οθονών.

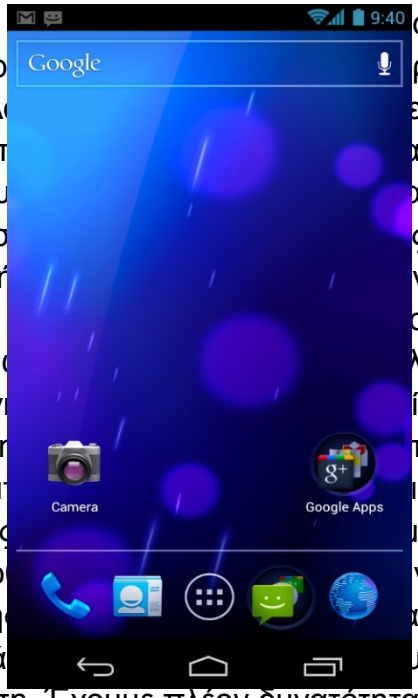
Android 3.0

Αρκετές ήταν και οι διαφοροποιήσεις αναφορικά με την προγραμματιστική πλευρά του λειτουργικού. Το Honeycomb προσέφερε πλήρη συμβατότητα με προηγούμενες εκδόσεις του Android, προσέφερε αλλαγή του προηγούμενου θέματος του Android στην έκδοση 3.0 με νέα εμφάνιση για τις εφαρμογές των προγραμματιστών, έδωσε την δυνατότητα προσθήκης εναλλακτικών διεπαφών ανάλογα με το μέγεθος της οθόνης, δίνεται η δυνατότητα διαχωρισμού μιας οθόνης σε κομμάτια(fragments) προσθήκη μπάρας ενεργειών(Action Bar) στις εφαρμογές, όπου κάθε εφαρμογή δίνει διάφορες επιλογές στον χρήστη, εισαγωγή του DragEvent framework έτσι ώστε οι εφαρμογές να παρέχουν όπως οι αντίστοιχες

των προσωπικών υπολογιστών χαρακτηριστικά drag ‘n’ drop. Ακόμη έχουμε πλέον υποστήριξη πολλαπλών πυρήνων, μέσω του Renderscript 3D παρέχεται ένα API για την δημιουργία και επιτάχυνση 3D γραφικών και δεδομένων. Επιτεύχθηκε επίσης και επιτάχυνση στον τομέα των δισδιάστατων γραφικών με χρήση του OpenGl renderer που έφερε αύξηση των επιδόσεων σε πάρα πολλές εφαρμογές του Android.

4.0 – 4.0.2 – 4.0.3 – 4.0.4 Ice Cream Sandwich

Android 4.0



Πλησιάζοντας όλο ονη μορφή του Android η επόμενη έκδοση (Οκτώβρ ρα νέων χαρακτηριστικών θέλοντας όμως παράλληλ εσμος μεταξύ των κινητών τηλεφώνων και των ταμτ αν μέχρι εκείνη τη στιγμή διαφορετικές εκδόσεις του α όμως αυτή διασύνδεσης των διαφορετικών τύπων σ ς προγραμματιστές από το να προσφέρουν ένα πλή ν. Έμφαση δόθηκε στην αλληλεπίδραση συσκευής ονομιών(Gestures) για την επίτευξη απλών ενεργει λτιώθηκε αισθητά με την προσθήκη νέων εικον ίχαν τη δυνατότατα να περιηγηθούν αυτόματα στ τις πρόσφατες εφαρμογές. Η μπάρα συστήματος μ ιμα εμφανής και κατά τη διάρκεια χρήσης κάποιας μένη στις εφαρμογές που χρησιμοποιούν πλήρη ο ν μπάρα ενεργειών. Μια σημαντική αλλαγή τη αν η προσπάθεια των προγραμματιστών να κά μόσιμο στις διαφορετικές ανάγκες του κάθε χρήστη. Έχουμε πλέον δυνατότητα αλλαγής μεγέθους των Widgets δυνατότητα δημιουργίας φακέλων εφαρμογών στην επιφάνεια εργασίας, εισαγωγή νέων ενεργειών στην οθόνη κλειδώματος, νέες επιλογές γρήγορης απάντησης κ.α. Στα δεδομένα κινητής τηλεφωνίας οι χρήστες έχουν την επιλογή να βάλουν όρια, προειδοποιήσεις ανάλογα με την χρήση και να διακόψουν τη χρήση τους όταν αυτή ξεπεράσει κάποιο όριο. Εισάγεται για πρώτη φορά ενσωματωμένος επεξεργαστής φωτογραφιών ενώ επανασχεδιάζεται η εφαρμογή της κάμερας και εισάγεται δυνατότητα εγγραφής 1080p βίντεο. Αλλαγές έχουμε και στους περιηγητές με δυνατότητα εισαγωγής σελιδοδεικτών στον Chrome από τον λογαριασμό του χρήστη. Ακόμη εισάγεται το Android Beam μια εφαρμογή NFC για την γρήγορη ανταλλαγή διάφορων δεδομένων(επαφών, σελιδοδεικτών του Internet, βίντεο κ.α.). Τέλος διορθώνονται πολλά σφάλματα στον κώδικα, γίνονται βελτιώσεις στα γραφικά, τις βάσεις δεδομένων, τη σταθερότητα της έκδοσης.

4.1 – 4.2 – 4.3 Jelly Bean

Η έκδοση αυτή (Ιούλιος 2012) είχε ως κύριο στόχο την βελτίωση της λειτουργικότητας και της απόδοσης του περιβάλλοντος χρήστη. Για να το πετύχει αυτό η Google χρησιμοποίησε το “Project Butter” του οποίου μέλημα ήταν να παρέχει μια πιο «ομαλή» αλληλεπίδραση στον χρήστη χρησιμοποιώντας πρόβλεψη αγγίγματος, triple buffering, vertical synchronization και σταθερό ρυθμό καρέ στα 60 fps. Βελτιώσεις είχαμε και στις οθόνες κλειδώματος, στον ήχο, την μπάρα ενημερώσεων, στην ενσωματωμένη εφαρμογή ρολογιού, στην προσβασιμότητα μέσω αλλαγών στις χειρονομίες και στην αναγνώριση φωνής κ.α.

4.4 – 4.4.2 KitKat

Για πρώτη φορά η ελάχιστη συνιστώμενη ram που απαιτείται για να τρέξουν οι συσκευές το λειτουργικό γίνεται 512 mb και μάλιστα όλες οι συσκευές με λιγότερη ram πρέπει να δηλώνονται ως low ram devices. Η έκδοση KitKat (Οκτώβριος 2013) έφερε αλλαγές αναφορικά με την εμφάνιση της διεπαφής χρήστη, του ρολογιού, της εμφάνισης της μπάρας ενεργειών και της μπάρας πλοήγησης μέσα στις εφαρμογές με την εισαγωγή δυνατότητας διαφάνειας στις μπάρες. Επιπλέον είχαμε βελτιώσεις για την απόδοση συσκευών με χαμηλή μνήμη, εισαγωγή δυνατότητας ασύρματης εκτύπωσης και σειρά αλλαγών σε επίπεδο προγραμματιστών με εισαγωγή νέων API’s όπως το ActivityManager.isLowRamDevice(), που σου επιτρέπει να αλλάξεις την συμπεριφορά μιας εφαρμογής ανάλογα με την μνήμη της απενεργοποιώντας για παράδειγμα χαρακτηριστικά της εφαρμογής που απαιτούν μεγάλα ποσά μνήμης. Ενέργειες έγιναν και στην κατεύθυνση της εξοικονόμησης ενέργειας, με την

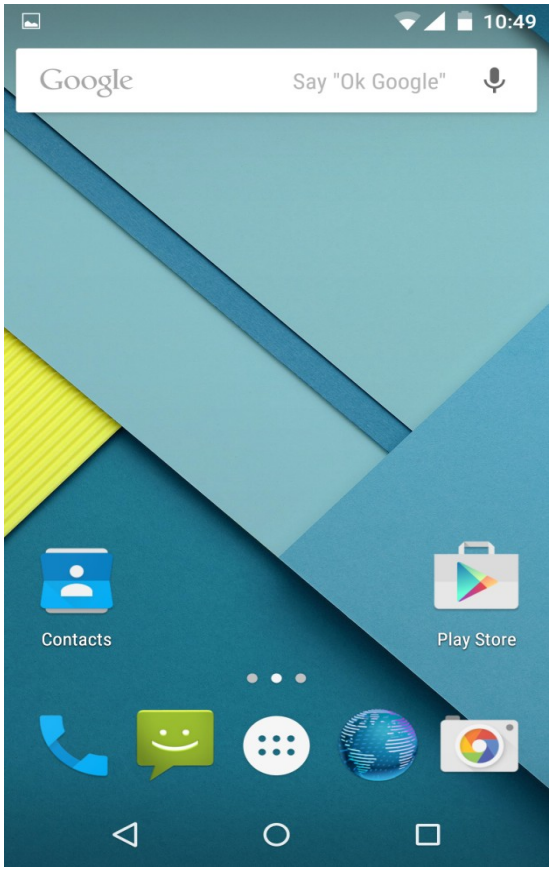
εμφάνιση μιας βελτιστοποίησης για δραστηριότητες διάφορων αισθητήρων στις συσκευές η οποία μπορεί να μειώσει δραματικά την κατανάλωση ενέργειας. Παρουσιάστηκε ένας νέος πάροχος για SMS με νέα APIs για την διαχείριση αποθήκευσης και ανάκτησης μηνυμάτων. Επίσης από την έκδοση αυτή οι εφαρμογές μπορούν να χρησιμοποιούν όλη την οθόνη χωρίς όμως να χάνεται η λειτουργικότητα της μπάρας πλοήγησης και της μπάρας κατάστασης οι οποίες μπορούν να αποκρυφτούν καθώς με μια απλή κίνηση μπορούν να επανέλθουν. Τέλος έγιναν αναβαθμίσεις σε επίπεδο ασφαλείας, με αλλαγή της πολιτικής πρωτοκόλλου ασφαλείας SELinux που χρησιμοποιεί το Android και την αναβάθμιση των αλγορίθμων κρυπτογράφησης με την προσθήκη δυο νέων για τις εφαρμογές που χρησιμοποιούν συνδέσεις δεδομένων.

5.0 Lollipop

Η πιο πρόσφατη μέχρι σήμερα έκδοση του λειτουργικού είναι η Lollipop (Νοέμβριος 2014) η οποία ήρθε με σημαντικές αλλαγές τόσο στο κομμάτι της εμφάνισης αλλά κυρίως στο προγραμματιστικό κομμάτι του λειτουργικού. Αντικαταστάθηκε ο μεταφραστής Dalvik ο οποίος είχε δημιουργηθεί αποκλειστικά για το Android Και χρησιμοποιούσε Just in time (JIT) compilation, έκανε δηλαδή την μετάφραση ενός προγράμματος κατά την διάρκεια εκτέλεσης με τον Android Runtime που χρησιμοποιεί Ahead of time (AOT) compilation, δηλαδή μεταγλώττιση του κώδικα μιας γλώσσας υψηλού ή ενδιάμεσου επιπέδου σε κώδικα μηχανής, βελτιώνοντας έτσι την απόδοση του συστήματος. Μια ακόμη μεγάλη αλλαγή σε επίπεδο υλικού αυτή τη φορά είναι πως για πρώτη φορά υποστηρίζονται 64μπιτοι επεξεργαστές ενώ για ακόμη μια φορά είχαμε βελτιώσεις και στον τομέα της κατανάλωσης ενέργειας με στόχο την βελτίωση της χρήσης της μπαταρίας μέσω του Project Volta. Μερικές από τις αλλαγές περιλαμβάνουν την εισαγωγή ενός νέου τύπου εξοικονόμησης μπαταρίας και την δημιουργία «παρτίδων εργασιών» με στόχο την μείωση του συνολικού χρόνου που χρησιμοποιείται το σύστημα. Όσον αφορά την εμφάνιση η έκδοση αυτή φέρει ένα ανανεωμένο σύστημα ειδοποιήσεων με τις ειδοποιήσεις πλέον να φαίνονται σε μορφή κάρτας. Οι ειδοποιήσεις είναι ορατές και στην οθόνη κλειδώματος ενώ εισάγεται και ένα κουμπί «μην ενοχλείτε» για τις ειδοποιήσεις. Επανασχεδιάζεται το Android 5.0

μενού των πρόσφατα χρησιμοποιημένων εφαρμογών με την δυνατότητα μια εφαρμογή να έχει πολλές καρτέλες στο μενού αυτό. Επίσης η έκδοση αυτή περιέχει πολλές νέες λειτουργίες για τους προγραμματιστές με την εισαγωγή πάνω από 5.000 νέων API για χρήση από τις εφαρμογές, όπως για παράδειγμα η δυνατότητα αποθήκευσης φωτογραφιών σε RAW μορφή.

Μπορούμε να δούμε πως το λειτουργικό έχει εξελιχθεί τρομερά με την πάροδο των χρόνων και το σίγουρο είναι ότι θα συνεχίσει να εξελίσσεται. Οι δυνατότητες που δίνονται για την ανάπτυξη εφαρμογών με το λειτουργικό Android είναι τεράστιες και συνεχίζουν να επεκτείνονται. Στην παρούσα εργασία προσπαθήσαμε να αξιοποιήσουμε το δυνατόν περισσότερες από τις δυνατότητες αυτές με στόχο την δημιουργία μιας εύχρηστης, εύρωστης και πάνω απ όλα



χρήσιμης εφαρμογής. Παρακάτω θα δούμε τα εργαλεία που χρησιμοποιήσαμε στην προσπάθεια μας αυτή.

2) Τα εργαλεία ανάπτυξης της εφαρμογής μας

Στο προηγούμενο κεφάλαιο ρίξαμε μια σύντομη ματιά στην πορεία εξέλιξης του λειτουργικού συστήματος Android από τα πρώτα του βήματα μέχρι σήμερα και των αλλαγών που έγιναν στο διάστημα αυτό. Σε αυτό το κεφάλαιο θα δώσουμε μια περιγραφή των βασικών εργαλείων που χρησιμοποιούμε για την ανάπτυξη της εφαρμογής με στόχο την καλύτερη κατανόηση του περιβάλλοντος και των τεχνικών που χρησιμοποιήσαμε.

2.1) Η γλώσσα προγραμματισμού Java

Η ανάπτυξη των εφαρμογών Android γίνεται με τη γλώσσα προγραμματισμού Java. Θα πρέπει όμως να ξεκαθαρίσουμε πως αυτή δεν είναι η πλήρης έκδοση της Java την οποία χρησιμοποιούν οι προγραμματιστές της Java Platform αλλά ένα κομμάτι των βιβλιοθηκών της Java οι οποίες είναι ειδικές για το Android. Αυτό το μικρότερο σύνολο της Java αποκλείει τις κλάσεις εκείνες οι οποίες δεν είναι κατάλληλες για κινητές συσκευές. Αυτός είναι ένας από τους λόγους για τους οποίους η ανάπτυξη εφαρμογών για το Android είχε τόσο μεγάλη απήχηση στους προγραμματιστές, ακριβώς το γεγονός ότι για την ανάπτυξη εφαρμογών χρησιμοποιείται η Java, μια γλώσσα γνωστή, διαδεδομένη και εύχρηστη. Για να χρησιμοποιήσουμε όμως την Java θα πρέπει πρώτα να εγκαταστήσουμε στον υπολογιστή μας το JDK, το εργαλείο ανάπτυξης της Java το οποίο περιέχει ένα

σύνολο εργαλείων που χρησιμοποιεί η Java, όπως ο Javac(μεταφραστής), ο Javah, ο Jar κ.λπ. Η Java ανήκει στην Oracle στης οποίας τον ιστότοπο μπορούμε να βρούμε και να κατεβάσουμε τα εργαλεία που θα μας χρειαστούν. Αναλυτικότερα η διαδικασία εγκατάστασης του JDK:

1. Βρίσκουμε το αρχείο εγκατάστασης ανάλογα με λειτουργικό μας σύστημα στον ιστότοπο της Oracle, το αποθηκεύουμε και το εκτελούμε.

Java SE Development Kit 8u31

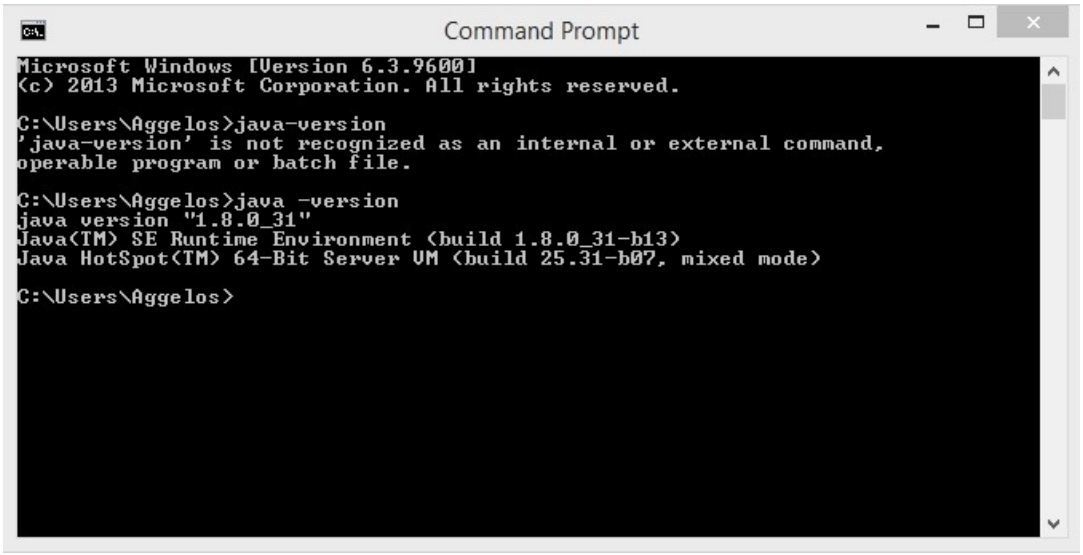
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

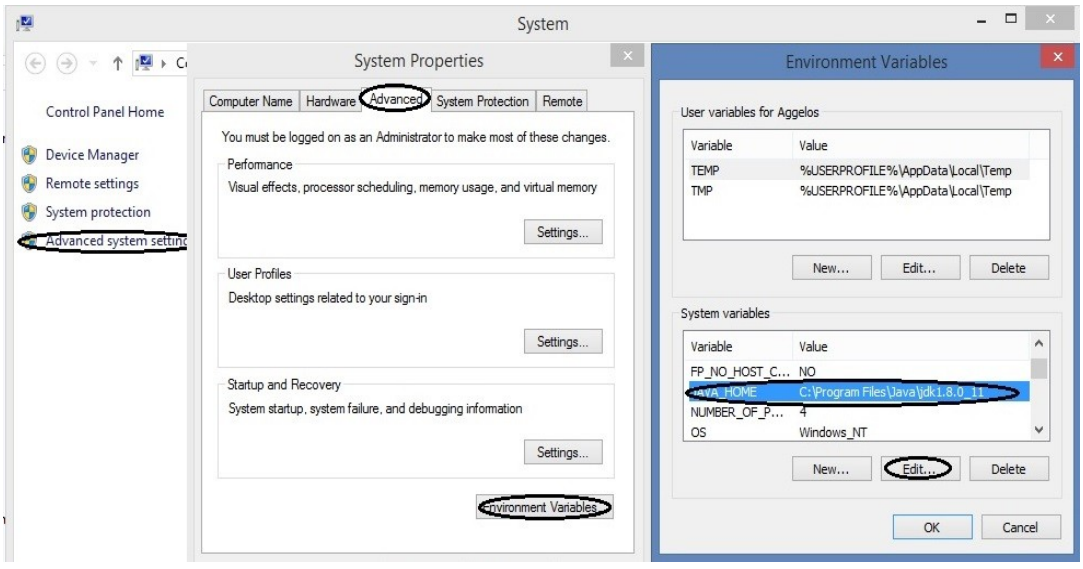
Product / File Description	File Size	Download
Linux x86	135.24 MB	jdk-8u31-linux-i586.rpm
Linux x86	154.91 MB	jdk-8u31-linux-i586.tar.gz
Linux x64	135.62 MB	jdk-8u31-linux-x64.rpm
Linux x64	153.45 MB	jdk-8u31-linux-x64.tar.gz
Mac OS X x64	209.17 MB	jdk-8u31-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	136.91 MB	jdk-8u31-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	97.11 MB	jdk-8u31-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	137.51 MB	jdk-8u31-solaris-x64.tar.Z
Solaris x64	94.82 MB	jdk-8u31-solaris-x64.tar.gz
Windows x86	157.96 MB	jdk-8u31-windows-i586.exe
Windows x64	170.36 MB	jdk-8u31-windows-x64.exe

2. Ακολουθούμε τα βήματα του οδηγού εγκατάστασης, στο βήμα αυτό θα μας ζητηθεί να επιλέξουμε ποια κομμάτια του JDK θέλουμε να εγκατασταθούν, σε ποιο σημείο θα γίνει η εγκατάσταση και τέλος θα μας ζητηθεί προαιρετικά να εγγραφούμε στην Oracle.

3. Στο σημείο αυτό θα πρέπει να βεβαιωθούμε ότι η εγκατάσταση της Java έγινε σωστά. Από την γραμμή εντολών (cmd) πληκτρολογούμε java -version της οποίας το αποτέλεσμα θα πρέπει να είναι κάπως έτσι:



Όπως μπορούμε να δούμε στο μήνυμα που εμφανίζεται αναγράφεται η έκδοση της Java που χρησιμοποιούμε, η έκδοση του Runtime Environment καθώς και η έκδοση του λειτουργικού μας συστήματος. Υπάρχει πάντα περίπτωση να μην λάβουμε το κατάλληλο μήνυμα. Σε αυτή την περίπτωση η το λειτουργικό μας σύστημα δεν γνωρίζει τον τόπο που έχει γίνει η εγκατάσταση, κάτι που θα πρέπει να κάνουμε χειροκίνητα. Αυτό γίνεται μέσω των μεταβλητών περιβάλλοντος της καρτέλας «για προχωρημένους» που βρίσκεται στις ρυθμίσεις συστήματος για προχωρημένους του υπολογιστή. Δημιουργούμε μια νέα μεταβλητή με όνομα JAVA_HOME η οποία «δείχνει» στο φάκελο που έχουμε εγκαταστήσει το JDK.



Μετά την εγκατάσταση του JDK μπορούμε να προχωρήσουμε στην εγκατάσταση του προγράμματος που θα χρησιμοποιήσουμε για την ανάπτυξη της εφαρμογής μας. Μέχρι πρότινος το βασικό εργαλείο για την ανάπτυξη εφαρμογών σε περιβάλλον android ήταν το Eclipse IDE όμως το Eclipse τείνει να αντικατασταθεί

από το επίσημο πλέον εργαλείο ανάπτυξης εφαρμογών για Android που είναι το Android Studio.

2.2) Το Eclipse

Για την ανάπτυξη της εφαρμογής μας θα χρησιμοποιήσουμε το Android Studio όμως καλό θα ήταν να πούμε μερικά λόγια για το Eclipse καθώς ήταν για χρόνια το εργαλείο που χρησιμοποιούσαν οι προγραμματιστές. Το Eclipse διαχειρίζεται από το ίδρυμα Eclipse και δεν χρησιμοποιείται αποκλειστικά για ανάπτυξη εφαρμογών σε Android. Παρέχει ένα περιβάλλον για την ανάπτυξη εφαρμογών ενσωματώνοντας ένα πλήθος λειτουργιών. Αυτό που έκανε δυνατή την ανάπτυξη εφαρμογών για Android είναι η δυνατότητα που προσφέρει το Eclipse για υποστήριξη διαφόρων plug-in τα οποία επεκτείνουν τις λειτουργίες που προσφέρει. Για την ανάπτυξη Android εφαρμογών χρησιμοποιείται το ADT(Android Development Tools) Plug-in το οποίο δίνει την δυνατότητα συγγραφής κώδικα για το περιβάλλον του Android. Η χρήση του ADT επιτυγχάνεται με την εγκατάσταση του Android SDK (Software Development Kit). Το ADT προσφέρει μια διεπαφή χρήστη ενώ το SDK περιέχει όλα τα εργαλεία που χρειάζονται για την δημιουργία, την εκτέλεση και τον έλεγχο των εφαρμογών. Περισσότερα για το Android SDK θα δούμε παρακάτω.

2.3) Το Android Studio

Το Android Studio ανακοινώθηκε τον Μάιο του 2013 όταν και έγινε διαθέσιμη η πρώτη beta έκδοση του. Η πρώτη σταθερή του έκδοση έγινε διαθέσιμη μόλις τον Δεκέμβριο του 2014.Βασίζεται στο IntelliJ IDEA και έχει σχεδιαστεί αποκλειστικά για την ανάπτυξη εφαρμογών Android ενώ είναι διαθέσιμο για λειτουργικά Windows, Mac OS και Linux. Παρέχει πληθώρα λειτουργιών οι οποίες επεκτείνονται με κάθε νέα έκδοση. Οι λειτουργίες που παρέχει στην τωρινή του έκδοση είναι οι παρακάτω:

1. Live Layout: Απόδοση της εμφάνισης της εφαρμογής σε πραγματικό χρόνο, ανάλογα με τις αλλαγές που πραγματοποιεί ο προγραμματιστής
2. Developer Console: Μια κονσόλα προγραμματιστή που προσφέρει συμβουλές βελτιστοποίησης, βοήθεια στη μετάφραση του προγράμματος, παρακολούθηση παραπομπών, μετρήσεις χρήσης.
3. Πρόνοια για διάθεση beta εκδόσεων εφαρμογών
4. Εκτέλεση των εφαρμογών με χρήση του προτύπου Gradle
5. Επανασχεδίαση του κώδικα για καλύτερη αλληλεπίδραση με το Android
6. Εργαλεία για την αποφυγή προβλημάτων σχετικά με την χρηστικότητα, την απόδοση και την συμβατότητα των προγραμμάτων.
7. Υποστήριξη για την δημιουργία εφαρμογών Android Wear.

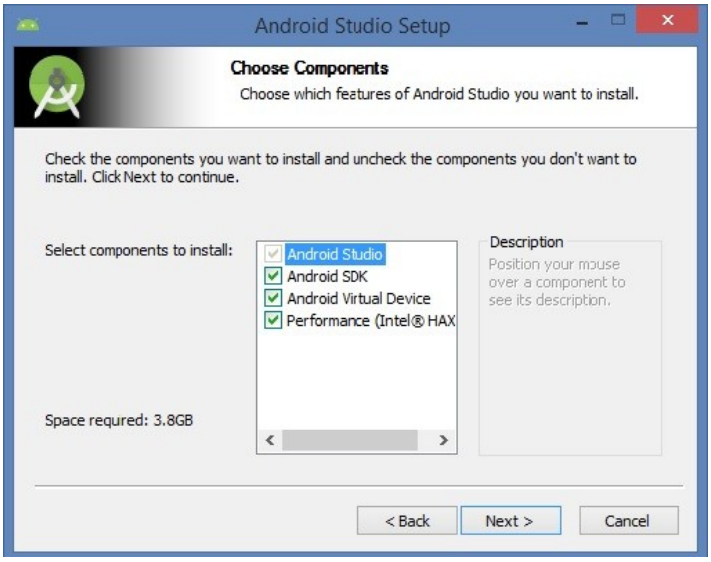
8. Ενσωματωμένη υποστήριξη του Google Cloud που δίνει τη δυνατότητα υποστήριξης του Google Cloud Messaging
9. Layout editor: Δίνει τη δυνατότητα στους χρήστες να χρησιμοποιούν Drag-and-Drop στοιχεία καθώς και επισκόπηση της εμφάνισης της εφαρμογής σε διαφορετικές οθόνες.
10. Πρότυπα για τη δημιουργία σχεδίων και συστατικών για το Android.
11. Δυνατότητα μετακίνησης από άλλα προγράμματα όπως το Eclipse

Επιλέγουμε τα χαρακτηριστικά που θέλουμε να εγκαταστήσουμε

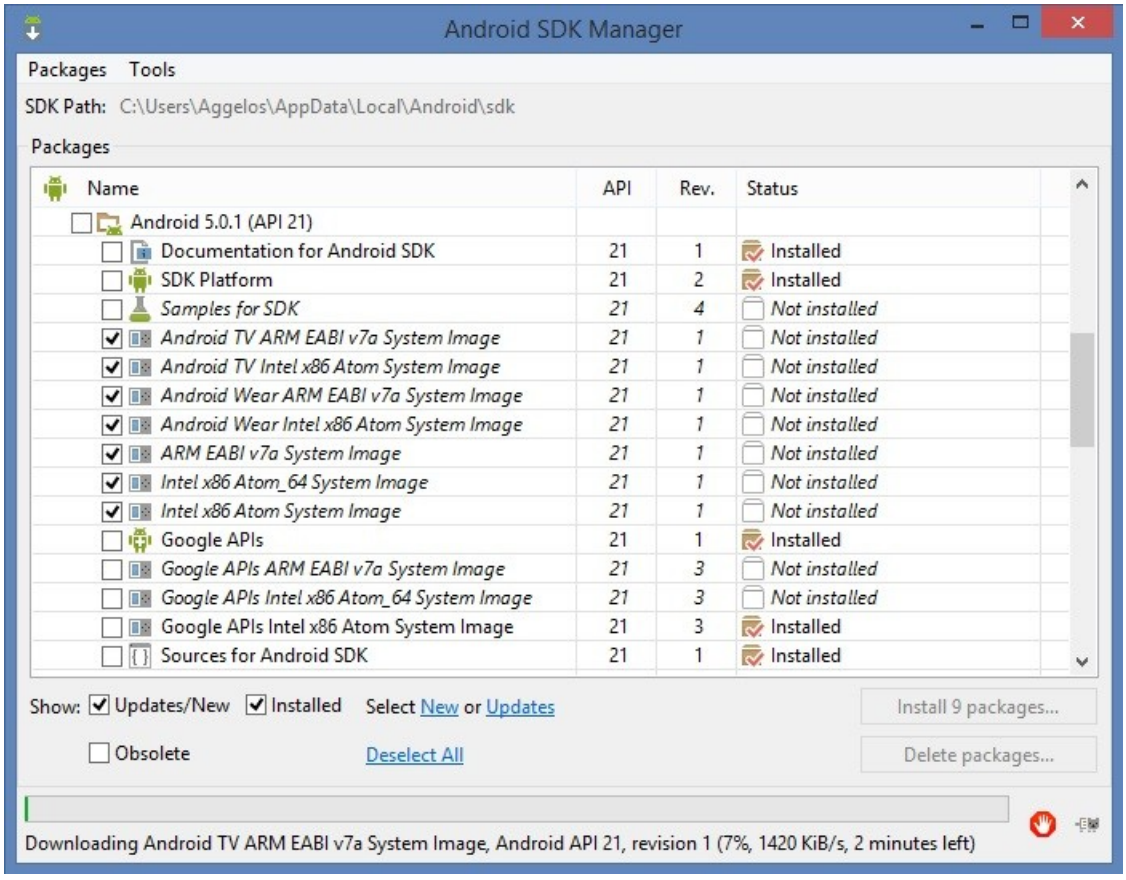
Η πλατφόρμα είναι διαθέσιμη μέσω της ιστοσελίδας http://developer.android.com/sdk/index.html και περιέχει όλα όσα χρειαζόμαστε για την δημιουργία εφαρμογών για το Android, συμπεριλαμβάνοντας το Android Studio IDE και τα εργαλεία Android Studio SDK. Αφού κατεβάσουμε και τρέξουμε το αρχείο εγκατάστασης του προγράμματος μπορούμε να ξεκινήσουμε την εγκατάσταση. Θα μας ζητηθεί να επιλέξουμε ποια χαρακτηριστικά θέλουμε να εγκατασταθούν καθώς και να επιλέξουμε τον τόπο εγκατάστασης για το πρόγραμμα. Μετά το τέλος της εγκατάστασης έχουμε την επιλογή να εισάγουμε τις ρυθμίσεις μας σε περίπτωση που ήδη χρησιμοποιούσαμε κάποια παλιότερη έκδοση του προγράμματος.

Οθόνη εγκατάστασης του Android Studio 1.0

Ένα σημαντικό βήμα πριν να ξεκινήσουμε την ανάπτυξη της εφαρμογής μας είναι να ελέγξουμε ποια πακέτα του Android SDK έχουμε εγκαταστήσει έτσι ώστε να σιγουρευτούμε ότι είμαστε έτοιμοι και δεν μας λείπει κάποιο σημαντικό πακέτο. Για να το κάνουμε αυτό πρέπει να χρησιμοποιήσουμε το εργαλείο Android SDK Manager στο οποίο έχουμε πρόσβαση από την επιλογή Διαμόρφωση – SDK Manager μέσα από την οθόνη καλωσορίσματος του προγράμματος. Μάλιστα το εργαλείο έχει επιλογή να ειδοποιούμαστε για νέες ενημερώσεις των πακέτων που χρησιμοποιούμε έτσι ώστε το πρόγραμμα να συμβαδίζει πάντα με την εξέλιξη του



λογισμικού του Android και να μας προσφέρει όσο το δυνατόν γρηγορότερα και πιο εύχρηστα νέες επιλογές για την ανάπτυξη των εφαρμογών μας.



Ο SDK Manager του Android Studio

2.4) Το σύστημα διαχείρισης βάσεων δεδομένων MySQL και η SQL

Πριν από την εμφάνιση της MySQL, η υλοποίηση μιας βάσης δεδομένων ήταν συνήθως ένα πολύπλοκο και ακριβό έργο που περιελάμβανε την αγορά,

εγκατάσταση και συντήρηση ενός ιδιόκτητου συστήματος διαχείρισης βάσης δεδομένων από έναν προμηθευτή, όπως η Oracle ή η IBM. Σε αντίθεση, η MySQL προσφέρει ένα δωρεάν, ανοικτού λογισμικού σύστημα διαχείρισης βάσεων δεδομένων που είναι εύκολο στην εγκατάσταση, εφαρμογή και συντήρηση. Επιπλέον, η MySQL είναι γρήγορη, εξαιρετικά αξιόπιστη και χρησιμοποιείται από πολλές εταιρείες και οργανισμούς σε όλο τον κόσμο. Όπως και η Java, η MySQL αναπτύσσεται διανέμεται και υποστηρίζεται από την Oracle. Η εφαρμογή μας απαιτεί πρόσβαση, εισαγωγή, ενημέρωση, αποθήκευση και διαγραφή δεδομένων καθώς όντας ένα σύστημα κράτησης εισιτηρίων χρειαζόμαστε πληθώρα δεδομένων να βρίσκονται αποθηκευμένα κάπου και να έχουμε εύκολη πρόσβαση σε αυτά.

Γίνεται λοιπόν σαφές ότι είναι αναγκαία η ύπαρξη μιας βάσης δεδομένων και ένα σύστημα για την διαχείριση των δεδομένων της εφαρμογής. Η MySQL μας δίνει αυτή τη δυνατότητα. Οι βάσεις δεδομένων σε MySQL είναι σχεσιακές, πράγμα που είναι ζωτικής σημασίας για μια εφαρμογή όπως η δική μας. Μπορούμε λοιπόν να αποθηκεύσουμε δεδομένα σε ξεχωριστούς πίνακες αντί να βάλουμε όλα τα δεδομένα σε ένα μεγάλο χώρο αποθήκευσης. Οι δομές δεδομένων οργανώνονται σε φυσικά αρχεία βελτιστοποιημένα για μεγαλύτερη ταχύτητα πρόσβασης και επεξεργασίας, ενώ το λογικό μοντέλο με αντικείμενα όπως βάσεις δεδομένων, πίνακες, γραμμές και στήλες προσφέρει ένα ευέλικτο προγραμματιστικό περιβάλλον. Χρησιμοποιούμε κανόνες οι οποίοι διέπουν τις σχέσεις μεταξύ διαφορετικών δεδομένων όπως *ένα προς ένα, ένα προς πολλά, μοναδικό* καθώς και δείκτες μεταξύ διαφορετικών πινάκων. Σωστή χρήση των κανόνων αυτών και των σχέσεων μεταξύ των δεδομένων εγγυώνται πως η βάση δεδομένων μας θα είναι εύρωστη, εύχρηστη και συνεπής. Θα αποφύγουμε έτσι ασυνέπειες, διπλοεγγραφές ή ελλείψεις στα δεδομένα.

Το κύριο πρόγραμμα, το οποίο και κάνει το μεγαλύτερο μέρος της δουλειάς σε μια εγκατάσταση της MySQL είναι ο MySQL server ή αλλιώς mysqld. Μέσω αυτού έχουμε πρόσβαση στον κατάλογο δεδομένων που περιέχει βάσεις και πίνακες. Εκεί βρίσκονται επίσης τα αρχεία καταγραφής και τα αρχεία κατάστασης του εξυπηρετητή. Στην δική μας περίπτωση, κατά την έναρξη του ο εξυπηρετητής της MySQL «ακούει» για απόπειρες σύνδεσης της εφαρμογής μας μαζί του και διαχειρίζεται την πρόσβαση στην βάση μας. Μέσω αυτού γίνονται όλες οι λειτουργίες των δεδομένων. Το εργαλείο που θα χρησιμοποιήσουμε για να χειριστούμε τον εξυπηρετητή είναι το phpMyAdmin, ένα ελεύθερο λογισμικό φτιαγμένο σε PHP για την διαχείριση MySQL servers μέσω του διαδικτύου. Μπορεί να δημιουργήσει και να διαγράψει βάσεις, να δημιουργήσει, μεταβάλει και διαγράψει πίνακες, να προσθέσει, επεξεργαστεί και διαγράψει πεδία, να διαχειριστεί κύρια και δευτερεύοντα κλειδιά, να ενεργοποιήσει και διαχειριστεί αποθηκευμένες διαδικασίες καθώς και να εξάγει δεδομένα σε διάφορες μορφές. Η αλληλεπίδραση γίνεται μέσω ενός περιηγητή διαδικτύου, ενώ αναγκαία είναι και η ύπαρξη έκδοσης PHP 5.3.0 η νεώτερης στον υπολογιστή. Επίσης θα χρειαστούμε

την επέκταση Standard PHP Library (SPL), υποστήριξη JSON και την επέκταση mbstring.

Τέλος πολύ σημαντικό ρόλο για την επιλογή της MySQL ως του συστήματος που θα χρησιμοποιήσουμε παίζει το γεγονός πως η MySQL είναι λογισμικό ανοιχτού κώδικα. Αυτό σημαίνει πως ο καθένας μπορεί να χρησιμοποιήσει και να τροποποιήσει τον πηγαίο κώδικα του λογισμικού ανάλογα με τις ανάγκες του. Το λογισμικό της MySQL χρησιμοποιεί το Generic Public License(GPL) στο οποίο και μπορεί κανείς να δει τι μπορεί ή όχι να κάνει με το λογισμικό σε διάφορες καταστάσεις.

Η γλώσσα που μας δίνει όλες αυτές τις δυνατότητες είναι η Structured Query Language (SQL). Είναι η πιο κοινή τυποποιημένη γλώσσα που χρησιμοποιείται για πρόσβαση σε βάσεις δεδομένων. Ξεκίνησε να αναπτύσσεται από την IBM στις αρχές της δεκαετίας του 70. Από τότε έχει τροποποιηθεί σημαντικά όμως ο σκοπός της ως μια γλώσσα διαχείρισης βάσεων δεδομένων παραμένει ο ίδιος. Η SQL υποστηρίζεται από σχεδόν όλα τα συστήματα διαχείρισης βάσεων δεδομένων καθώς δεν είναι αποκλειστική ιδιοκτησία κάποιας εταιρίας, πράγμα που την κάνει πολύ προσφιλή καθώς μαθαίνοντας την χρήση της για μια εφαρμογή σημαίνει ότι μπορεί κανείς να χρησιμοποιήσει τις γνώσεις του και σε πολλές άλλες. Ένα ακόμη σημαντικό πλεονέκτημα της SQL είναι πως σε σχέση με άλλες προγραμματιστικές γλώσσες είναι αρκετά απλή στην εκμάθηση της. Παρόλη την απλότητα της όμως παραμένει μια πολύ ισχυρή γλώσσα μέσω της οποίας μπορεί κανείς να επιτύχει μέγιστα αποτελέσματα με ελάχιστη προσπάθεια.

2.5) Η χρήση της PHP ως ενδιάμεσου μεταξύ της εφαρμογής και της βάσης δεδομένων

Η γλώσσα προγραμματισμού PHP σχεδιάστηκε με στόχο την ανάπτυξη-διαχείριση ιστοσελίδων χρησιμοποιείται όμως πλέον και ως μια γενική γλώσσα προγραμματισμού. Στην εφαρμογή μας η PHP χρησιμοποιείται για να παρέχει πρόσβαση στα δεδομένα της βάσης του MySQL server. Αυτό επιτυγχάνεται ενσωματώνοντας εντολές SQL σε κώδικα PHP. Έτσι μέρος του περιεχομένου της εφαρμογής παράγεται από τις πληροφορίες που εξάγουμε από την βάση δεδομένων. Αναλυτικότερα για να αποκτήσουμε πρόσβαση στο περιεχόμενο της βάσης δεδομένων μας πρέπει να δημιουργήσουμε μια σύνδεση στην MySQL μέσω ενός PHP Script. Σε επόμενο κεφάλαιο θα αναλύσουμε βήμα προς βήμα την διαδικασία μέσω της οποίας γίνεται η επικοινωνία εφαρμογής-βάσης δεδομένων που περιλαμβάνει τη σύνδεση-αποσύνδεση με τη βάση, την διαδικασία εκτέλεσης ερωτημάτων και λήψης αποτελεσμάτων καθώς και την διαδικασία μετατροπής των αποτελεσμάτων σε JSON strings τα οποία αποστέλλουμε στην εφαρμογή.

Συνοψίζοντας για την δημιουργία της εφαρμογής μας τα εργαλεία που χρησιμοποιήσαμε ήταν: Α) το Android Studio για την ανάπτυξη του κώδικα και του περιβάλλοντος της εφαρμογής, Β) PHP Scripts για την επικοινωνία της εφαρμογής με τον MySQL server, Γ)Το λογισμικό PhpMyAdmin για την διαχείριση του MySQL

server, Δ)MySQL server για την αποθήκευση των δεδομένων της εφαρμογής. Παρακάτω θα μελετήσουμε την διαδικασία ανάπτυξης της εφαρμογής, από το κομμάτι του σχεδιασμού με τις απαιτήσεις για την δημιουργία ενός εύχρηστου και εύρωστου συστήματος λογισμικού, μέχρι και το κομμάτι της υλοποίησης με την δημιουργία του κώδικα.

3) Το σύστημα κράτησης εισιτηρίων κινηματογράφου ως έργο λογισμικού

Η πραγματοποίηση μιας διαδικασίας ανάπτυξης λογισμικού διαφέρει σε σημαντικό βαθμό από την ανάπτυξη ενός καλού και σωστού λογισμικού. Η ανάπτυξη καλά δομημένων συστημάτων λογισμικού προϋποθέτει την εφαρμογή τεχνικών σχεδίασης. Στο κεφάλαιο αυτό θα αναλύσουμε την ανάπτυξη της εφαρμογής στην θεωρία ως έργο λογισμικού και θα μελετήσουμε τις τεχνικές που χρησιμοποιήσαμε κατά την διάρκεια εξαγωγής και ανάλυσης των απαιτήσεων του συστήματος, τις υπηρεσίες δηλαδή που θα πρέπει να παρέχει το σύστημα μας καθώς και τους λειτουργικούς περιορισμούς του. Επίσης θα χρησιμοποιήσουμε τεχνικές όπως η δημιουργία περιπτώσεων χρήσης και τα διαγράμματα δραστηριοτήτων, ακολουθίας και συνεργασίας για την τεκμηρίωση του συστήματος καθώς και για τη διερεύνηση διαφόρων σεναρίων χρήσης του συστήματος με σκοπό τον εντοπισμό των λειτουργιών κάθε αντικειμένου.

3.1) Προσδιορισμός των απαιτήσεων του συστήματος

Το σύστημα μας όπως και κάθε άλλο έχει έναν σκοπό, ο οποίος εκφράζεται με τις δυνατότητες του, οι οποίες διατυπώνονται ως απαιτήσεις, χαρακτηριστικά δηλαδή του συστήματος τα οποία πρέπει να ικανοποιεί ώστε να εκπληρώσει τον σκοπό του. Υπάρχουν πέντε βασικές τεχνικές για την εξαγωγή των απαιτήσεων. Στην περίπτωση μας χρησιμοποιήσαμε δυο εξ αυτών, τις συνεντεύξεις και την επί τόπου παρακολούθηση προκειμένου να βρούμε τις απαιτήσεις των χρηστών, απευθυνθήκαμε δηλαδή σε μελλοντικούς χρήστες και τους ρωτήσαμε τι επιλογές θα επιθυμούσαν να τους προσφέρει ένα online σύστημα κράτησης εισιτηρίων. Επίσης δημιουργήσαμε ένα πρωτότυπο το οποίο διανείμαμε σε μια ομάδα χρηστών προκειμένου να βρούμε σημεία στα οποία οι χρήστες θα μπορούσαν να αντιμετωπίσουν δυσκολίες και τα οποία θα χρειάζονταν βελτίωση, τυχόν ασυνέπειες στο πρόγραμμα και τις λειτουργίες του, ελλείψεις καθώς και για να ακούσουμε προτάσεις για επιπλέον χαρακτηριστικά που θα ήταν επιθυμητά. Η διαδικασία αυτή διήρκησε περίπου έναν μήνα και μέσω αυτής καταλήξαμε σε ένα σύνολο χαρακτηριστικών τα οποία και ενσωματώσαμε στο σύστημα μας. Μια σύντομη περιγραφή των χαρακτηριστικών αυτών δίνεται παρακάτω.

Θα πρέπει να πούμε στο σημείο αυτό πως αντιμετωπίσαμε ορισμένα προβλήματα καθώς οι πιο πρόσφατες εκδόσεις του Android έχουν καταστήσει απαρχαιωμένες ορισμένες τεχνικές που χρησιμοποιούσαν οι προγραμματιστές προκειμένου να δημιουργήσουν περιεχόμενο στις εφαρμογές τους και επίσης ορισμένες από τις νέες τεχνολογίες που έχουν αναπτυχθεί δεν υποστηρίζονται από

παλαιότερες εκδόσεις του λογισμικού. Με γνώμονα λοιπόν την διαλειτουργικότητα του συστήματος ενσωματώσαμε στο σύστημά μας χαρακτηριστικά που το καθιστούν συμβατό με όλες τις εκδόσεις του Android από την 2.3.7 και έπειτα.

Το σύστημα μας όπως γίνεται κατανοητό και από τον τίτλο του αποτελεί ένα πρόγραμμα για την κράτηση εισιτηρίων σε κινηματογράφο μέσω internet. Ο χρήστης λοιπόν του συστήματος θα πρέπει να μπορεί να επιλέξει μέσα από ένα σύνολο ταινιών που προβάλλονται σε έναν κινηματογράφο την ταινία που επιθυμεί να παρακολουθήσει. Η διαδικασία αυτή πραγματοποιείται με την επιλογή ταινίας από μια λίστα η οποία εμφανίζεται στον χρήστη με το άνοιγμα του προγράμματος. Ήταν πολύ σημαντικό λοιπόν να εξασφαλίσουμε την συμβατότητα του προγράμματος με τις διάφορες συσκευές που κυκλοφορούν στο εμπόριο. Όπως είδαμε στο πρώτο κεφάλαιο το λειτουργικό σύστημα Android έχει τεράστιο αριθμό συσκευών που το χρησιμοποιούν από επίσης πολύ μεγάλο αριθμό κατασκευαστών. Αυτό μεταφράζεται σε πληθώρα συσκευών με διαφορετικά χαρακτηριστικά, είτε αυτά έχουν να κάνουν με την οθόνη της συσκευής είτε με την μνήμη, τον επεξεργαστή κλπ. Προσπαθήσαμε να εξασφαλίσουμε πως ανεξαρτήτου έκδοσης του λογισμικού και εύρους οθόνης που έχει η συσκευή του χρήστη το σύστημα θα ήταν εύχρηστο και με φιλικό προς τον χρήστη περιβάλλον όσο αυτό ήταν δυνατό έχοντας παράλληλα υπ όψιν μας πως θα πρέπει το σύστημα μας να συμβαδίζει με τις νέες τεχνολογίες. Έτσι το μέγεθος της λίστας των διαθέσιμων προς παρακολούθηση ταινιών προσαρμόζεται ανάλογα με το μέγεθος οθόνης της συσκευής που τρέχει την εφαρμογή και λειτουργεί τόσο σε έξυπνα τηλέφωνα όσο και σε ταμπλέτες. Η διεύθυνση του κινηματογράφου επιθυμεί να προβάλει το προϊόν της(ταινίες) όσο καλύτερα γίνεται και παράλληλα οι πελάτες επιθυμούν να είναι ενημερωμένοι, να μπορούν να αποκτήσουν πληροφορίες για τις ταινίες εύκολα και γρήγορα. Για να ικανοποιήσουμε την απαίτηση αυτή φτιάξαμε την εφαρμογή μας έτσι ώστε όταν ο χρήστης επιλέξει μια ταινία να μπορεί να δει επιπλέον χαρακτηριστικά της, όπως το είδος, την πλοκή, τους ηθοποιούς, η να παρακολουθήσει το trailer της η κάποιες εικόνες προκειμένου να αποφασίσει εάν θα τον ενδιέφερε να την παρακολουθήσει η όχι. Κάθε φορά που ο χρήστης επιλέγει κάποια ταινία από την λίστα της αρχικής οθόνης εμφανίζεται μια νέα λίστα με την σύνοψη όλων των παραπάνω χαρακτηριστικών την οποία ο χρήστης μπορεί να επεκτείνει για περαιτέρω πληροφορίες. Από αυτή τη νέα λίστα μπορεί να επιλέξει και τον κινηματογράφο και την προβολή που τον ενδιαφέρει. Η εφαρμογή υποστηρίζει την ύπαρξη πολλών κινηματογράφων που προβάλλουν πολλές ταινίες σε ένα σύνολο από αίθουσες. Αυτό ικανοποιεί την απαίτηση μιας εταιρίας κινηματογράφων που επιθυμεί να έχει ένα σύστημα το οποίο περιλαμβάνει όλους της τους κινηματογράφους οι οποίοι θα ενημερώνονται όλοι μαζί με βάση ένα εβδομαδιαίο πρόγραμμα. Επίσης οι χρήστες έτσι μπορούν να διαλέξουν τον κινηματογράφο της αρεσκείας τους, αυτόν δηλαδή που ανταποκρίνεται καλύτερα στις ανάγκες τους. Ένας χρήστης μπορεί να θέλει να παρακολουθήσει μια ταινία αλλά παράλληλα να βρίσκεται το δυνατόν πιο κοντά στο σπίτι του, ή μπορεί να θέλει να παρακολουθήσει την ταινία της επιλογής του σε

πολύ μεγάλη οθόνη με αντίστοιχα καλό ηχοσύστημα, ή ακόμη η ταινία που θέλει να παρακολουθήσει να προβάλλεται σε 3D. Ακόμη ο χρήστης μπορεί να επιλέξει την ημερομηνία και την ώρα της προβολής που τον ενδιαφέρει. Αφού ο χρήστης διαλέξει κινηματογράφο, αίθουσα και ώρα προβολής έχει τη δυνατότητα να επιλέξει πόσες θέσεις θέλει να κρατήσει καθώς και το σημείο στην αίθουσα που υπάρχουν διαθέσιμες θέσεις. Δίνεται έτσι η δυνατότητα στον χρήστη χρησιμοποιώντας την εφαρμογή του κινητού του να δει εάν υπάρχουν διαθέσιμες θέσεις στην προβολή που τον ενδιαφέρει και εφόσον αυτό είναι εφικτό να κρατήσει καλές θέσεις. Με αυτόν τον τρόπο δεν χρειάζεται επικοινωνία μεταξύ κινηματογράφου και χρήστη, η διαδικασία αυτοματοποιείται, ο χρήστης μπορεί ανά πάσα στιγμή να γνωρίζει την διαθεσιμότητα η όχι εισιτηρίων για την προβολή που τον ενδιαφέρει. Μια άλλη απαίτηση των χρηστών είναι να τους δίνονται πολλές επιλογές αναφορικά με τον τρόπο πληρωμής των εισιτηρίων που έχουν κρατήσει. Έτσι οι πελάτες μπορούν κάνοντας μια κράτηση να πληρώσουν απευθείας μέσω πιστωτικής κάρτας ενώ υπάρχει και η επιλογή να λάβουν έναν αριθμό κράτησης και να μπορούν να πληρώσουν στο ταμείο πριν την έναρξη της προβολής. Τέλος οι χρήστες θέλουν να μπορούν να δουν τις κρατήσεις τους, τόσο αυτές που είναι ενεργές, έτσι ώστε να μην χρειάζεται να θυμούνται πότε είναι η κράτηση τους αλλά και αυτές που έχουν παρέλθει, κρατώντας ιστορικό των ταινιών που έχουν παρακολουθήσει. Μπορούν να δημιουργήσουν προσωπικό λογαριασμό στον οποίο θα διατηρείται το ιστορικό των κρατήσεων τους ή να χρησιμοποιήσουν τον λογαριασμό τους του facebook. Παράλληλα θέλοντας να εξασφαλίσουμε και την επιθυμία του κινηματογράφου να μην μένουν θέσεις στις προβολές αχρησιμοποίητες επειδή κάποιος χρήστης δεν χρησιμοποίησε τις κρατημένες θέσεις του και ακολουθώντας την πολιτική των κινηματογράφων κάναμε τις κρατήσεις της εφαρμογής μας έτσι ώστε οι κρατήσεις των χρηστών να ισχύουν μέχρι κάποια συγκεκριμένη ώρα , η οποία αν παρέλθει η κράτηση ακυρώνεται ελευθερώνοντας τις αντίστοιχες θέσεις. Μια απαίτηση των χρηστών την οποία αναφέραμε και προηγουμένως είναι να μπορούν να επιλέξουν τον κινηματογράφο της αρεσκείας τους βάση της απόστασης τους από αυτόν. Θέλοντας να απευθυνθούμε στην απαίτηση αυτή των χρηστών ενσωματώσαμε στην εφαρμογή μας την πλατφόρμα Google Maps μέσω της οποίας οι πελάτες μπορούν να βρουν την τοποθεσία κάθε κινηματογράφου που υποστηρίζεται από την εφαρμογή μέσω κουμπιών τα οποία τους κατευθύνουν απευθείας στη θέση του κινηματογράφου τον οποίο έχουν επιλέξει. Μια άλλη απαίτηση των χρηστών ήταν να είναι διαθέσιμη η εφαρμογή και στα αγγλικά. Έτσι ενσωματώσαμε την δυνατότητα επιλογής γλώσσας προβολής για τον χρήστη στην οποία ο χρήστης έχει πρόσβαση μέσα από την εφαρμογή. Λάβαμε υπ όψιν μας την επιθυμία των χρηστών για την ύπαρξη μιας μοναδικής εφαρμογής με δυνατότητες προσαρμογής αντί για την δημιουργία ξεχωριστής εφαρμογής ανάλογα με την γλώσσα που επιθυμεί να χρησιμοποιεί ο χρήστης. Τέλος αναγνωρίζοντας πως η εποχή μας είναι η εποχή των μέσων κοινωνικής δικτύωσης ενσωματώσαμε την δυνατότητα για τους χρήστες να μοιραστούν μέσω facebook με τους φίλους τους την ταινία που θα

παρακολουθήσουν – παρακολούθησαν και να σχολιάσουν στην δημοσίευση τους τις απόψεις – κριτικές τους σχετικά με την ταινία.

3.2)Σενάρια χρήσης της εφαρμογής

Στην προσπάθεια μας να αποκτήσουμε μια σαφέστερη εικόνα του τρόπου χρήσης της εφαρμογής μας δημιουργήσαμε έναν αριθμό σεναρίων μέσω των οποίων προσομοιώσαμε την διαδικασία αλληλεπίδρασης ενός χρήστη με την εφαρμογή σε διάφορες καταστάσεις. Τα σενάρια μας περιλαμβάνουν την περιγραφή της αρχικής κατάστασης, την κανονική ροή των γεγονότων, μια περιγραφή πιθανών προβλημάτων, καθώς και την περιγραφή της κατάστασης ολοκλήρωσης του σεναρίου. Έπειτα βάση των σεναρίων θα δημιουργήσουμε τις περιπτώσεις χρήσης που θα προσδιορίσουν τους συμμετέχοντες που συμμετέχουν και περιγράφουν μια αλληλεπίδραση.

3.2.1) Σενάριο κράτησης εισιτηρίου

Αρχική παραδοχή: Ο χρήστης έχει αποφασίσει ποια ταινία θέλει να παρακολουθήσει και είναι συνδεδεμένος στο ίντερνετ.

Κανονική ροή: Ο χρήστης ανοίγει την εφαρμογή στην συσκευή του. Αυτή συνδέεται με την βάση δεδομένων στον MySQL Server και κατεβάζει στο κινητό του χρήστη μια λίστα με τις διαθέσιμες ταινίες. Ο χρήστης επιλέγει την ταινία που επιθυμεί από την λίστα. Η εφαρμογή εμφανίζει μια νέα καρτέλα με τις λεπτομέρειες της ταινίας καθώς και επιλογές για κινηματογράφο, ημερομηνίες και ώρες προβολής. Ο χρήστης επιλέγει την ημερομηνία που επιθυμεί, τον κινηματογράφο και την αίθουσα. Η εφαρμογή στο σημείο αυτό εμφανίζει στον χρήστη τις διαθέσιμες θέσεις με λευκό χρώμα, τις κρατημένες με κίτρινο ενώ κάθε θέση που επιλέγει ο χρήστης μαρκάρεται με πράσινο χρώμα. Ο χρήστης επιλέγει τις θέσεις που επιθυμεί και προχωρά στην κράτηση τους. Η εφαρμογή εμφανίζει μια νέα καρτέλα με τις λεπτομέρειες της κράτησης του χρήστη (ταινία, αριθμό εισιτηρίων, κόστος, ώρα προβολής και αίθουσα) και δίνει στον χρήστη δυο δυνατότητες πληρωμής, είτε με πιστωτική κάρτα ή με πληρωμή απευθείας στο ταμείο. Ο χρήστης επιλέγει τρόπο πληρωμής και προχωρά σε επικύρωση της κράτησης του. Ενημερώνεται η βάση δεδομένων της εφαρμογής με την κράτηση του χρήστη και τον επιστρέφει στην αρχική καρτέλα.

Πιθανά προβλήματα: Η προβολή που έχει επιλέξει ο χρήστης δεν έχει αρκετές διαθέσιμες θέσεις. Στην περίπτωση αυτή ο χρήστης θα πρέπει να επιλέξει μια νέα προβολή. Ο χρήστης έχει επιλέξει για τρόπο πληρωμής την πιστωτική κάρτα τα στοιχεία της οποίας όμως δεν επικυρώνονται από το σύστημα. Η εφαρμογή θα πρέπει να εμφανίσει μήνυμα απόρριψης της κάρτας του χρήστη και να ζητήσει νέο αριθμό ή εναλλακτικό τρόπο πληρωμής. Η εφαρμογή μπορεί να αποτύχει να ενημερώσει την βάση του με την κράτηση του χρήστη. Στην περίπτωση αυτή θα πρέπει ο χρήστης να ενημερωθεί με κατάλληλο μήνυμα ώστε να μπορέσει να κάνει εκ νέου την κράτηση του.

Κατάσταση του συστήματος: Ο χρήστης παραμένει συνδεδεμένος στην

εφαρμογή. Σε περίπτωση επιτυχούς κράτησης το ιστορικό κρατήσεων ενημερώνεται με την τελευταία κράτηση του χρήστη._

Με βάση το σενάριο κράτησης εισιτηρίου θα δημιουργήσουμε την **περίπτωση χρήσης κράτησης εισιτηρίου**.

Σύντομη περιγραφή: Ο χρήστης χρησιμοποιεί την εφαρμογή προκειμένου να κάνει μια κράτηση για την ταινία που επιθυμεί να παρακολουθήσει.

Κύριος χρήστης: Πελάτης κινηματογράφου.

Εμπλεκόμενοι και ενδιαφέροντα: α)Χρήστης: θέλει εύχρηστο περιβάλλον, επιλογές αναφορικά με το ποιες ταινίες μπορεί να παρακολουθήσει και πότε, εξυπηρέτηση με τον ελάχιστο δυνατό κόπο. Επιπλέον θέλει απόδειξη για την κράτηση που θα πραγματοποιήσει. β) Εταιρία κινηματογράφου: Θέλει να καταγράφονται οι κρατήσεις που πραγματοποιούνται με ακρίβεια στην βάση, αυτόματη και γρήγορη ενημέρωση της βάσης προκειμένου να αποφεύγονται σφάλματα στις κρατήσεις.

Προϋποθέσεις: Ο χρήστης είναι συνδεδεμένος στο διαδίκτυο και αναγνωρίζεται είτε μέσω της συσκευής του είτε σε περίπτωση που έχει κάνει εγγραφή μέσω του ονόματος χρήστη του.

Εγγύηση Επιτυχίας: Η κράτηση αποθηκεύεται στην βάση που ενημερώνεται για τις κρατημένες θέσεις και σε ποια προβολή αντιστοιχούν. Εκδίδεται αποδεικτικό κράτησης ανάλογα με τον τρόπο πληρωμής που έχει επιλέξει ο χρήστης.

Κύριο Σενάριο Επιτυχίας:

1. Ο χρήστης ανοίγει την εφαρμογή προκειμένου να πραγματοποιήσει μια κράτηση.
2. Η εφαρμογή εμφανίζει στον χρήστη μια λίστα με τις διαθέσιμες ταινίες.
3. Ο χρήστης επιλέγει από την λίστα με τις διαθέσιμες προς παρακολούθηση ταινίες αυτή που επιθυμεί.
4. Η εφαρμογή παρουσιάζει στον χρήστη τα χαρακτηριστικά της ταινίας και του δίνει επιλογές για τον κινηματογράφο την ημερομηνία προβολής, την αίθουσα και την ώρα προβολής της ταινίας.
5. Ο χρήστης επιλέγει την προβολή που ανταποκρίνεται στις επιθυμίες του.
6. Το σύστημα εμφανίζει ένα σχεδιάγραμμα της αίθουσας στο οποίο καλεί τον χρήστη να επιλέξει τις θέσεις που επιθυμεί να κρατήσει.
7. Ο χρήστης επιλέγει τις θέσεις που τον ικανοποιούν
8. Ο χρήστης προχωρά στην κράτηση των θέσεων

9. Η εφαρμογή εμφανίζει μια καρτέλα με πληροφορίες αναφορικά με την κράτηση του χρήστη (ταινία, αριθμό εισιτηρίων, κόστος, ώρα προβολής και αίθουσα) και τις επιλογές που έχει ο χρήστης για να πληρώσει.

10. Ο χρήστης επιλέγει τρόπο πληρωμής και οριστικοποιεί την κράτηση

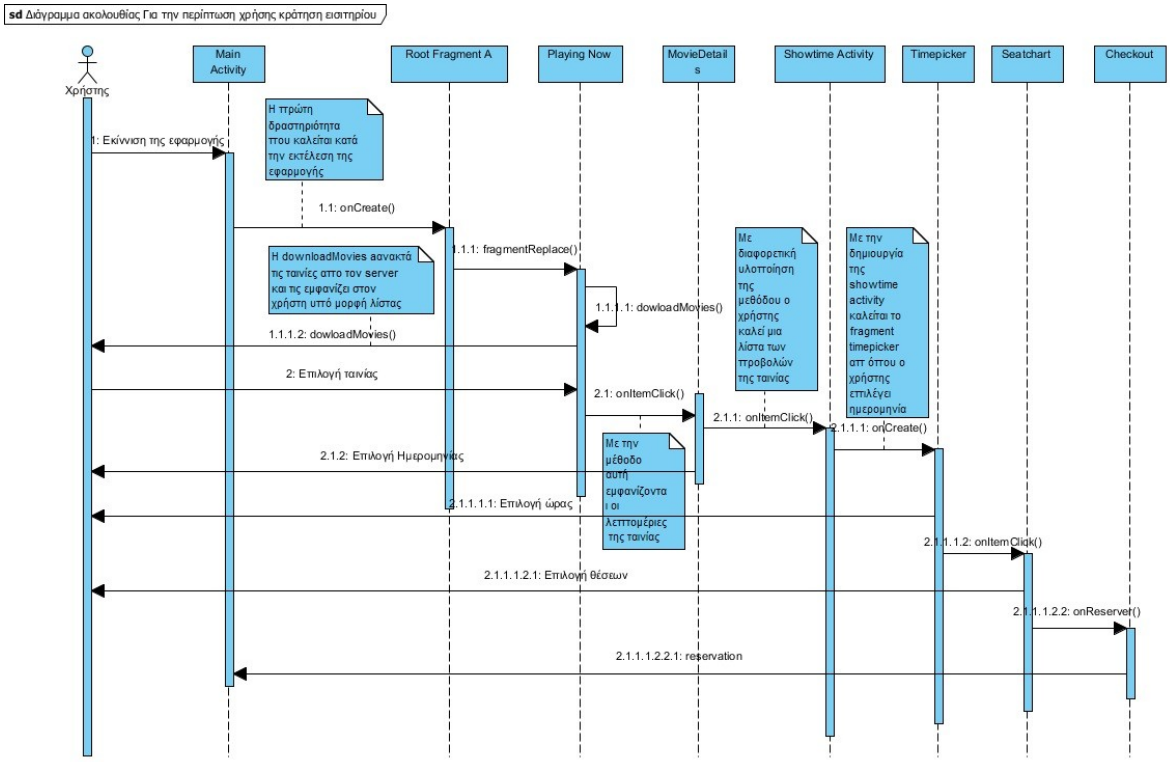
11. Η εφαρμογή εμφανίζει μήνυμα πως η κράτηση καταχωρήθηκε και εμφανίζει την οθόνη ιστορικού κρατήσεων.

12. Ο χρήστης μπορεί να ελέγξει την κράτηση του η να ξεκινήσει κάποια άλλη διαδικασία

    Επεκτάσεις:

    Α)Η εφαρμογή αποτυγχάνει σε κάποιο βήμα: Εμφανίζεται στον χρήστη μήνυμα σφάλματος και η εφαρμογή τερματίζεται. Ο χρήστης επανεκκινεί την εφαρμογή και ξεκινά εκ νέου τη διαδικασία κράτησης εισιτηρίου.
    2.α)Η εφαρμογή δεν εμφανίζει στον χρήστη την λίστα με τις ταινίες που παίζουν αυτή τη στιγμή στους κινηματογράφους. Εμφανίζεται στον χρήστη μήνυμα αδυναμίας φόρτωσης της λίστας και προτροπή να επαναλάβει την φόρτωση.
    6.α1)Η προβολή την οποία έχει επιλέξει ο χρήστης δεν έχει διαθέσιμες θέσεις
    6.α2)Ο χρήστης ακυρώνει την κράτηση και επιλέγει μια νέα προβολή για να παρακολουθήσει.
    9.α1)Οι λεπτομέρειες που εμφανίζει η εφαρμογή σχετικά με την κράτηση του χρήστη δεν τον ικανοποιούν (μπορεί ο χρήστης να επέλεξε λάθος αριθμό θέσεων ή να μην συμφωνεί με την τιμή των θέσεων που έχει επιλέξει)
    9.α2)Ο χρήστης ακυρώνει την κράτηση του επιλέγοντας μια άλλη καρτέλα η επιστρέφοντας στην αρχική της εφαρμογής.
    10.α1)Ο χρήστης αποφασίζει πως θέλει να πληρώσει με χρήση της πιστωτικής του κάρτας αντί με μετρητά στο ταμείο του κινηματογράφου.
    10.α2)Η εφαρμογή εμφανίζει ένα μενού στο οποίο ο χρήστης καλείται να εισάγει τα στοιχεία της κάρτας του(αριθμό, ημ. λήξης, ονοματεπώνυμο,cvv).

Για να κατανοήσουμε καλύτερα την αλληλεπίδραση μεταξύ του χρήστη και της εφαρμογής καθώς και των επιμέρους τμημάτων της εφαρμογής δημιουργήσαμε το διάγραμμα ακολουθίας που μας δείχνει τα μηνύματα που ανταλλάσουν τα αντικείμενα μεταξύ τους.



Διάγραμμα ακολουθίας της περίπτωσης χρήσης Κράτηση εισιτηρίου

3.2.2) Σενάριο Δημιουργίας λογαριασμού

Αρχική παραδοχή: Ο χρήστης επιθυμεί να δημιουργήσει έναν λογαριασμό στην εφαρμογή.

Κανονική ροή: Ο χρήστης ανοίγει την εφαρμογή στην συσκευή του. Αυτή του εμφανίζει την αρχική καρτέλα με την λίστα των προβαλλόμενων ταινιών και την επιλογή να μεταβεί σε κάποια άλλη. Ο χρήστης επιλέγει την καρτέλα «Λογαριασμός». Η εφαρμογή εμφανίζει την καρτέλα στην οποία ο χρήστης μπορεί να δημιουργήσει νέο λογαριασμό ή να κάνει είσοδο με λογαριασμό είτε της εφαρμογής είτε του Facebook. Ο χρήστης επιλέγει τη δημιουργία νέου λογαριασμού. Η εφαρμογή εμφανίζει μια νέα καρτέλα και καλεί τον χρήστη να εισάγει τα στοιχεία του. Ο χρήστης εισάγει τα στοιχεία του και πατά υποβολή. Η εφαρμογή στέλνει την αίτηση εγγραφής στη βάση και κατόπιν ενημερώνει τον χρήστη για την επιτυχή εγγραφή του. Ο χρήστης τώρα μπορεί να κάνει είσοδο στην εφαρμογή με τον λογαριασμό του.

Πιθανά προβλήματα: Το όνομα που θέλει να χρησιμοποιήσει ο χρήστης είναι δεσμευμένο. Η εφαρμογή θα πρέπει να ενημερώσει τον χρήστη για την μη διαθεσιμότητα του ονόματος και να τον προτρέψει να χρησιμοποιήσει κάποιο άλλο. Ο χρήστης έχει εισάγει λανθασμένη μορφή e-mail ή δεν έχει πληκτρολογήσει σωστά την επαλήθευση του κωδικού πρόσβασης. Η εφαρμογή θα πρέπει να

ενημερώσει τον χρήστη πως έχει εισάγει λανθασμένα στοιχεία και θα πρέπει να εισάγει μια αποδεκτή μορφή e-mail ή το ίδιο κωδικό με τον πρώτο στην επιβεβαίωση κωδικού πρόσβασης.
Κατάσταση του συστήματος: Ο χρήστης παραμένει συνδεδεμένος στην εφαρμογή. Το σύστημα έχει ενημερωθεί με την νέα εγγραφή και έχει αποθηκεύσει τα στοιχεία του χρήστη.

Με βάση το σενάριο αναζήτησης ταινίας θα δημιουργήσουμε την **περίπτωση χρήσης δημιουργίας λογαριασμού**.

Σύντομη περιγραφή: Ο χρήστης χρησιμοποιεί την εφαρμογή προκειμένου να δημιουργήσει ένα λογαριασμό μέσα από τον οποίο θα μπορεί να ενημερώνεται καλύτερα για τις κρατήσεις του.

Κύριος χρήστης: Πελάτης κινηματογράφου.

Εμπλεκόμενοι και ενδιαφέροντα: α)Χρήστης: θέλει εύχρηστο περιβάλλον, η διαδικασία δημιουργίας λογαριασμού να είναι απλή και κατανοητή, θέλει επιβεβαίωση πως ο λογαριασμός του δημιουργήθηκε και να μπορεί να αλλάξει της πληροφορίες του λογαριασμού του. β) Εταιρία κινηματογράφου: Θέλει η βάση δεδομένων να είναι πάντα ενημερωμένη με τους λογαριασμούς των χρηστών. Επίσης θέλει να είναι εξασφαλισμένη η ασφάλεια των λογαριασμών, να μην είναι δυνατό κάποιος να υποκλέψει τις πληροφορίες των χρηστών.

Προϋποθέσεις: Ο χρήστης είναι συνδεδεμένος στο διαδίκτυο. Ο MySQL Server λειτουργεί σωστά και δέχεται τα δεδομένα που εισάγει ο χρήστης στη φόρμα εγγραφής του.

Εγγύηση Επιτυχίας: Η βάση με τους λογαριασμούς των χρηστών έχει ενημερωθεί σωστά. Ο χρήστης μπορεί να πραγματοποιήσει είσοδο με τον λογαριασμό του στην εφαρμογή και να δημιουργήσει μια κράτηση που θα φαίνεται στο ιστορικό κρατήσεων του λογαριασμού του.

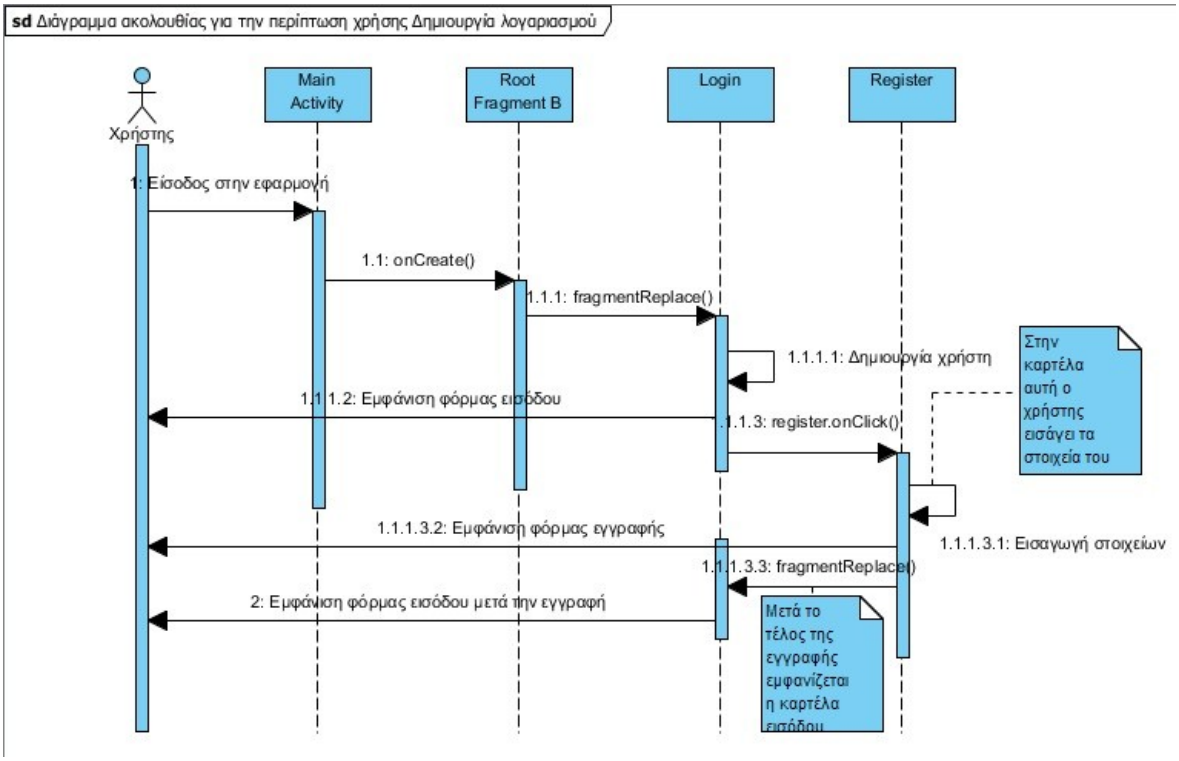
Κύριο Σενάριο Επιτυχίας:

1. Ο χρήστης ανοίγει την εφαρμογή προκειμένου να και θέλει να δημιουργήσει έναν καινούργιο λογαριασμό.
2. Η εφαρμογή εμφανίζει στον χρήστη την αρχική οθόνη με τις καρτέλες και την λίστα με τις προβαλλόμενες ταινίες.
3. Ο χρήστης μεταβαίνει στην καρτέλα με όνομα «λογαριασμός».
4. Η εφαρμογή προβάλλει την καρτέλα που δίνει στο χρήστη επιλογή εισόδου με υπάρχοντα λογαριασμό, είσοδο με λογαριασμό Facebook ή εγγραφή.
5. Ο χρήστης διαλέγει την επιλογή εγγραφή για να κάνει νέο λογαριασμό.

6. Η εφαρμογή εμφανίζει την καρτέλα εγγραφής νέου χρήστη και ζητά να εισάγει τα στοιχεία του (Ψευδώνυμο, κωδικό, ονοματεπώνυμο, e-mail).
7. Ο χρήστης εισάγει τα στοιχεία του και πατά εγγραφή.
8. Η εφαρμογή ενημερώνει την βάση για την δημιουργία νέου λογαριασμού και αποθηκεύει .
9. Η εφαρμογή εμφανίζει στον χρήστη μήνυμα επιτυχούς εγγραφής και τον επιστρέφει στην καρτέλα εισόδου χρήστη.

Επεκτάσεις:

Α)Η εφαρμογή αποτυγχάνει σε κάποιο βήμα: Εμφανίζεται στον χρήστη μήνυμα σφάλματος και η εφαρμογή τερματίζεται. Ο χρήστης επανεκκινεί την εφαρμογή και ξεκινά εκ νέου την περιήγηση του.
7.α) Ο χρήστης εισάγει όνομα που υπάρχει ήδη
7.α1) Η εφαρμογή ενημερώνει τον χρήστη πως το όνομα που επιθυμεί δεν είναι διαθέσιμο και τον προτρέπει να διαλέξει κάποιο άλλο.
7.β) Ο χρήστης εισάγει λανθασμένα την επιβεβαίωση κωδικού πρόσβασης.
7.β1) Η εφαρμογή ενημερώνει τον χρήστη για το λάθος και του ζητά να εισάγει και πάλι τον κωδικό και την επιβεβαίωση του.
7.γ) Ο χρήστης εισάγει μη έγκυρη μορφή e-mail.
7.γ1) Η εφαρμογή ενημερώνει τον χρήστη για την μορφή την οποία μπορεί να έχει μια διεύθυνση e-mail.
8) Η εφαρμογή αποτυγχάνει να ενημερώσει τη βάση για την δημιουργία νέου χρήστη. Ο χρήστης ενημερώνεται με μήνυμα και προτρέπεται να δοκιμάσει και πάλι.



Διάγραμμα ακολουθίας της περίπτωσης χρήσης Δημιουργία λογαριασμού

3.2.3) Σενάριο Αναζήτησης ταινίας

Αρχική παραδοχή: Ο χρήστης θέλει να αποφασίσει ποια ταινία θα παρακολουθήσει και είναι συνδεδεμένος στο ίντερνετ.
Κανονική ροή: Ο χρήστης ανοίγει την εφαρμογή στην συσκευή του. Αυτή συνδέεται με την βάση δεδομένων στον MySQL Server και κατεβάζει στο κινητό του χρήστη μια λίστα με τις διαθέσιμες ταινίες. Η λίστα περιλαμβάνει εγγραφές οι οποίες αποτελούνται από μια μικρογραφία της αφίσας της ταινίας, ορισμένες πολύ γενικές πληροφορίες και την ένδειξη 3D στις ταινίες που προβάλλονται σε τρισδιάστατη ανάλυση. Ο χρήστης επιλέγει κάποια από τις ταινίες της λίστας. Η εφαρμογή εμφανίζει τα στοιχεία της ταινίας που επέλεξε ο χρήστης, αυτή τη φορά όμως αναλυτικά. Δίπλα από την εικόνα της ταινίας εμφανίζεται μια περίληψη της πλοκής, ενώ από κάτω υπάρχει ένας αριθμός κουμπιών που εξυπηρετούν επιπλέον λειτουργίες. Ο χρήστης μπορεί να δει το είδος της ταινίας, την διάρκεια, μπορεί να δει φωτογραφίες από σκηνές της ταινίας, το trailer της στην ενσωματωμένη εφαρμογή του YouTube, τους ηθοποιούς που πρωταγωνιστούν, την ημερομηνία παραγωγής και τέλος εφόσον το επιλέξει να δει τις ημερομηνίες κατά τις οποίες προβάλετε στους κινηματογράφους. Όταν τελειώσει πατώντας το κουμπί «πίσω» εμφανίζεται και πάλι η αρχική οθόνη με την λίστα των ταινιών που παίζουν αυτή τη στιγμή στον κινηματογράφο. Από εκεί ο χρήστης μπορεί εκ νέου να επιλέξει κάποια ταινία και να δει τα αναλυτικά χαρακτηριστικά της.
Πιθανά προβλήματα: Η αρχική λίστα των ταινιών δεν φορτώνεται από την εφαρμογή. Στην περίπτωση αυτή η εφαρμογή θα πρέπει να εμφανίσει μήνυμα αποτυχίας φόρτωσης της λίστας και ο χρήστης αφού βεβαιωθεί πως είναι συνδεδεμένος στο διαδίκτυο θα πρέπει να επαναλάβει την φόρτωση της λίστας των ταινιών. Ένα άλλο πιθανό πρόβλημα είναι η εφαρμογή να εμφανίσει την λίστα

των ταινιών όμως να μην εμφανίσει τις λεπτομέρειες της επιλεγμένης από τον χρήστη ταινίας. Η εφαρμογή στο σημείο αυτό θα πρέπει να ενημερώσει τον χρήστη με μήνυμα για την αποτυχία της να εμφανίσει τις λεπτομέρειες της ταινίας και ο χρήστης θα πρέπει να προσπαθήσει να επαναλάβει την φόρτωση των λεπτομερειών η να επιλέξει μια άλλη ταινία από την αρχική λίστα.
Κατάσταση του συστήματος: Ο χρήστης παραμένει συνδεδεμένος στην εφαρμογή. Το σύστημα ενημερώνεται διαρκώς για τυχόν αλλαγές (στις ταινίες, στο πρόγραμμα).

Με βάση το σενάριο αναζήτησης ταινίας θα δημιουργήσουμε την **περίπτωση χρήσης αναζήτηση ταινίας**.

Σύντομη περιγραφή: Ο χρήστης χρησιμοποιεί την εφαρμογή προκειμένου να δει ποιες ταινίες προβάλλονται αυτή τη στιγμή στον κινηματογράφο καθώς και λεπτομέρειες για την κάθε ταινία.

Κύριος χρήστης: Πελάτης κινηματογράφου.

Εμπλεκόμενοι και ενδιαφέροντα: α)Χρήστης: θέλει εύχρηστο περιβάλλον, πολλές επιλογές αναφορικά με το ποιες ταινίες μπορεί να παρακολουθήσει και πληροφορίες σχετικά με την κάθε ταινία, εικόνες, trailer κλπ. β) Εταιρία κινηματογράφου: Θέλει η βάση δεδομένων να είναι πάντα ενημερωμένη με τις ταινίες που προβάλλονται αυτή τη στιγμή καθώς και με σωστές πληροφορίες για την κάθε ταινία. Επίσης θέλει η εφαρμογή να εξυπηρετεί όλες τις ανάγκες του χρήστη για πληροφορίες εύκολα και με ταχύτητα.

Προϋποθέσεις: Ο χρήστης είναι συνδεδεμένος στο διαδίκτυο. Ο MySQL Server λειτουργεί σωστά και στέλνει τις πληροφορίες των ταινιών στην εφαρμογή του κινητού του χρήστη.

Εγγύηση Επιτυχίας: Η λίστα με τις προβαλλόμενες ταινίες είναι σωστά ενημερωμένη. Για κάθε ταινία που έχει επιλέξει ο χρήστης προβάλλονται οι λεπτομέρειες της.

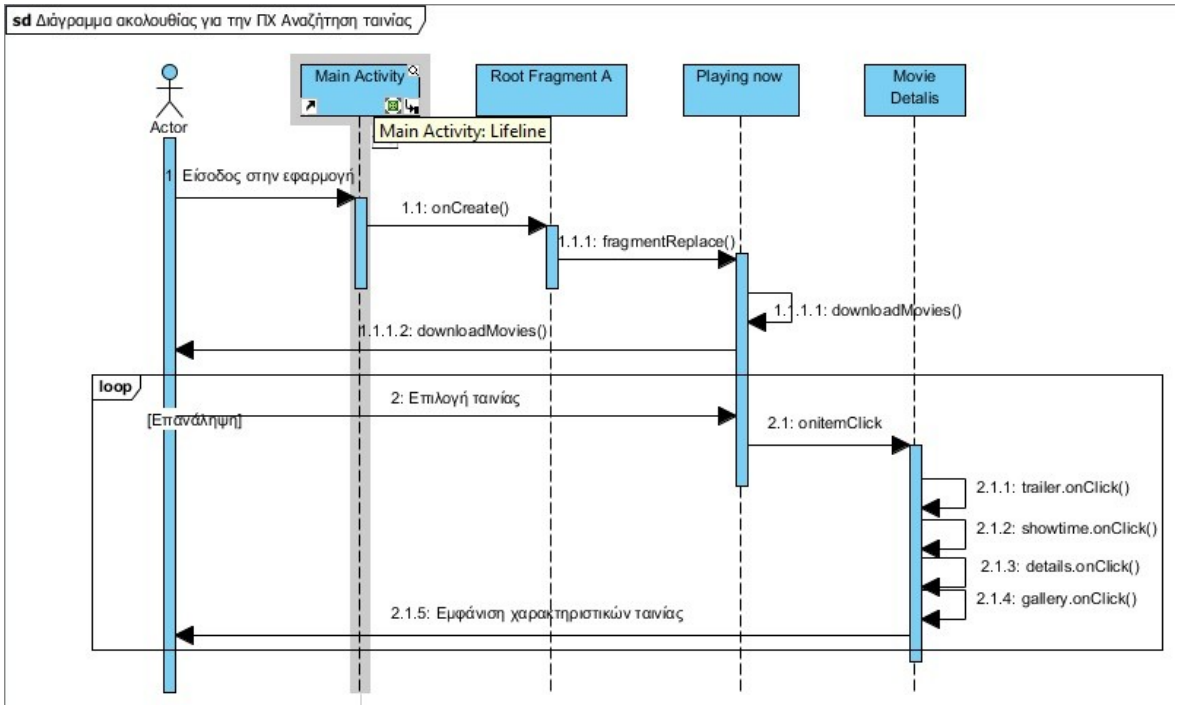
Κύριο Σενάριο Επιτυχίας:

1. Ο χρήστης ανοίγει την εφαρμογή προκειμένου να ενημερωθεί για τις ταινίες που προβάλλονται αυτή τη στιγμή στον κινηματογράφο.
2. Η εφαρμογή εμφανίζει στον χρήστη μια λίστα με τις διαθέσιμες ταινίες.
3. Ο χρήστης μελετά την λίστα των ταινιών και επιλέγει κάποια από αυτές.
4. Η εφαρμογή προβάλλει μια νέα λίστα με τις λεπτομέρειες της ταινίας που επιλέχθηκε
5. Ο χρήστης διαλέγει από ένα σύνολο επιλογών αναφορικά με τα χαρακτηριστικά της ταινίας.

6. Η εφαρμογή ανάλογα με την επιλογή του χρήστη του δείχνει τα επιθυμητά χαρακτηριστικά.
7. Ο χρήστης αφού δει τα χαρακτηριστικά της ταινίας που επέλεξε γυρνά με το κουμπί «πίσω» στην αρχική οθόνη και επιλέγει νέα ταινία.
8. Τα βήματα 4 έως 7 επαναλαμβάνονται για όσες ταινίες επιλέξει ο χρήστης να προβάλει τις λεπτομέρειες τους.
9. Ο χρήστης μπορεί είτε να κάνει μια κράτηση για κάποια ταινία η να τερματίσει την εφαρμογή.

Επεκτάσεις:

Α)Η εφαρμογή αποτυγχάνει σε κάποιο βήμα: Εμφανίζεται στον χρήστη μήνυμα σφάλματος και η εφαρμογή τερματίζεται. Ο χρήστης επανεκκινεί την εφαρμογή και ξεκινά εκ νέου την περιήγηση του.
2.α)Η εφαρμογή δεν εμφανίζει στον χρήστη την λίστα με τις ταινίες που παίζουν αυτή τη στιγμή στους κινηματογράφους. Εμφανίζεται στον χρήστη μήνυμα αδυναμίας φόρτωσης της λίστας και προτροπή να επαναλάβει την φόρτωση.
3.α1) Η εφαρμογή αποτυγχάνει να εμφανίσει τις λεπτομέρειες της επιλεγμένης ταινίας. Εμφανίζεται μήνυμα αδυναμίας φόρτωσης των λεπτομερειών και προτροπή να επαναλάβει την φόρτωση η να επιλέξει μια νέα ταινία για προβολή
3.β1) Η εφαρμογή εμφανίζει τις λεπτομέρειες της ταινίας.
3.β2) Ο χρήστης επιλέγει να δει το trailer της ταινίας στον ενσωματωμένο YouTube Player.
3.β3) Ο YouTube Player αποτυγχάνει να αναπαράγει το βίντεο.
3.β4) Η εφαρμογή εμφανίζει στον χρήστη μήνυμα αδυναμίας αναπαραγωγής του βίντεο και προσφέρει τη δυνατότητα μετάβασης στην εφαρμογή του YouTube για κινητά για απευθείας παρακολούθηση.



Διάγραμμα ακολουθίας της περίπτωσης χρήσης Αναζήτηση ταινίας

3.2.4) Σενάριο Προβολής ιστορικού κρατήσεων

Αρχική παραδοχή: Ο χρήστης θέλει να ενημερωθεί για το ιστορικό κρατήσεων του και είναι συνδεδεμένος στο ίντερνετ.
Κανονική ροή: Ο χρήστης ανοίγει την εφαρμογή στην συσκευή του. Αυτή εμφανίζει στον χρήστη την αρχική οθόνη με την λίστα των ταινιών που προβάλλονται στον κινηματογράφο. Ο χρήστης μεταβαίνει στην καρτέλα «Ιστορικό Κρατήσεων» προκειμένου να ελέγξει τις κρατήσεις του. Η εφαρμογή ελέγχει εάν ο χρήστης είναι συνδεδεμένος με κάποιον λογαριασμό (είτε της εφαρμογής ή του facebook) και του εμφανίζει μια λίστα με τις κρατήσεις που έχει πραγματοποιήσει ο λογαριασμός. Σε περίπτωση που ο χρήστης δεν έχει συνδεθεί σε κάποιον λογαριασμό η εφαρμογή του εμφανίζει λίστα τις κρατήσεις που έχουν πραγματοποιηθεί από το συγκεκριμένο τηλέφωνο. Στην λίστα αυτή φαίνεται η ημερομηνία της κράτησης και η ταινία στην οποία έχει κάνει κράτηση ο χρήστης. Οι ενεργές κρατήσεις εμφανίζονται στο πάνω μέρος της λίστας ενώ οι κρατήσεις που έχουν παρέλθει στο κάτω. Στο σημείο αυτό ο χρήστης μπορεί να επιλέξει κάποια από τις κρατήσεις για να προβάλει λεπτομέρειες αναφορικά με τον αριθμό των θέσεων που έχει κρατήσει καθώς και ποιες είναι αυτές. Επίσης έχει την δυνατότητα να μοιραστεί στο facebook την κράτηση του, ενημερώνοντας τους φίλους του για το πότε, που και ποια ταινία σκοπεύει να παρακολουθήσει.
Πιθανά προβλήματα: Η εφαρμογή δεν αναγνωρίζει τον λογαριασμό του χρήστη. Στο σημείο αυτό θα πρέπει να του εμφανίσει κατάλληλο μήνυμα και να του δώσει την επιλογή να δοκιμάσει ξανά να εισάγει τα στοιχεία του ή εάν πρόκειται για νέο χρήστη να δημιουργήσει επί τόπου λογαριασμό. Ο χρήστης μπορεί να μην έχει πραγματοποιήσει κάποια κράτηση. Η εφαρμογή θα πρέπει να τον ενημερώσει πως δεν υπάρχουν κρατήσεις από τον συγκεκριμένο χρήστη-συσκευή. Η εφαρμογή

μπορεί να αποτύχει να ανακτήσει την λίστα με τις κρατήσεις του χρήστη. Ο χρήστης θα πρέπει να ενημερωθεί με κατάλληλο μήνυμα ότι απέτυχε η ανάκτηση του ιστορικού κρατήσεων και πως θα πρέπει να δοκιμάσει ξανά. Κατάσταση του συστήματος: Ο χρήστης παραμένει συνδεδεμένος στην εφαρμογή. Το σύστημα είναι σωστά ενημερωμένο σχετικά με τους λογαριασμούς που βρίσκονται στη βάση του και τις κρατήσεις που έχει κάνει κάθε λογαριασμός.

Με βάση το σενάριο κράτησης εισιτηρίου θα δημιουργήσουμε την **περίπτωση χρήσης προβολής ιστορικού κρατήσεων**.

Σύντομη περιγραφή: Ο χρήστης χρησιμοποιεί την εφαρμογή προκειμένου να δει τις κρατήσεις που έχει δημιουργήσει καθώς και λεπτομέρειες για την κάθε κράτηση.

Κύριος χρήστης: Πελάτης κινηματογράφου.

Εμπλεκόμενοι και ενδιαφέροντα: α)Χρήστης: θέλει εύχρηστο περιβάλλον, σωστή ενημέρωση αναφορικά με τις λεπτομέρειες της κράτησης του, (πότε, σε ποια αίθουσα, ποιες θέσεις). β) Εταιρία κινηματογράφου: Θέλει η βάση δεδομένων να είναι πάντα ενημερωμένη με τις τρέχουσες κρατήσεις για κάθε ταινία και κάθε αίθουσα. Η βάση πρέπει να είναι ενημερωμένη σωστά και να ενημερώνεται σε πραγματικό χρόνο προκειμένου να αποφεύγονται διπλοκρατήσεις και να είναι πάντα έγκυρο το διάγραμμα των ελεύθερων και κρατημένων θέσεων.

Προϋποθέσεις: Ο χρήστης είναι συνδεδεμένος στο διαδίκτυο. Ο MySQL Server λειτουργεί σωστά και στέλνει τις πληροφορίες των κρατήσεων του στην εφαρμογή του κινητού του χρήστη.

Εγγύηση Επιτυχίας: Η λίστα με τις κρατήσεις του χρήστη, τόσο τις ενεργές όσο και αυτές που έχουν παρέλθει είναι σωστά ενημερωμένη. Για κάθε κράτηση που έχει επιλέξει ο χρήστης προβάλλονται οι θέσεις που αντιστοιχούν στην συγκεκριμένη κράτηση.

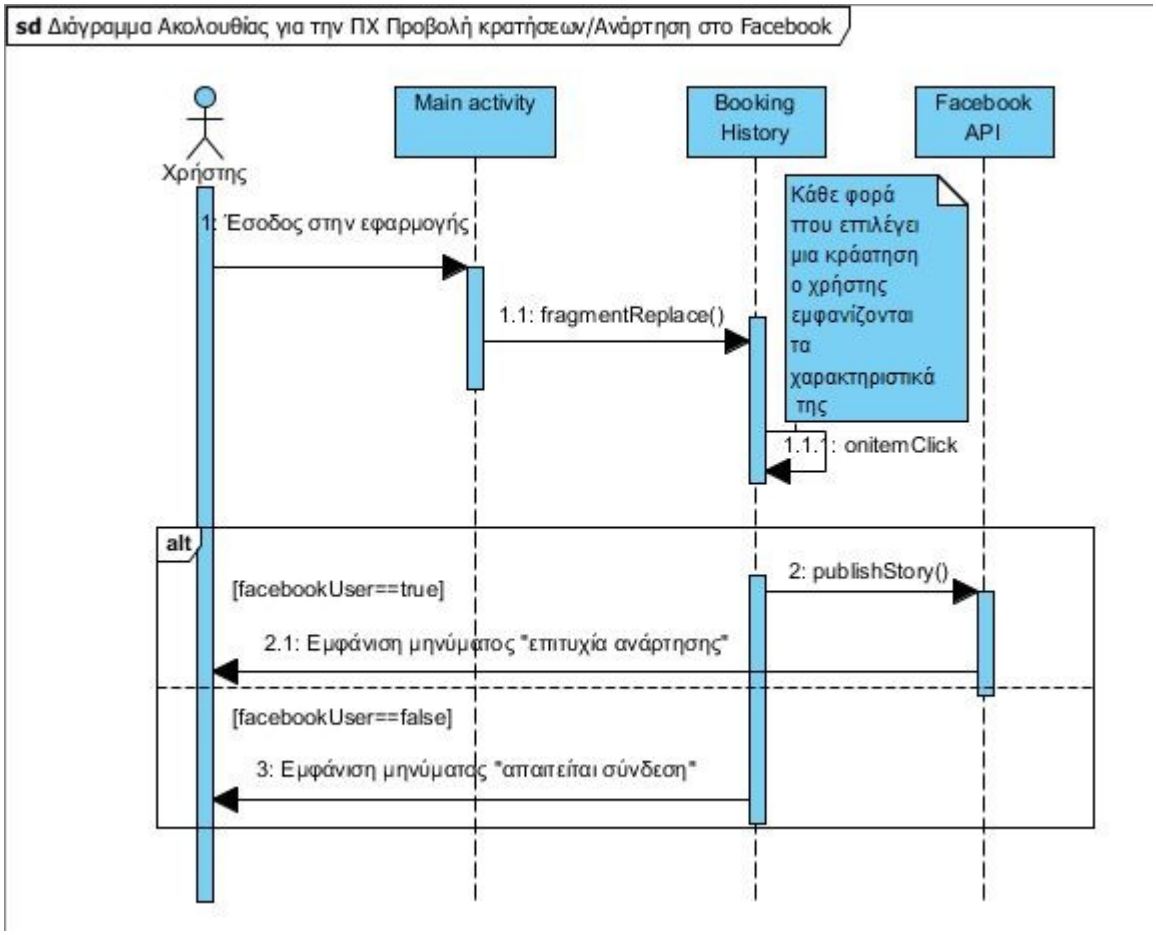
Κύριο Σενάριο Επιτυχίας:

1. Ο χρήστης ανοίγει την εφαρμογή προκειμένου να ελέγξει το ιστορικό κρατήσεων του.
2. Η εφαρμογή εμφανίζει στον χρήστη μια την αρχική οθόνη της που περιλαμβάνει την καρτέλα με την λίστα των ταινιών καθώς και τρεις άλλες καρτέλες στις οποίες μπορεί να μεταβεί ο χρήστης.
3. Ο χρήστης επιλέγει την καρτέλα «Ιστορικό Κρατήσεων»
4. Η εφαρμογή ελέγχει εάν ο χρήστης είναι συνδεδεμένος με κάποιο λογαριασμό στην εφαρμογή. Έπειτα αναλόγως εμφανίζει την λίστα με τις κρατήσεις που αντιστοιχούν στον συγκεκριμένο χρήστη σε δυο μικρότερες λίστες, επάνω τις ενεργές και κάτω αυτές που έχουν παρέλθει.

5. Ο χρήστης επιλέγει μια κράτηση από την λίστα
6. Η εφαρμογή ανασύρει τις λεπτομέρειες της κράτησης από την βάση και εμφανίζει στον χρήστη τις θέσεις που αντιστοιχούν στην κράτηση του.
7. Ο χρήστης αφού δει τα χαρακτηριστικά της κράτησης του γυρνά με το κουμπί «πίσω» στην οθόνη κρατήσεων.
8. Τα βήματα 4 έως 7 επαναλαμβάνονται για όσες κρατήσεις επιλέξει ο χρήστης να προβάλει τις λεπτομέρειες τους.
9. Ο χρήστης έχει μια σειρά επιλογών. Μπορεί να αναρτήσει την κράτηση του στο facebook, να επιλέξει κάποια άλλη καρτέλα ή να τερματίσει την εφαρμογή.

Επεκτάσεις:

Α)Η εφαρμογή αποτυγχάνει σε κάποιο βήμα: Εμφανίζεται στον χρήστη μήνυμα σφάλματος και η εφαρμογή τερματίζεται. Ο χρήστης επανεκκινεί την εφαρμογή και την ξεκινά εκ νέου.
3.α)Η εφαρμογή δεν εμφανίζει στον χρήστη την λίστα με τις κρατήσεις που αντιστοιχούν στον λογαριασμό του. Εμφανίζεται στον χρήστη μήνυμα αδυναμίας φόρτωσης της λίστας και προτροπή να επαναλάβει την φόρτωση ή να ελέγξει τα διαπιστευτήρια του λογαριασμού του.
6.α1) Η εφαρμογή αποτυγχάνει να εμφανίσει τις λεπτομέρειες της επιλεγμένης κράτησης. Εμφανίζεται μήνυμα αδυναμίας φόρτωσης στον χρήστη και προτροπή να επαναλάβει την φόρτωση.
7.α) Η εφαρμογή εμφανίζει τις λεπτομέρειες της κράτησης.
7.α1) Ο χρήστης επιλέγει να δει μοιραστεί τις λεπτομέρειες της κράτησης στο facebook.
7.α2) Το Facebook API αποτυγχάνει να αναρτήσει στο ιστολόγιο του χρήστη τις λεπτομέρειες της κράτησης.
7.α3) Η εφαρμογή εμφανίζει στον χρήστη μήνυμα αδυναμίας ανάρτησης της κράτησης και επιστρέφει στην καρτέλα του Ιστορικού Κρατήσεων.



Διάγραμμα ακολουθίας της περίπτωσης χρήσης Προβολής κρατήσεων

3.2.5) Σενάριο Αναζήτησης τοποθεσίας κινηματογράφου

Αρχική παραδοχή: Ο χρήστης έχει πραγματοποιήσει μια κράτηση και θέλει να μάθει την τοποθεσία του κινηματογράφου που θα παρακολουθήσει την ταινία. Κανονική ροή: Ο χρήστης ανοίγει την εφαρμογή στην συσκευή του. Αυτή εμφανίζει στον χρήστη την αρχική οθόνη με την λίστα των ταινιών που προβάλλονται στον κινηματογράφο. Ο χρήστης μεταβαίνει στην καρτέλα «Τοποθεσία» προκειμένου να ελέγξει την τοποθεσία του κινηματογράφου. Η εφαρμογή φορτώνει το Google Maps API που είναι ενσωματωμένο στην εφαρμογή. Στην καρτέλα αυτή ο χρήστης μπορεί να επιλέξει να μάθει την τοποθεσία κάποιου κινηματογράφου πατώντας το αντίστοιχο κουμπί. Ο χρήστης επιλέγει έναν κινηματογράφο, η εφαρμογή ελέγχει τις αποθηκευμένες στη βάση συντεταγμένες του κινηματογράφου και εμφανίζει μέσω του Google Maps την ακριβή τοποθεσία του. Πιθανά προβλήματα: Η εφαρμογή δεν φορτώνει σωστά τις συντεταγμένες του κινηματογράφου. Ο χρήστης στο σημείο αυτό θα πρέπει να επαναλάβει την φόρτωση της τοποθεσίας. Το Google Maps API μπορεί να αποτύχει να φορτώσει. Ο χρήστης θα πρέπει να ενημερωθεί με κατάλληλο μήνυμα ότι απέτυχε η φόρτωση των χαρτών και πως θα πρέπει να δοκιμάσει ξανά καθώς και να ελέγξει την σύνδεσή του στο ίντερνετ.

Κατάσταση του συστήματος: Ο χρήστης παραμένει συνδεδεμένος στην εφαρμογή και στο ίντερνετ. Το σύστημα είναι σωστά ενημερωμένο σχετικά με την τοποθεσία των κινηματογράφων. Το Google Maps API αποστέλλει σωστά την τοποθεσία και λαμβάνει ακριβείς συντεταγμένες.

Με βάση το σενάριο κράτησης εισιτηρίου θα δημιουργήσουμε την **περίπτωση αναζήτησης κινηματογράφου**.

Σύντομη περιγραφή: Ο χρήστης χρησιμοποιεί την εφαρμογή προκειμένου να δει την τοποθεσία του κινηματογράφου που έχει κάνει κράτηση.

Κύριος χρήστης: Πελάτης κινηματογράφου.

Εμπλεκόμενοι και ενδιαφέροντα: α)Χρήστης: Θέλει εύχρηστο περιβάλλον, γρήγορη ανταπόκριση της εφαρμογής. Ακριβή δεδομένα αναφορικά με την τοποθεσία των κινηματογράφων. β) Εταιρία κινηματογράφου: Θέλει η βάση δεδομένων να είναι ενημερωμένη με τις σωστές τοποθεσίες. Θέλει οι χάρτες να ανταποκρίνονται πιστά στην πραγματικότητα.

Προϋποθέσεις: Ο χρήστης είναι συνδεδεμένος στο διαδίκτυο. Ο MySQL Server λειτουργεί σωστά και στέλνει τις πληροφορίες των κινηματογράφων στην εφαρμογή του κινητού του χρήστη. Το Google Maps API λειτουργεί και δείχνει σωστά την τοποθεσία του κινηματογράφου

Εγγύηση Επιτυχίας: Ο χάρτης με τις τοποθεσίες των κινηματογράφων είναι σωστά ενημερωμένος. Η βάση έχει τις συντεταγμένες κάθε κινηματογράφου αποθηκευμένες.

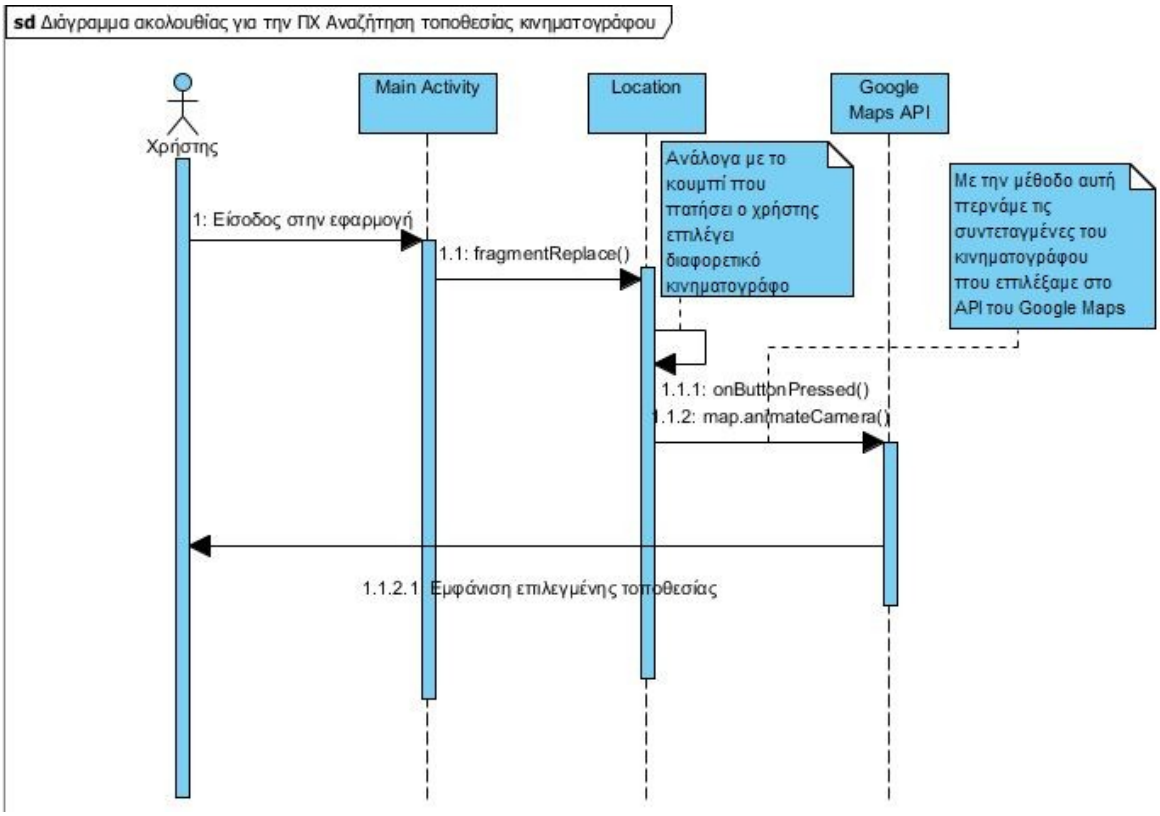
Κύριο Σενάριο Επιτυχίας:

1. Ο χρήστης ανοίγει την εφαρμογή προκειμένου να ελέγξει την τοποθεσία του κινηματογράφου.
2. Η εφαρμογή εμφανίζει στον χρήστη μια την αρχική οθόνη της που περιλαμβάνει την καρτέλα με την λίστα των ταινιών καθώς και τρεις άλλες καρτέλες στις οποίες μπορεί να μεταβεί ο χρήστης.
3. Ο χρήστης επιλέγει την καρτέλα «Τοποθεσία»
4. Η εφαρμογή εμφανίζει κουμπιά με τις συντεταγμένες τριών διαφορετικών κινηματογράφων καθώς και έναν χάρτη.
5. Ο χρήστης επιλέγει μια τοποθεσία από τις τρεις διαθέσιμες.
6. Η εφαρμογή ελέγχει τις συντεταγμένες που αντιστοιχούν στην συγκεκριμένη τοποθεσία.
7. Η εφαρμογή στέλνει τις συντεταγμένες στο API των χαρτών το οποίο με τη σειρά του εμφανίζει την τοποθεσία του κινηματογράφου στον χρήστη.

8. Ο χρήστης μπορεί να επιλέξει την τοποθεσία ενός άλλου κινηματογράφου για εμφάνιση, να μεταβεί σε κάποια άλλη καρτέλα η να τερματίσει την εφαρμογή.

Επεκτάσεις:

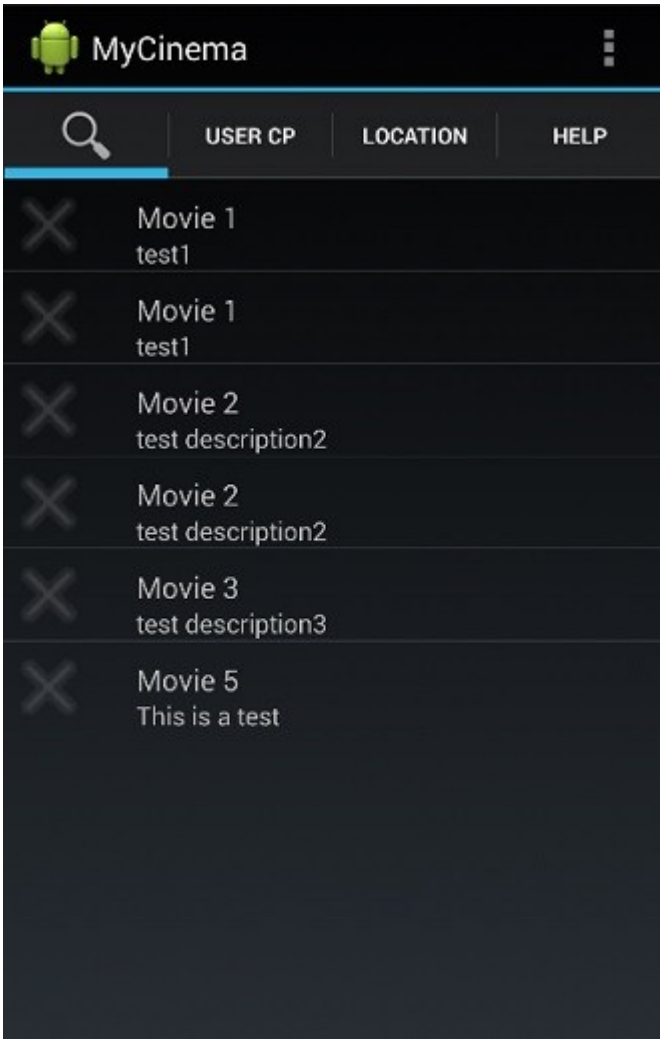
Α)Η εφαρμογή αποτυγχάνει σε κάποιο βήμα: Εμφανίζεται στον χρήστη μήνυμα σφάλματος και η εφαρμογή τερματίζεται. Ο χρήστης επανεκκινεί την εφαρμογή και την ξεκινά εκ νέου.
4.α)Η εφαρμογή δεν εμφανίζει στον χρήστη τον χάρτη. Εμφανίζεται στον χρήστη μήνυμα αδυναμίας φόρτωσης του χάρτη και προτροπή να επαναλάβει την φόρτωση ή να ελέγξει την σύνδεση του με το ίντερνετ.
6.α) Η εφαρμογή αποτυγχάνει να αναγνωρίσει τις συντεταγμένες του κινηματογράφου.
6.α1) Ο χρήστης επαναλαμβάνει τη φόρτωση του χάρτη.



3.3) Στάδια εξέλιξης της εφαρμογής, Εξελικτικά Πρωτότυπα

Κατά την προηγούμενη φάση της διαδικασίας ανάπτυξης χρησιμοποιήσαμε σενάρια και περιπτώσεις χρήσης προκειμένου να προσδιορίσουμε τις ανάγκες των χρηστών της εφαρμογής. Αφού λοιπόν καθορίσαμε τα χαρακτηριστικά τα οποία

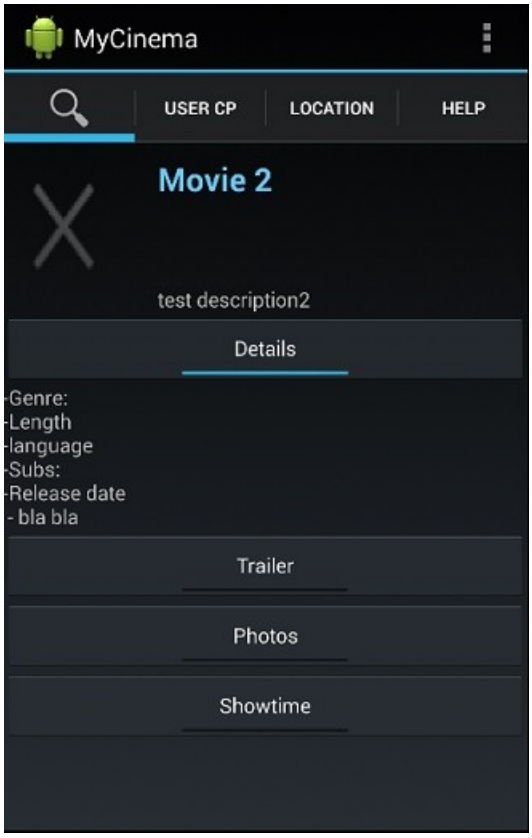
πρέπει να ενσωματώνει η εφαρμογή μας για να ικανοποιεί τους χρήστες προχωρήσαμε στην δημιουργία πρωτοτύπων προκειμένου να επαληθεύσουμε την ορθότητα της καταγραφής των απαιτήσεων, να διερευνήσουμε τα βασικά εναλλακτικά σενάρια σχεδιασμού και να δοκιμάσουμε νέες τεχνολογίες. Τα πρωτότυπα που δημιουργήσαμε ακολουθούν την τελική αρχιτεκτονική του συστήματος χρησιμοποιώντας τα δομικά συστατικά του, ενώ θα πρέπει στο σημείο αυτό να πούμε πως το τελικό προϊόν αποτελεί το αποτέλεσμα της διαδικασίας αυτή.



Πρώτα βήματα: Σε αυτή τη φάση αυτό που θέλαμε ήταν μια πρώτη άποψη για την διαρρύθμιση της εφαρμογής, τον τρόπο εμφάνισης καθώς και το πώς θα μετακινούνταν ο χρήστης από το ένα σημείο της εφαρμογής στο άλλο. Το σχέδιο στο οποίο καταλήξαμε ήταν το παρακάτω. Υπήρχαν τέσσερις καρτέλες με διαφορετική λειτουργία για την κάθε μια, ταινίες, ρυθμίσεις λογαριασμού χρήστη, τοποθεσία και βοήθεια. Οι ταινίες εμφανίζονται υπό τη μορφή λίστας χωρίς επιπλέον χαρακτηριστικά εκτός του τίτλου και μιας μικρής περιγραφής. Οι ρυθμίσεις βρίσκονται στην επάνω δεξιά γωνία της εφαρμογής.

Έκδοση 0.1 Καρτέλα λεπτομερειών ταινίας

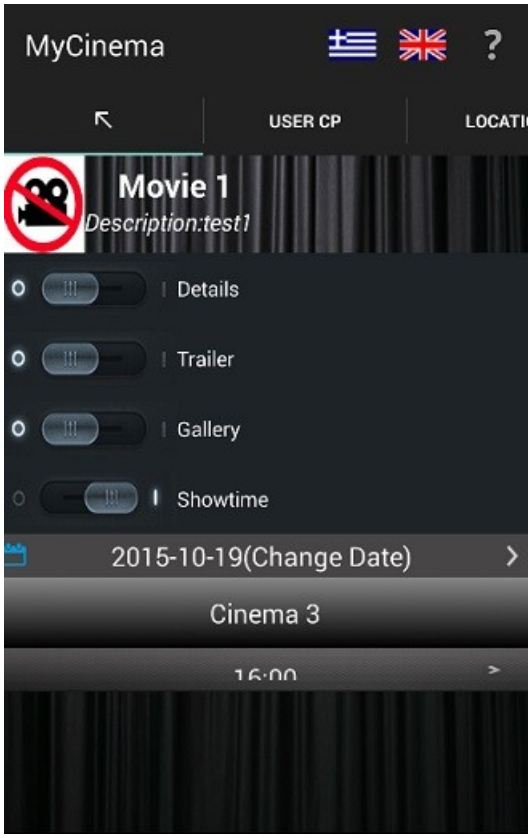
Έκδοση 0.1 Καρτέλα ταινιών



Μετά την επιλογή κάποιας ταινίας η εφαρμογή εμφανίζει την καρτέλα των λεπτομερειών. Ο χρήστης μπορεί να δει πληροφορίες για την ταινία, φωτογραφίες, το trailer της και το πρόγραμμα προβολών της.

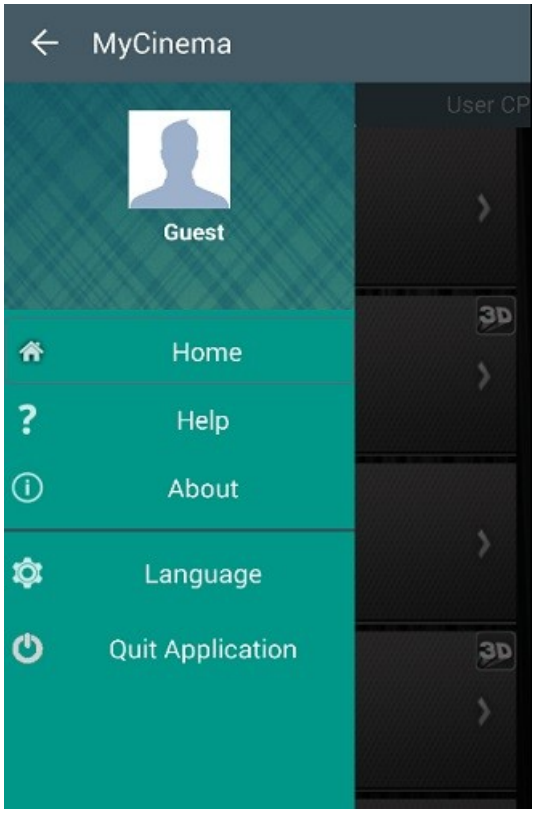
Έκδοση 0.3 Εισαγωγή καρτέλας ιστορικού κρατήσεων

Εξελίσσοντας την εφαρμογή και θέλοντας να συμβαδίζουμε με τις νέες τεχνολογίες που ενσωματώνουν οι νεότερες εκδόσεις του Android αλλάξαμε την διεπαφή χρήστη. Πλέον οι μετακίνηση από καρτέλα σε καρτέλα μπορεί να γίνει και με χειρονομίες (gestures). Η εμφάνιση του Swipe έκανε την ζωή των χρηστών πιο εύκολη, καθώς δεν χρειαζόταν πλέον ο χρήστης να προσπαθεί να πατήσει μικροσκοπικά κουμπιά στην οθόνη του, αλλά με μια κίνηση να αλλάξει καρτέλα. Επίσης στην συγκεκριμένη φάση κάναμε πιο εμφανή κάποιες ρυθμίσεις της εφαρμογής, πιο συγκεκριμένα την επιλογή γλώσσας. Η τελευταία αλλαγή σε αυτή τη φάση ήταν η αντικατάσταση της καρτέλας βοήθειας με αυτήν του ιστορικού κρατήσεων.



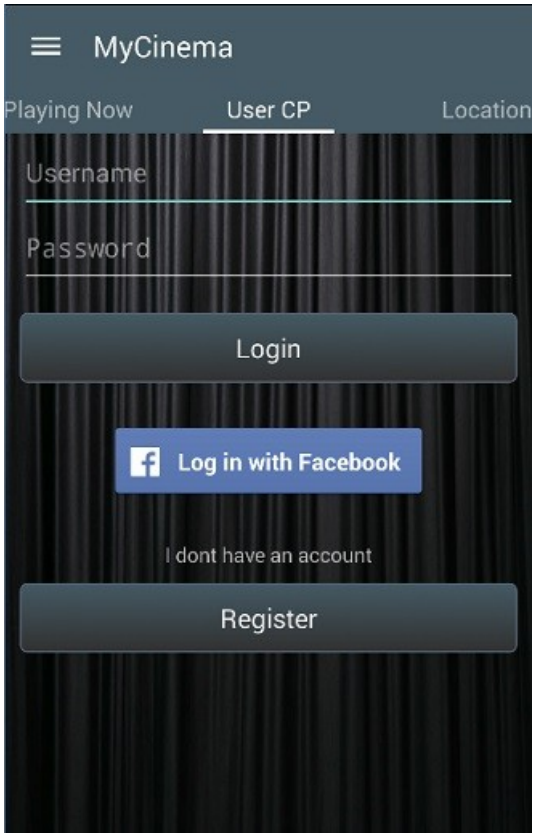
Σε αυτή την έκδοση εισάγαμε λειτουργικότητα στην επιλογή της λίστας προβολών της ταινίας που έχει επιλέξει ο χρήστης. Πλέον ο χρήστης μπορεί να δει και να επιλέξει την ημερομηνία, τον κινηματογράφο και την προβολή της επιλογής του για να κάνει κράτηση σε κάποια ταινία. Επίσης στην έκδοση αυτή χρησιμοποιήσαμε το YouTube Player API προκειμένου να μπορούν οι χρήστες να παρακολουθήσουν trailer της ταινίας που έχουν επιλέξει χωρίς να βγουν από την εφαρμογή. Τέλος λειτουργικότητα απέκτησε και η καρτέλα Gallery στην οποία προβάλλονται φωτογραφίες της επιλεγμένης ταινίας που είναι αποθηκευμένες στη βάση.

Έκδοση 0.3 Εισαγωγή λειτουργικότητας στην καρτέλα λεπτομερειών ταινίας



Για ακόμη μια φορά αλλάζουμε την διεπαφή χρήστη, ενσωματώνοντας αυτή τη φορά τον Android Navigation Drawer. Στόχος μας ήταν τόσο η εξοικονόμηση χώρου προκειμένου να προσφέρουμε περισσότερο περιεχόμενο της εφαρμογής στον χρήστη αντί να καταναλώνουμε μέρος της οθόνης του μόνο για την πρόσβαση στις ρυθμίσεις της εφαρμογής, όσο και να προσφέρουμε μια στον χρήστη μια επισκόπηση της εφαρμογής, πληροφορίες σχετικά με τη χρήση της, σχετικά με τον λογαριασμό του και παράλληλα να διατηρήσουμε προηγούμενα χαρακτηριστικά όπως την επιλογή γλώσσας.

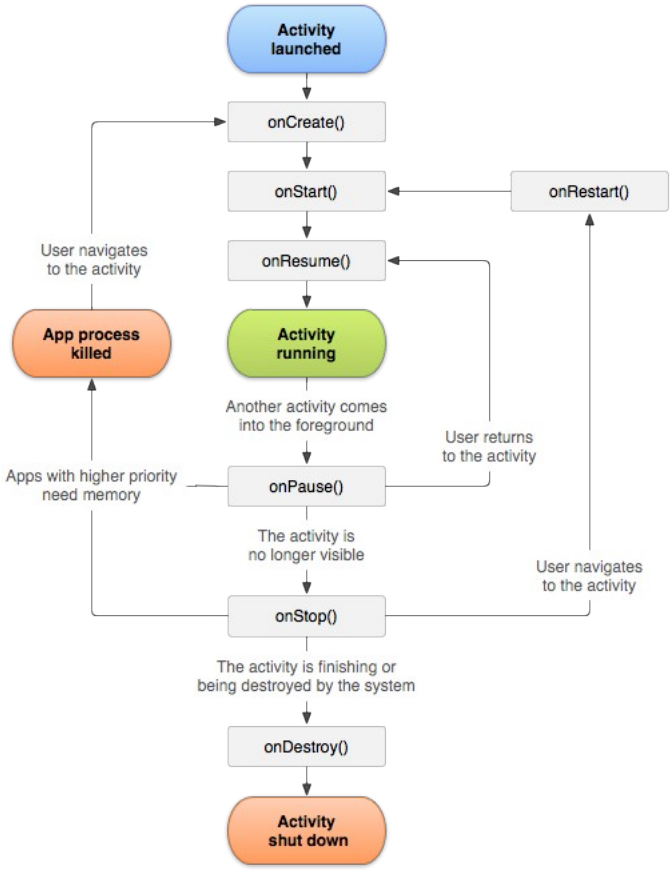
Έκδοση 0.8. Εισαγωγή καρτέλας ρυθμίσεων απευθείας στην διεπαφή χρήστη με υλοποίηση Android Navigation Drawer



Στην τελευταία έκδοση της εφαρμογής ενσωματώσαμε το Facebook API μέσω του οποίου ο χρήστης μπορεί πλέον να χρησιμοποιήσει τον λογαριασμό του στο Facebook προκειμένου να συνδεθεί στην εφαρμογή. Επίσης ενσωματώσαμε την δυνατότητα «μοιράσματος» κάποιας κράτησης που έχει κάνει ο χρήστης. Από την καρτέλα ιστορικού κρατήσεων πλέον ο χρήστης μπορεί να επιλέξει μια κράτηση και να την μοιραστεί με τους φίλους του ενημερώνοντας τους για την ταινία που πρόκειται να παρακολουθήσει.

Έκδοση 1.0. Ενσωμάτωση του Facebook API, ο χρήστης έχει τη δυνατότητα να «μοιράζεται» τις ταινίες που παρακολουθεί

4) Ανάλυση του κώδικα της εφαρμογής



Μια εφαρμογή Android μπορεί να χωριστεί σε τέσσερα διακριτά κομμάτια. Την συμπεριφορά της η οποία δημιουργείται σε κώδικα Java και ο οποίος ελέγχει τι συμβαίνει όταν πατιούνται κουμπιά, γίνονται κλήσεις σε κάποιον εξυπηρετητή ή συμβαίνει οποιαδήποτε άλλη συμπεριφορά στην εφαρμογή. Ο κώδικας βρίσκεται στο source directory του project μας. Ένα άλλο κομμάτι οποιασδήποτε εφαρμογής Android είναι τα binary assets. Εκεί βρίσκονται οι εικόνες που μορφοποιούν την εφαρμογή μας καθώς και άλλοι πόροι που καθορίζουν την εμφάνιση - το στυλ της εφαρμογής Τα binary assets βρίσκονται στον φάκελο assets του project μας. Ακόμη υπάρχουν τα XML Layouts. Ο σκοπός τους είναι η μορφοποίηση της όσον αφορά τη διάταξη των στοιχείων της. Στην XML επίσης καθορίζεται πλήθος άλλων ιδιοτήτων της εφαρμογής, όπως τιμές μεταβλητών, χρώματα κλπ. Τα αρχεία XML βρίσκονται στον φάκελο res του Project. Το τελευταίο κομμάτι οποιουδήποτε Android Project που αποτελεί την κόλα που κρατά ενωμένα όλα τα υπόλοιπα είναι τα Configuration files. Σε αυτά ορίζονται τα πάντα από το όνομα της εφαρμογής μας στην αρχική σελίδα μέχρι τις διαφορετικές οθόνες της εφαρμογής. Παρακάτω θα αναλύσουμε την λειτουργικότητα του κώδικα της εφαρμογής καθώς και το σκεπτικό πίσω από την χρήση συγκεκριμένων τεχνικών .

Στο διπλανό διάγραμμα παραθέτουμε τον τυπικό κύκλο ζωής μιας δραστηριότητας σε ένα πρόγραμμα Android. Μπορούμε να ξεχωρίσουμε τον κύκλο ζωής σε τρεις ξεχωριστές επαναλήψεις, τον ολικό (από την πρώτη κλήση της onCreate() μέχρι την τελική κλίση της onDestroy()), τον ορατό (μεταξύ μιας κλήσης στην onStart() και μιας κλήσης στην onStop()) και τέλος αυτόν που

βρίσκεται στο προσκήνιο (από μια κλήση στην onResume() μέχρι μια κλήση στην onPause().

4.1) Activities

Η εφαρμογή μας χωρίζεται σε Activities και Fragments. Κάθε Activity περιέχει έναν αριθμό από Fragments. Κάθε ξεχωριστό Activity χρησιμοποιείται όταν αλλάζει σημαντικά η χρήση του προγράμματος.

4.1.1)Main Activity

Σε αντίθεση με άλλα προγραμματιστικά παραδείγματα που οι εφαρμογές ξεκινούν με την μέθοδο main() οι εφαρμογές του Android ξεκινούν τον κώδικα μέσα σε Activities, καλώντας συγκεκριμένες μεθόδους που αντιστοιχούν σε συγκεκριμένα στάδια του κύκλου ζωής τους. Υπάρχει μια ακολουθία μεθόδων που ξεκινούν μια δραστηριότητα και άλλη ακολουθία που καταστρέφει την δραστηριότητα. Στην Main Activity περιέχονται τα Fragments Playing Now, Movie Details, Login, Register, User Cp Location και Booking History.

```
import ...

public class MainActivity extends ActionBarActivity {
    FragmentPagerAdapter mAdapter;
    DrawerLayout mDrawerLayout;
    LinearLayout mDrawerList;
    ActionBarDrawerToggle mDrawerToggle;
    ViewPager mPager;
    ProfilePictureView drawerUserImg;
    TextView drawerName;
    Button drawerHome, drawerHelp, drawerAbout, drawerLanguage, drawerQuit;
    private UiLifecycleHelper uiHelper;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        PreferenceManager.getDefaultSharedPreferences(this).edit().putInt("framePosition", 0).commit();
        String storedLocale=PreferenceManager.getDefaultSharedPreferences(this).getString("locale","en");
        Locale myLocale = new Locale(storedLocale);
        Resources res = getResources();
        DisplayMetrics dm = res.getDisplayMetrics();
        Configuration conf = res.getConfiguration();
        conf.locale = myLocale;
        res.updateConfiguration(conf, dm);
        initiateViews();
        initiateListeners();
        uiHelper = new UiLifecycleHelper(this, callback);
        uiHelper.onCreate(savedInstanceState);
        //getSession(savedInstanceState);


    }
```

Η ActionBarActivity είναι η κλάση που κληρονομούμε η οποία μας δίνει πρόσβαση στις μεθόδους του κύκλου ζωής της δραστηριότητας όπως την

onCreate(), την onResume() και την onDestroy() . Επίσης η ActionBarActivity μας δίνει τη λειτουργικότητα της ActionBar που παρέχει έναν χώρο στο πάνω μέρος της οθόνης με πληροφορίες σχετικά με την ταυτότητα της εφαρμογής.

Η πρώτη μέθοδος που καλείται κατά την έναρξη της εφαρμογής είναι η onCreate(). Στην μέθοδο αυτή αρχικοποιείται το περιβάλλον της εφαρμογής. Με την χρήση του PreferenceManager αποκτούμε πρόσβαση στα SharedPreferences της εφαρμογής όπου αποθηκεύονται οι global μεταβλητές της, όπως το framePosition και το locale. Το framePosition καθορίζει την αρχική οθόνη της εφαρμογής(την λίστα με τις προβαλλόμενες ταινίες), ενώ το locale την γλώσσα εμφάνισης. Στη συνέχεια με την τιμή που βρήκαμε στα SharedPreferences καθορίζουμε την γλώσσα εμφάνισης. Με την χρήση της μεθόδου getResources() παίρνουμε μια αναφορά των πόρων της εφαρμογής που δεν σχετίζονται με τον κώδικα. Με την μέθοδο getDisplayMetrics() λαμβάνουμε πληροφορίες σχετικά με την απεικόνιση της πληροφορίας στην εφαρμογή (μέγεθος γραμματοσειράς, πυκνότητα). Μέσω της μεταβλητής conf καθορίζουμε το locale και με το updateConfiguration δίνουμε τιμές για την γλώσσα και την εμφάνιση.

Με την initiateViews() παρέχουμε μια διασύνδεση μεταξύ της XML και της Java έτσι ώστε να έχουμε πρόσβαση στα στοιχεία της XML μέσω του κώδικα όπως φαίνεται παρακάτω. Στην μέθοδο αυτή γίνεται και η υλοποίηση του Navigation Drawer που εμφανίζεται όταν ο χρήστης κάνει swipe προς τα δεξιά. Ο Navigation Drawer προσφέρει πληροφορίες στον χρήστη σχετικά με την εφαρμογή και του

δίνει την δυνατότητα να αλλάξει γλώσσα.

```
private void initiateViews() {
    getSupportActionBar().setBackgroundDrawable(getResources().getDrawable(R.drawable.cap));
    setContentView(R.layout.activity_main);
    drawerName = (TextView) findViewById(R.id.drawer_name);
    mDrawerLayout = (DrawerLayout) findViewById(R.id.drawerLayout);
    mDrawerList = (LinearLayout) findViewById(R.id.left_drawer);
    drawerUserImg = (ProfilePictureView) findViewById(R.id.userImage);

    drawerHome = (Button) findViewById(R.id.drawer_home);
    drawerHelp = (Button) findViewById(R.id.drawer_Help);
    drawerAbout = (Button) findViewById(R.id.drawer_about);
    drawerLanguage = (Button) findViewById(R.id.drawer_language);
    drawerQuit = (Button) findViewById(R.id.drawer_quit);
    mDrawerToggle = new ActionBarDrawerToggle(
            this,
            mDrawerLayout,
            "No seats selected",
            "Booked Seats"
    ) {

        /** Called when a drawer has settled in a completely closed state. */
        public void onDrawerClosed(View view) { super.onDrawerClosed(view); }

        /** Called when a drawer has settled in a completely open state. */
        public void onDrawerOpened(View drawerView) { super.onDrawerOpened(drawerView); }
    };
    mDrawerLayout.setDrawerListener(mDrawerToggle);
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    getSupportActionBar().setHomeButtonEnabled(true);
    mDrawerLayout.setFocusableInTouchMode(false);
    mAdapter = new SlidePagerAdapter(this);
    mPager = (ViewPager) findViewById(R.id.pager);
    mPager.setOffscreenPageLimit(3);
    mPager.setAdapter(mAdapter);
}
```

Με την InitiateListeners() ορίζουμε τις ενέργειες που θα πραγματοποιούν τα στοιχεία της XML κατά την αλληλεπίδραση με τον χρήστη. Τι γίνεται δηλαδή κατά το πάτημα ενός κουμπιού η κατά την κύλιση μιας λίστας. Εδώ γίνεται και η υλοποίηση των λειτουργιών που παρέχει ο Navigation Drawer.

```
private void initiateListeners() {
    mDrawerList.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

        }
    });

    drawerUserImg.setOnClickListener((v) -> {
            mPager.setCurrentItem(1);
            mDrawerLayout.closeDrawer(Gravity.LEFT);

    });
    drawerHome.setOnClickListener((v) -> {
            if (getSupportFragmentManager().getBackStackEntryCount() > 0) {
                getSupportFragmentManager().popBackStackImmediate(null, FragmentManager.POP_BACK_STACK_INCLUSIVE);
            }
            mPager.setCurrentItem(0);
            mDrawerLayout.closeDrawer(Gravity.LEFT);

    });
    drawerLanguage.setOnClickListener((v) -> {
            AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
            builder.setTitle("Select Language");
            final CharSequence[] items = {
                    "English", "Greek"
            };

            builder.setItems(items, (dialog, item) -> {
                    if (item == 0) {
                        setLocale("en");
                    } else {
                        setLocale("gr");
                    }
            });
            AlertDialog alert = builder.create();
            alert.show();

            mDrawerLayout.closeDrawer(Gravity.LEFT);

    });
}
```

Το uiHelper μας δίνει πρόσβαση στον κύκλο ζωής του Facebook session το οποίο θα αναλύσουμε παρακάτω.

```
@Override
public void onBackPressed() {
    if (mDrawerLayout.isDrawerOpen(Gravity.LEFT)) {
        mDrawerLayout.closeDrawer(Gravity.LEFT);
    } else {
        super.onBackPressed();
    }
}
```

Στην κλάση Main Activity έχουμε υπέρβαση της onBackPressed() που καθορίζει τι συμβαίνει όταν πατήσουμε το κουμπί «πίσω» στην εφαρμογή. Σύμφωνα με τον παρακάτω κώδικα, σε περίπτωση που ο Navigation Drawer είναι ανοιχτός τότε με πατώντας το κουμπί «πίσω» κλείνουμε τον Navigation Drawer και επιστρέφουμε στην ροή του προγράμματος. Αλλιώς η λειτουργικότητα της μεθόδου παραμένει όπως καθορίστηκε από την υπερκλάση ActionBar Activity.

Η μέθοδος changeFragment δέχεται σαν ορίσματα ένα fragment και έναν ακέραιο ο οποίος είναι ένας δείκτης αναφορικά με το αν θέλουμε να αποθηκεύσουμε στην ουρά της στοίβας των fragment το fragment που αλλάζουμε. Σε περίπτωση που ο

ακέραιος πάρει την τιμή «1» το fragment προστίθεται στην στοίβα και θα είναι αυτό που θα εμφανιστεί σε περίπτωση που πατήσουμε το «πίσω»

```
private void changeFragment(Fragment fra,int i){

    FragmentTransaction ft =getSupportFragmentManager().beginTransaction();
    ft.replace(R.id.root_frame_b, fra);
    if(i==1)
    {
        ft.addToBackStack(null);
    }
    ft.commit();
}
```

4.1.2) Showtime Activity

Η κλάση αυτή χρησιμοποιείται αφού ο χρήστης επιλέξει ταινία και ημερομηνία προβολής σε κάποιο κινηματογράφο. Η κλάση εμπεριέχει τα fragments τα οποία χρησιμοποιούνται προκειμένου ο χρήστης να διαλέξει προβολή, θέσεις στην αίθουσα και τρόπο κράτησης του εισιτηρίου του.

```
public class ShowtimeActivity extends ActionBarActivity implements TimePicker.OnFragmentInteractionListener {
    DrawerLayout mDrawerLayout;
    LinearLayout mDrawerList;
    ActionBarDrawerToggle mDrawerToggle;
    ProfilePictureView drawerUserImg;
    TextView drawerName;
    Button drawerHome,drawerHelp,drawerAbout,drawerLanguage,drawerQuit;
    @Override
    protected void onCreate(Bundle savedInstanceState) {...}
    private void initiateViews() {...}
    private void initiateListeners() {...}

    @Override
    public void onConfigurationChanged(Configuration newConfig) {...}

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {...}

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {...}


    public void setLocale(String lang) {...}

    @Override
    public void onFragmentInteraction(Uri uri) {...}


    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {...}
}
```

Η Showtime Activity περιέχει τα Fragments Time Picker, Seat Chart και Checkout.

4.2) Fragments

Τα fragments περιέχονται στις δραστηριότητες προκειμένου να ξεχωρίσουμε επιμέρους διαδικασίες που πραγματοποιούνται κατά τη διάρκεια χρήσης του προγράμματος. Αντιπροσωπεύουν ένα κομμάτι της διεπαφής χρήστη σε μια δραστηριότητα. Μπορούμε να συνδυάσουμε πολλαπλά fragments σε μια δραστηριότητα για να δημιουργήσουμε μια διεπαφή χρήστη με πολλά τμήματα καθώς και να χρησιμοποιήσουμε ένα fragment σε πολλαπλές δραστηριότητες. Ένα fragment είναι πάντα ενσωματωμένο σε μια δραστηριότητα και ο κύκλος ζωής του επηρεάζεται άμεσα από αυτόν της δραστηριότητας στην οποία ανήκει. Τα fragments κάθε δραστηριότητας υπάρχουν ως μέρη ενός ViewGroup στην ιεραρχία προβολής της δραστηριότητας και καθορίζουν την δική τους διάταξη. Θα πρέπει όμως να τονίσουμε πως ένα fragment δεν ανήκει απαραίτητα στην διάταξη μιας δραστηριότητας, μπορούν να χρησιμοποιηθούν χωρίς την δική τους διεπαφή χρήστη κάνοντας κάποια εργασία για την δραστηριότητα που ανήκουν στο παρασκήνιο.

4.2.1) Playing Now

Βασική λειτουργία της κλάσης είναι να κατεβάσει την λίστα των ταινιών από τον server και να τις παρουσιάσει στον χρήστη υπό την μορφή λίστας. Οι πληροφορίες κάθε ταινίας υπάρχουν στην κλάση SingleMovie ενώ ο τρόπος εμφάνισης καθορίζεται από την MoviesAdapter.

Η κλάση MoviesAdapter είναι υπεύθυνη για τον τρόπο εμφάνισης της λίστας που περιέχει τα αντικείμενα τύπου SingleMovie. Χρησιμοποιείται προκειμένου να καθορίσει τον τρόπο εμφάνισης των χαρακτηριστικών της κάθε ταινίας μέσω της μεθόδου getView, την θέση δηλαδή του τίτλου, του είδους, των ηθοποιών, του μήκους της ταινίας και της μικρογραφίας της εικόνας που εμφανίζεται στον χρήστη. Σε περίπτωση που η εικόνα που αντιστοιχεί στην ταινία δεν βρεθεί υπάρχει try-catch statement που δείχνει στον χρήστη ότι η εικόνα δεν βρέθηκε.

```
public View getView(int i, View view, ViewGroup viewGroup) {
    //get the root view to find other views
    View movieRow=view;
    if (view==null)
    {
        LayoutInflater inflater=(LayoutInflater)context.getSystemService
            (context.LAYOUT_INFLATER_SERVICE);
        movieRow=inflater.inflate(R.layout.movie_list_item,viewGroup,false);

    }
    TextView movieTitle = (TextView)movieRow.findViewById(R.id.txtTitle);
    TextView movieGenre = (TextView)movieRow.findViewById(R.id.txtGenre);
    TextView movieActors = (TextView)movieRow.findViewById(R.id.txtActr);
    TextView movieLength = (TextView)movieRow.findViewById(R.id.txtLen);
    ImageView imageView=(ImageView)movieRow.findViewById(R.id.imgPicture);
    ImageView image3d=(ImageView)movieRow.findViewById(R.id.img3d);
    SingleMovie tempMovie=list.get(i);
```

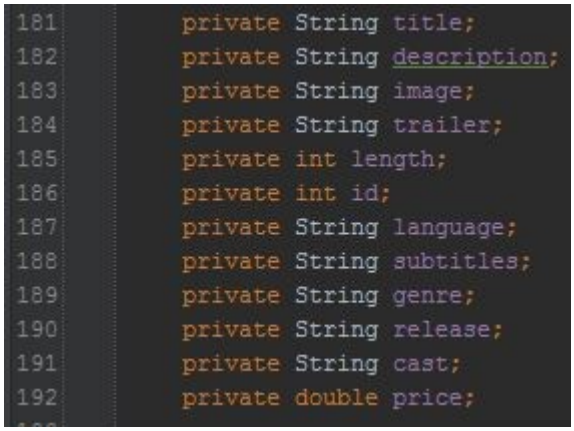
```
        movieTitle.setText("Title:"+tempMovie.getTitle());
        movieActors.setText("Cast:"+tempMovie.getCast());
        int t=tempMovie.getLength();
        int hours = t / 60;
        int minutes = t % 60;
        movieLength.setText("Length:"+hours+"hrs"+minutes+"min");
        movieGenre.setText("Genre:"+tempMovie.getGenre());
        Drawable movieImage;
        if(tempMovie.getPrice()!=12)
        {
            image3d.setVisibility(View.GONE);
        }
        try {
            movieImage = context.getResources().getDrawable(context.getResources().getIdentifier
                (tempMovie.getImage(),"drawable", context.getPackageName()));
        }
        catch (Exception exc)
        {
             movieImage=context.getResources().getDrawable(R.drawable.nopreview);
        }

        imageView.setImageDrawable(resize(movieImage));
        return movieRow;
    }
```

Η getView καλείται κάθε φορά που πρέπει να εμφανιστεί ένα αντικείμενο της λίστας στον χρήστη. Επίσης στην κλάση αυτή υπάρχει και η μέθοδος resize() που χρησιμοποιείται για να αλλάξει το μέγεθος της εικόνας σε 80x90 pixels σε περίπτωση που στην βάση οι εικόνες δεν έχουν όλες το ίδιο μέγεθος.

Η επόμενη κλάση που περιέχει το fragment PlayingNow είναι η SingleMovie. Η κλάση αυτή αναπαριστά ένα αντικείμενο τύπου SingleMovie. Το αντικείμενο αυτό έχει συγκεκριμένες ιδιότητες. Οι περισσότερες από αυτές αφορούν τα χαρακτηριστικά της ταινίας που παρουσιάζονται στον χρήστη, υπάρχουν όμως και κάποιες που χρησιμοποιούνται από το πρόγραμμα. Τα χαρακτηριστικά του αντικειμένου είναι:

181: Τίτλος της ταινίας.
182: Σύντομη περιγραφή της ταινίας.
183: Ένα αλφαριθμητικό που περιέχει την τοποθεσία της εικόνας στον server την οποία θα χρησιμοποιήσει η Java προκειμένου να την βρει και να την εμφανίσει.
184: Ένα αλφαριθμητικό που περιέχει την τοποθεσία του trailer στον server την οποία θα χρησιμοποιήσει η Java προκειμένου να την βρει και να την εμφανίσει.
185: Η διάρκεια της ταινίας.
186: Ο αύξων αριθμός της ταινίας που χρησιμοποιείται σαν αναγνωριστικό από την βάση δεδομένων.
187: Η γλώσσα της ταινίας.
188: Η γλώσσα των υποτίτλων (αν αυτοί υπάρχουν) της ταινίας.
189: Το είδος της ταινίας.
190: Η ημερομηνία κυκλοφορίας της ταινίας.
191: Οι ηθοποιοί που εμφανίζονται στην ταινία.



192: Η τιμή του εισιτηρίου (Η τιμή μπορεί να αλλάζει ανάλογα αν η ταινία είναι 3D ή όχι).

Κάθε μια από τις μεταβλητές έχει τον δικό της getter και τέλος υπάρχει ο δομητής που αρχικοποιεί τα χαρακτηριστικά της ταινίας. Η μεταβλητή bitmap χρησιμοποιείται ξεχωριστά από τις υπόλοιπες γιατί πρώτα πρέπει να ολοκληρωθεί η λήψη της φωτογραφίας από τον server και έπειτα να αποθηκευθεί στην μεταβλητή bitmap για την αποφυγή λαθών κατά την εκτέλεση του προγράμματος.

```
class SingleMovie
{
    private String title;
    private String description;
    private String image;
    private String trailer;
    private int length;
    private int id;
    private String language;
    private String subtitles;
    private String genre;
    private String release;
    private String cast;
    private double price;
    private Bitmap bitmap;

    SingleMovie(int id,String title, String description, String image,String trailer,int length,
                String language,String subtitles,String genre,String release,String cast,double price) {
        this.id=id;
        this.title = title;
        this.description = description;
        this.image = image;
        this.trailer=trailer;
        this.length=length;
        this.cast=cast;
        this.language=language;
        this.subtitles=subtitles;
        this.genre=genre;
        this.release=release;
        this.price=price;
    }
    public String getTitle() { return title; }
    public String getDescription() { return description; }
    public String getImage() { return image; }
    public String getTrailer() { return trailer; }
    public int getLength() { return length; }
    public int getId() { return id; }
    public String getLanguage() { return language; }
    public String getSubtitles() { return subtitles; }
    public String getGenre() { return genre; }
    public String getRelease() { return release; }
    public String getCast() { return cast; }
    public double getPrice() { return price; }
    public Bitmap getBitmap() { return bitmap; }
    public void setBitmap(Bitmap bitmap){this.bitmap=bitmap;}
}
```

Επιστρέφοντας στην κλάση PlayingNow έχουμε την δημιουργία της λίστας μέσω του ArrayList list, την μεταβλητή DOWNLOAD_URL που περιέχει την διαδρομή του php script που χρησιμοποιούμε για να κατεβάσουμε τις ταινίες από τη βάση. Δημιουργούμε ένα αντικείμενο τύπου JSONParser. Αυτό είναι υπεύθυνο για τον τρόπο εμφάνισης του αποτελέσματος του SQL ερωτήματος που τρέχουμε στην βάση μέσω του php script. Τέλος το listview είναι ένα XML αντικείμενο στο οποίο θα εμφανιστούν τα αποτελέσματα μετά την εκτέλεση του php script.

```
ArrayList list = new ArrayList<SingleMovie>();
private  final String DOWNLOAD_URL = "http://nikkouk.zapto.org:1024/webservice/MovieList.php";
JSONParser jsonParser=new JSONParser();
ListView listView;
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    View v= inflater.inflate(R.layout.fragment_playing_now, container, false);
    listView=(ListView)v.findViewById(R.id.listView1);
    Button refresh=(Button)v.findViewById(R.id.ButtonRefresh);
    refresh.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            new DownloadMovies().execute();
        }
    });
    listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
        SingleMovie item=(SingleMovie)listView.getItemAtPosition(i);
            FragmentTransaction fr= getFragmentManager().beginTransaction();
            Bundle bundle = new Bundle();
            bundle.putString("title",item.getTitle());
            bundle.putString("description", item.getDescription());
            bundle.putString("image", item.getImage());
            bundle.putInt("id",item.getId());
            bundle.putString("language",item.getLanguage());
            bundle.putInt("length",item.getLength());
            bundle.putString("cast",item.getCast());
            bundle.putString("subtitles",item.getSubtitles());
            bundle.putString("genre",item.getGenre());
            bundle.putString("release",item.getRelease());
            bundle.putString("trailer",item.getTrailer());
            bundle.putDouble("price",item.getPrice());
            Fragment frag = new MovieDetails();
            frag.setArguments(bundle);
            //replace existing fragment with the new one
                fr.replace(R.id.root_frame_a,frag);
            fr.addToBackStack("replace1");
            fr.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_OPEN);
            fr.addToBackStack(null);
            fr.commit();
        }
    });
    new DownloadMovies().execute();
    return v;
}
```

Στη μέθοδο onCreateView() δημιουργείται η λίστα με τις ταινίες. Το κουμπί refresh χρησιμοποιείται για να κάνουμε ανανέωση της λίστας σε περίπτωση που αποτύχει η φόρτωση. Με την μέθοδο onItemClick() μετακινούμαστε στο fragment MovieDetails. Όταν ο χρήστης επιλέξει κάποιο αντικείμενο της λίστας δημιουργείται το fragment και με την μέθοδο setArguments() δίνουμε στο fragment ένα αντικείμενο τύπου bundle που περιέχει τα χαρακτηριστικά της ταινίας που επιλέχθηκε. Η μετάβαση από το ένα fragment στο άλλο αποθηκεύεται στο backstack το οποίο καθορίζει ποιο fragment θα κληθεί σε περίπτωση που ο χρήστης πατήσει το κουμπί πίσω. Τέλος καλούμε την μέθοδο execute() της κλάσης DownloadMovies μέσω της οποίας γεμίζει η λίστα.

Η υποκλάση DownloadMovies κληρονομεί την AsyncTask. Αυτό γίνεται προκειμένου να κατεβάσουμε την λίστα των ταινιών και να εμφανίσουμε τα αποτελέσματα στο κύριο νήμα της διεπαφής χρήστη. Στην κλάση αυτή περιέχονται τρεις μέθοδοι. Η μέθοδος onPreExecute() μας εμφανίζει το παράθυρο διαλόγου που μας ενημερώνει ότι κατεβάζουμε την λίστα των ταινιών στη συσκευή μας ενώ η μέθοδος onPostExecute() σβήνει το συγκεκριμένο παράθυρο μετά την δημιουργία της λίστας. Η κύρια λειτουργία του γεμίσματος της λίστας με αντικείμενα τύπου SingleMovie γίνεται με την μέθοδο doInBackground().

```
protected String doInBackground(String... strings) {
        String msg="0";
    try {
        List<NameValuePair> params=new ArrayList<NameValuePair>();
        JSONObject jsonObject=jsonParser.makeHttpRequest(DOWNLOAD_URL, "POST",params);
        JSONArray jsonArray=jsonObject.getJSONArray("array");
        msg=jsonObject.getString("message");
        list.clear();
        for(int i=0; i<jsonArray.length(); i++)
        {
            SingleMovie s=new SingleMovie(
                    jsonArray.getJSONObject(i).getInt("id"),
                    jsonArray.getJSONObject(i).getString("title"),
                    jsonArray.getJSONObject(i).getString("description"),
                    jsonArray.getJSONObject(i).getString("image"),
                    jsonArray.getJSONObject(i).getString("trailer"),
                    jsonArray.getJSONObject(i).getInt("length"),
                    jsonArray.getJSONObject(i).getString("language"),
                    jsonArray.getJSONObject(i).getString("subtitles"),
                    jsonArray.getJSONObject(i).getString("genre"),
                    jsonArray.getJSONObject(i).getString("release"),
                    jsonArray.getJSONObject(i).getString("cast"),
                    jsonArray.getJSONObject(i).getDouble("price")
            );
```

Η μέθοδος αυτή δεν τρέχει στο κύριο νήμα της εφαρμογής, αλλά σε ένα βοηθητικό. Δημιουργούμε ένα αντικείμενο τύπου JSONObject το οποίο γεμίζει μέσω ενός αιτήματος διακομιστή. Στο αντικείμενο JSONArray αποθηκεύουμε έναν πίνακα με τα αποτελέσματα του SQL ερωτήματος που τρέξαμε με το php script του οποίου η τοποθεσία βρίσκεται αποθηκευμένη στην μεταβλητή DOWNLOAD_URL. Για κάθε εγγραφή στον πίνακα jsonArray δημιουργείται ένα καινούργιο αντικείμενο τύπου SingleMovie και τοποθετείται στην λίστα με τις ταινίες. Όλα τα χαρακτηριστικά του αντικειμένου εκτός του bitmap αρχικοποιούνται από τον δομητή. Το bitmap αρχικοποιείται μέσω της setBitmap() αφού ολοκληρωθεί η λήψη της φωτογραφίας από τον server. Όλα τα παραπάνω εμπεριέχονται σε ένα block ελέγχου try – catch για την αποφυγή σφαλμάτων που μπορεί να οφείλονται σε λάθος URL της εικόνας, μη απόκρισης του server ή απώλεια πακέτων .

```
                try {
                    Bitmap bitmap1;
                    URL url1 = new URL("http://nikkouk.zapto.org:1024/img/" + s.getImage()
                            + "1.png");
                    HttpGet httpRequest = null;
                    httpRequest = new HttpGet(url1.toURI());
                    HttpClient httpclient = new DefaultHttpClient();
                    HttpResponse response = httpclient.execute(httpRequest);
                    HttpEntity entity = response.getEntity();
                    BufferedHttpEntity b_entity = new BufferedHttpEntity(entity);
                    InputStream input = b_entity.getContent();
                    bitmap1 = BitmapFactory.decodeStream(input);
                    s.setBitmap(bitmap1);
                } catch (MalformedURLException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                } catch (URISyntaxException e) {
                    e.printStackTrace();
                } catch (Exception e) {
                    e.printStackTrace();
                }
                list.add(s);
            }
        }
        catch (JSONException e){
            e.printStackTrace();
        }
        catch(Exception e){
            e.printStackTrace();
        }
        return msg;
    }
```

4.2.2) Login

Το fragment αυτό χρησιμοποιείται προκειμένου να κάνει ο χρήστης είσοδο με τον λογαριασμό του, είτε αυτός είναι λογαριασμός της εφαρμογής είτε λογαριασμός του facebook.

```
public class Login extends Fragment {

    private ProgressDialog pDialog;

    private  final String TAG_SUCCESS = "success";
    private  final String TAG_MESSAGE= "message";
    private  final String LOGIN_URL = "http://nikkouk.zapto.org:1024/webservice/login.php";
    private  final String ARG_SECTION_NUMBER = "section_number";
    Button btnLogin, btnRegister;
    EditText user;
    EditText pass;
    JSONParser jsonParser=new JSONParser();
    public Login() {
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {...}
    private void changeFragment(Fragment fra,int i){...}
    public class AttemptLogin extends AsyncTask<String, String, String> {...}
```

```
    //FACEBOOK LOGIN
    private static final String TAG = "MainFragment";
    private UiLifecycleHelper uiHelper;

    @Override
    public void onCreate(Bundle savedInstanceState) {...}

    @Override
    public void onResume() {...}

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {...}

    @Override
    public void onPause() {...}

    @Override
    public void onDestroy() {...}

    private Session.StatusCallback callback = (session, state, exception) -> {
            onSessionStateChange(session, state, exception);

    };

    @Override
    public void onSaveInstanceState(Bundle outState) {...}
    private void onSessionStateChange(Session session, SessionState state, Exception exception)
}
```

Στην καρτέλα αυτή έχουμε δυο EditText στα οποία ο χρήστης καλείται να γράψει το όνομα και τον κωδικό του λογαριασμού του. Ακόμη έχουμε τρία κουμπιά για είσοδο του χρήστη με λογαριασμό της εφαρμογής, είσοδο με λογαριασμό του facebook ή εγγραφή νέου χρήστη. Ο JSONParser θα χρησιμοποιηθεί αργότερα προκειμένου να στείλουμε στον server το αίτημα σύνδεσης του χρήστη και τα αναγνωριστικά του. Η onCreateView() καλείται κατά την δημιουργία της καρτέλας. Μέσω αυτής ορίζουμε τις ενέργειες που θα πραγματοποιούν τα στοιχεία της XML κατά την αλληλεπίδραση με τον χρήστη. Τι γίνεται δηλαδή κατά το πάτημα ενός κουμπιού. Με το κουμπί btnLogin καλούμε την μέθοδο execute() της κλάσης AttemptLogin, ενώ με το κουμπί btnRegister μετακινούμαστε στο fragment Register μέσω της μεθόδου changeFragment(). Το κουμπί authButton χρησιμοποιεί το API του facebook. Αν είναι η πρώτη φορά που ο χρήστης κάνει log in μέσω facebook, τότε θα του ζητηθεί πρόσβαση για την εφαρμογή από το facebook API στο προφίλ του χρήστη.

```
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    View view=inflater.inflate(R.layout.fragment_login, container, false);
    btnLogin = (Button) view.findViewById(R.id.btnLogin);
    btnRegister = (Button) view.findViewById(R.id.btnRegister);
    user=(EditText)view.findViewById(R.id.txtUsername);
    pass=(EditText)view.findViewById(R.id.txtPassword);
    LoginButton authButton = (LoginButton) view.findViewById(R.id.authButton);
    authButton.setFragment(this);
    authButton.setReadPermissions(Arrays.asList("user_likes", "user_status"));
    btnLogin.setOnClickListener((view) -> { new AttemptLogin().execute(); });
    btnRegister.setOnClickListener((view) -> {
            changeFragment(new Register(), 1);
    });
```

Η κλάση AttemptLogin είναι υπεύθυνη για την αποστολή του αιτήματος εισόδου στην εφαρμογή στον server. Περιέχει τις μεθόδους onPreExecute(), doInBackground() και onPostExecute(). Η πρώτη μας δείχνει το μήνυμα απόπειρας εισόδου ενώ η τρίτη ανάλογα με το αποτέλεσμα που πήραμε μας ενημερώνει για την επιτυχία η αποτυχία εισόδου. Η μέθοδος doInBackground() δημιουργεί μια λίστα με τα περιεχόμενα των EditText. Χρησιμοποιούμε ένα php script προκειμένου να ελέγξουμε εάν τα στοιχεία που δώσαμε αντιστοιχούν σε

κάποιον λογαριασμό που βρίσκεται στην βάση.

```
public class AttemptLogin extends AsyncTask<String, String, String> {

    @Override
    protected void onPreExecute() [...]

    @Override
    protected String doInBackground(String... strings) {
        String msg = "-1";
        try {
            List<NameValuePair> params = new ArrayList<~>();
            String username = user.getText().toString();
            String password = pass.getText().toString();
            params.add(new BasicNameValuePair("username", username));
            params.add(new BasicNameValuePair("password", password));

            JSONObject jsonObject = jsonParser.makeHttpRequest(LOGIN_URL, "POST", params);

            msg = jsonObject.getString(TAG_MESSAGE);


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

        } catch (Exception e) {
            e.printStackTrace();
        }
        return msg;
    }

    protected void onPostExecute(String msg) [...]

}
```

Σε περίπτωση που ο χρήστης δεν έχει λογαριασμό στην εφαρμογή μπορεί να επιλέξει να δημιουργήσει έναν. Τότε πατώντας το κουμπί Register της προηγούμενης καρτέλας φορτώνεται το fragment Register. Στο fragment αυτό ο χρήστης καλείται να δώσει τα στοιχεία του σε κάποια EditText όπως αυτά που χρησιμοποιήσαμε προηγουμένως. Η κλάση Register περιέχει τις μεθόδους onCreateView(), initiateViews(), InitiateListeners() και changeFragment() καθώς και την ασύγχρονη κλάση AttemptRegister. Η μέθοδος onCreateView() είναι υπεύθυνη για την δημιουργία της καρτέλας Register. Με την μέθοδο initiateViews() φτιάχνουμε τα EditText και το κουμπί που περιέχει η καρτέλα ενώ μέσω της μεθόδου InitiateListeners() δίνουμε λειτουργικότητα στο κουμπί Register καθώς και κάνουμε επικύρωση των στοιχείων που έδωσε ο χρήστης, ελέγχουμε δηλαδή εάν συμπλήρωσε όλα τα πεδία. Η μέθοδος changeFragment() είναι υπεύθυνη για την

μετακίνηση μεταξύ του Register και του Login fragment.

```
public class Register extends Fragment {

    private ProgressDialog pDialog;
    public Register() {...}


    private  final String TAG_MESSAGE= "message";
    private  final String LOGIN_URL = "http://nikkouk.zapto.org:1024/webservice/Register.php";
    Button but;
    EditText user,pass,pass2,email,name,surname;
    JSONParser jsonParser=new JSONParser();
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {...}

    private void initiateViews(View view) {...}
    private void initiateListeners() {...}

    private void changeFragment(Fragment fra){...}
    public class AttemptRegister extends AsyncTask<String, String, String> {...}
}
```

Η κλάση AttemptRegister είναι υπεύθυνη για την αποστολή του αιτήματος εγγραφής λογαριασμού στην εφαρμογή στον server. Περιέχει τις μεθόδους onPreExecute(), doInBackground() και onPostExecute(). Η πρώτη μας δείχνει το μήνυμα απόπειρας εγγραφής ενώ η τρίτη ανάλογα με το αποτέλεσμα που πήραμε μας ενημερώνει για την επιτυχία η αποτυχία εγγραφής, καθώς και την αιτία της αποτυχίας, π.χ. σφάλμα στην βάση, λανθασμένα στοιχεία κ.λπ. Η μέθοδος doInBackground() δημιουργεί μια λίστα με τα περιεχόμενα των EditText. Έπειτα τρέχουμε το Register.php που περιέχει τα στοιχεία που έδωσε ο χρήστης και στέλνει στην βάση αίτημα νέας εγγραφής χρήστη υπό τη μορφή SQL-Insert.

```
    public class AttemptRegister extends AsyncTask<String, String, String> {
        @Override
        protected void onPreExecute() {...}
        @Override
        protected String doInBackground(String... strings) {
            String msg=null;
            try {
                List<NameValuePair> params=new ArrayList<~>();
                String username = user.getText().toString();
                String password = pass.getText().toString();
                String emailString = email.getText().toString();
                String nameString = name.getText().toString();
                String surnameString = surname.getText().toString();
                params.add(new BasicNameValuePair("username", username));
                params.add(new BasicNameValuePair("password", password));
                params.add(new BasicNameValuePair("email", emailString));
                params.add(new BasicNameValuePair("name", nameString));
                params.add(new BasicNameValuePair("surname", surnameString));
                JSONObject jsonObject=jsonParser.makeHttpRequest(REGISTER_URL, "POST", params);
                msg = jsonObject.getString(TAG_MESSAGE);
            }
            catch (JSONException e){
                e.printStackTrace();
            }
            catch(Exception e){
                e.printStackTrace();
            }
            return msg;
        }
        protected void onPostExecute(String msg) {...}
    }
}
```

Η τελευταία καρτέλα που ανήκει στην ομάδα του Login την οποία θα μελετήσουμε είναι η καρτέλα του User CP. Το fragment UserCP φορτώνεται μετά από επιτυχή είσοδο του χρήστη με κάποιον λογαριασμό. Σε ένα πεδίο τύπου TextView εμφανίζεται ο λογαριασμός του χρήστη. Υπάρχει κουμπί για έξοδο από τον λογαριασμό ανάλογα με το αν ο λογαριασμός είναι της εφαρμογής ή του facebook ενώ τέλος υπάρχει ένα κουμπί με το οποίο ενεργοποιείται ένα TableLayout στο οποίο ο χρήστης μπορεί να εισάγει τα στοιχεία της πιστωτικής του κάρτας τα οποία και θα μπορεί αργότερα να συμπληρώσει αυτόματα όταν θα πρέπει να επιλέξει τρόπο πληρωμής.

```
public class UserCp extends Fragment {

    private String username;
    private Boolean facebook;
    TextView textView;
    Button btnLogOut,btnAccept;
    ToggleButton btnCreditCard;
    TableLayout tableLayout;
    LoginButton loginButton;
    EditText creditNo,cv2,expDate,cardName;
    public UserCp() {...}
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {...}

    private void initiateViews(View view) {...}

    private void initiateListeners() {...}

    private void changeFragment(Fragment fra){...}
}
```

Η κλάση UserCP περιέχει τις μεθόδους onCreateView(), initiateViews(), InitiateListeners() και changeFragment(). Η μέθοδος onCreateView() είναι υπεύθυνη για την δημιουργία της καρτέλας Register. Με την μέθοδο initiateViews() φτιάχνουμε τα EditText του TableLayout στο οποίο ο χρήστης δίνει στοιχεία πιστωτικής κάρτας και το κουμπί που περιέχει η καρτέλα, τα οποία παίρνουν τιμές σε περίπτωση που ο χρήστης έχει ήδη δώσει σε προηγούμενη περίπτωση τα στοιχεία του. Μέσω της μεθόδου InitiateListeners() δίνουμε λειτουργικότητα στα κουμπιά btnCreditCard , btnLogOut και btnAccept. Το btnCreditCard μας εμφανίζει το TableLayout με τα στοιχεία της κάρτας, ενώ το κουμπί btnAccept σώζει τις πληροφορίες της πιστωτικής του που έδωσε ο χρήστης . Το κουμπί btnLogOut εμφανίζεται μετά από έλεγχο που κάνουμε για το αν ο χρήστης είναι συνδεδεμένος με λογαριασμό της εφαρμογής. Θα πρέπει σε αυτό το σημείο να αναφέρουμε πως σε περίπτωση που είμαστε συνδεδεμένοι με λογαριασμό του facebook το API του facebook μετατρέπει αυτόματα τη λειτουργικότητα του loginButton σε κουμπί εξόδου, αλλάζοντας παράλληλα και την εμφάνιση του. Η μέθοδος changeFragment() είναι υπεύθυνη για την μετακίνηση μεταξύ του UserCP και του Login fragment.

```
private void initiateListeners() {
    btnCreditCard.setOnClickListener((view) -> {
            tableLayout.setVisibility(tableLayout.isShown() ? View.GONE : View.VISIBLE);

    });
    if(username!="Database Error.Please log in again!" && !facebook)
    {
        btnLogOut.setVisibility(View.VISIBLE);
    }
    if( username!="Database Error.Please log in again!" && facebook)
    {
        loginButton.setVisibility(View.VISIBLE);
    }
    btnLogOut.setOnClickListener((view) -> {
            PreferenceManager.getDefaultSharedPreferences(getActivity()).edit()
                    .remove("username").commit();
            ((TextView)getActivity().findViewById(R.id.drawer_name)).setText("Guest");
            changeFragment(new Login());
    });
    btnAccept.setOnClickListener((v) -> {
            PreferenceManager.getDefaultSharedPreferences(getActivity()).edit().putString
                    ("creditNo",creditNo.getText().toString()).commit();
            PreferenceManager.getDefaultSharedPreferences(getActivity()).edit().putString
                    ("cv2",cv2.getText().toString()).commit();
            PreferenceManager.getDefaultSharedPreferences(getActivity()).edit().putString
                    ("cardName",cardName.getText().toString()).commit();
            PreferenceManager.getDefaultSharedPreferences(getActivity()).edit().putString
                    ("expDate",expDate.getText().toString()).commit();
            Toast.makeText(getActivity(), getResources().getString(R.string.changes_done),
                    Toast.LENGTH_SHORT);
            tableLayout.setVisibility(tableLayout.isShown() ? View.GONE : View.VISIBLE);
    });
}
```

4.2.3) Location

Το fragment αυτό χρησιμοποιείται προκειμένου να εμφανίσει την τοποθεσία του κινηματογράφου που θα επιλέξει ο χρήστης. Στην καρτέλα αυτή έχουμε τρία κουμπιά το καθένα από τα οποία αντιστοιχεί σε διαφορετικό κινηματογράφο και έναν χάρτη που παίρνουμε μέσω του API του Google Maps.

```
public class Location extends Fragment {
    MapView m;
    Button button,button2,button3;

    public Location() {...}
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {...}
    void initiateViews(View view)
    {...}
    void initiateListeners()
    {...}
    @Override
    public void onResume() {...}
    @Override
    public void onPause() {...}
    @Override
    public void onDestroy() {...}
    @Override
    public void onLowMemory() {...}
    @Override
    public void setUserVisibleHint(boolean isVisibleToUser) {...}
}
```

Η κλάση Location περιέχει τις μεθόδους onCreateView(), InitiateViews(), InitiateListeners() onResume(), onPause(), onDestroy(), onLowMemory(), setUserVisibleHint(). Οι μέθοδοι onResume(), onPause(), onDestroy(), onLowMemory(), υπάρχουν σε όλα τα fragments όμως εδώ τις υλοποιούμε για να δώσουμε επιπλέον λειτουργικότητα αναφορικά με τον χάρτη που υπάρχει στην συγκεκριμένη καρτέλα. Η μέθοδος onCreateView() δημιουργεί την καρτέλα Location. Η initiateViews() δημιουργεί τον χάρτη καθώς και τα κουμπιά που αντιστοιχούν σε κάθε κινηματογράφο, ενώ η InitiateListeners() είναι υπεύθυνη για την λειτουργικότητα των κουμπιών. Σε αυτήν καθορίζουμε τις συντεταγμένες κάθε κινηματογράφου και έπειτα κάθε κουμπί αντιστοιχίζεται σε ένα σύνολο συντεταγμένων. Όταν το κουμπί πατηθεί εισάγεται μια νέα πινέζα στην τοποθεσία του χάρτη με τις αντίστοιχες συντεταγμένες ενώ ο χάρτης εστιάζει στην συγκεκριμένη τοποθεσία. Τέλος η μέθοδος setUserVisibleHint() χρησιμοποιείται για να «κλειδώσουμε» την λειτουργία του Navigation Drawer της εφαρμογής. Με την μέθοδο αυτή όταν βρισκόμαστε στην καρτέλα Location δεν εμφανίζεται ο Navigation Drawer σε περίπτωση που κάνουμε swipe δεξιά. Αυτό γίνεται για να μην επηρεαστεί η λειτουργικότητα του χάρτη, δεν θέλουμε όταν ο χρήστης προσπαθεί να μετακινηθεί στον χάρτη να εμφανίζεται ο Navigation Drawer. Στο σημείο αυτό θα πρέπει να αναφέρουμε ότι η λειτουργικότητα στον χάρτη

προσδίδεται εισάγοντας τις βιβλιοθήκες του Google Maps API.

```
void initiateListeners()
{
    final LatLng CINEMA1 = new LatLng(40.580528, 22.980158);
    final LatLng CINEMA2 = new LatLng(40.556969, 22.992363);
    final LatLng CINEMA3 = new LatLng(40.633104, 22.942745);
    final GoogleMap map=m.getMap();
    button.setOnClickListener((v) -> {
            map.clear();
            map.addMarker(new MarkerOptions().position(CINEMA1).title("Cinema1"));
            map.animateCamera(CameraUpdateFactory.newLatLngZoom(CINEMA1, 17));

    });
    button2.setOnClickListener((v) -> {
            map.clear();
            map.addMarker(new MarkerOptions().position(CINEMA2).title("Cinema2"));
            map.animateCamera(CameraUpdateFactory.newLatLngZoom(CINEMA2, 16));
    });
    button3.setOnClickListener((v) -> {
            map.clear();
            map.addMarker(new MarkerOptions().position(CINEMA3).title("Cinema3"));
            map.animateCamera(CameraUpdateFactory.newLatLngZoom(CINEMA3, 18));
    });
}
```

```
public void setUserVisibleHint(boolean isVisibleToUser) {
    super.setUserVisibleHint(isVisibleToUser);
    if(getView()!=null) {
        DrawerLayout mDrawerLayout;
        mDrawerLayout = (DrawerLayout) getActivity().findViewById(R.id.drawerLayout);

        if (isVisibleToUser) {
            mDrawerLayout.setDrawerLockMode(DrawerLayout.LOCK_MODE_LOCKED_CLOSED);
        } else {
            mDrawerLayout.setDrawerLockMode(DrawerLayout.LOCK_MODE_UNLOCKED);
        }
    }
}
```

4.2.4) Booking History

Βασική λειτουργία της κλάσης είναι να κατεβάσει την λίστα των ταινιών από τον server και να τις παρουσιάσει στον χρήστη υπό την μορφή λίστας. Οι πληροφορίες κάθε ταινίας υπάρχουν στην κλάση SingleBooking ενώ ο τρόπος εμφάνισης καθορίζεται από την HistoryAdapter.

Η κλάση HistoryAdapter είναι υπεύθυνη για τον τρόπο εμφάνισης των λιστών που περιέχουν τα αντικείμενα τύπου SingleBooking. Χρησιμοποιείται προκειμένου να καθορίσει τον τρόπο εμφάνισης λεπτομερειών της κάθε κράτησης μέσω της μεθόδου getView, το όνομα της ταινίας που αντιστοιχεί στην κράτηση,

την ημερομηνία και τον κωδικό της κράτησης.

```
public View getView(int i, View view, ViewGroup viewGroup) {
    View movieRow=view;
    if (view==null)
    {
        LayoutInflater inflater=(LayoutInflater)context.getSystemService
                (context.LAYOUT_INFLATER_SERVICE);
        movieRow=inflater.inflate(R.layout.history_list_item,viewGroup,false);
    }
    TextView movieTitle = (TextView)movieRow.findViewById(R.id.textTitle);
    TextView movieDate = (TextView)movieRow.findViewById(R.id.textDate);
    TextView movieResId = (TextView)movieRow.findViewById(R.id.textResId);
    SingleBooking tempBooking=list.get(i);
    movieTitle.setText(tempBooking.getTitle());
    movieResId.setText(Integer.toString(tempBooking.getResId()));
    movieDate.setText(tempBooking.getDate());
    return movieRow;
}
```

Η επόμενη κλάση που περιέχει το fragment BookingHistory είναι η SingleBooking. Η κλάση αυτή αναπαριστά ένα αντικείμενο τύπου SingleBooking. Το αντικείμενο αυτό έχει συγκεκριμένες ιδιότητες. Αυτές αφορούν τις λεπτομέρειες της κράτησης που παρουσιάζονται στον χρήστη. Οι λεπτομέρειες αυτές είναι οι εξής:

422: Το όνομα του κινηματογράφου.
423: Τίτλος της ταινίας.
424: Ημερομηνία της κράτησης.
425: Ο κωδικός της κράτησης (χρειάζεται προκειμένου ο χρήστης να παραλάβει τα εισιτήρια του από το ταμείο του κινηματογράφου.

```
422         private String cinema;
423         private String title;
424         private String date;
425         private int id;
426         private int seat;
```

426: Οι θέσεις που έχει κρατήσει ο χρήστης.

```
class SingleBooking
{
    private String cinema;
    private String title;
    private String date;
    private int id;
    private int seat;
    SingleBooking(int id, String title, String date,int seat,String cinema) {
        this.cinema=cinema;
        this.id=id;
        this.title = title;
        this.date=date;
        this.seat=seat;
    }
    @Override
    public int hashCode() { return id; }
    @Override
    public boolean equals(Object o) {...}
    public String getTitle() { return title; }
    public String getCinema() { return cinema; }
    public String getDate() { return date; }
    public int getSeat() { return seat; }
    public int getResId() { return id; }
}
```

Επιστρέφοντας στην κλάση BookingHistory δηλώνουμε μια μεταβλητή τύπου ProgressDialog που χρησιμοποιούμε στην μέθοδο onPreExecute() της υποκλάσης DownloadHistory για να δηλώσει πως φορτώνεται το ιστορικό κρατήσεων, τις μεταβλητές username, selectedMovie, selectedCinema και selectedDate οι οποίες χρησιμοποιούνται επίσης από τις μεθόδους της υποκλάσης DownloadHistory, δημιουργούμε μια λίστα τύπου ArrayList που περιέχει αντικείμενα τύπου SingleBooking καθώς και δυο Listview στα οποία θα χωριστούν οι κρατήσεις της πρώτης λίστας ανάλογα με το αν η κράτηση είναι ενεργή ή έχει παρέλθει η ημερομηνία της. Ακόμη δημιουργούμε ένα αντικείμενο τύπου JSONParser. Αυτό είναι υπεύθυνο για τον τρόπο εμφάνισης του αποτελέσματος του SQL ερωτήματος που τρέχουμε στην βάση μέσω του php script. Τέλος δημιουργούμε ορισμένες μεταβλητές που θα μας χρειαστούν για την δημοσίευση της κράτησης στο facebook. Όπως και στα υπόλοιπα fragments μας η καρτέλα δημιουργείται με την μέθοδο onCreateView() η οποία είναι επίσης υπεύθυνη στην περίπτωση αυτή να τρέξει το Php script που με τη σειρά του θα τρέξει το SQL ερώτημα στη βάση για να μας εμφανίσει –αν αυτό υπάρχει- το ιστορικό κρατήσεων του χρήστη. Η onCreateView() δίνει ακόμη τιμή στις κενές λίστες τύπου listView1 και listView2 ενώ με τη χρήση της μεθόδου makeMyScrollSmart επιτρέπει τις λίστες να είναι Scrollable όταν ο χρήστης χρησιμοποιεί το scroll μέσα σε μια λίστα

και όχι σε ολόκληρη την καρτέλα. Η μέθοδος requestDisallowParentInterceptTouchEvent() χρησιμοποιείται προκειμένου να απενεργοποιήσουμε την ιδιότητα της κύλισης στον γονέα της λίστας την οποία αγγίζει ο χρήστης την συγκεκριμένη στιγμή.

```
public class BookingHistory extends Fragment {
    ProgressDialog pDialog;
    private String username,selectedMovie="",selectedCinema="",selectedDate="";
    ArrayList list = new ArrayList<SingleBooking>();
    private String BOOKINGHSTR_URL;
    JSONParser jsonParser=new JSONParser();
    private ListView listView1;
    private ListView listView2;
    private TextView noBooking;
    AlertDialog.Builder builder;
    Session session;
    private static final List<String> PERMISSIONS = Arrays.asList("publish_actions");
    protected boolean pendingPublishReauthorization ;

    public BookingHistory() {...}

    @Override
    public void setUserVisibleHint(boolean isVisibleToUser) {...}

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {...}

    private void makeMyScrollSmart(final ListView listView) {...}

    private void requestDisallowParentInterceptTouchEvent
            (View __v, Boolean __disallowIntercept) {...}
```

Η υποκλάση DownloadHistory όπως και η DownloadMovies που μελετήσαμε παραπάνω κληρονομεί την AsyncTask και περιέχει τις μεθόδους onPreExecute(), doInBackground() και onPostExecute(). Η πρώτη καθαρίζει το περιεχόμενο της καρτέλας σε περίπτωση που δεν είναι η πρώτη φορά που φορτώνουμε το ιστορικό και εμφανίζει το μήνυμα φόρτωσης. Η μέθοδος doInBackground() έχει πολλαπλές λειτουργίες. Αρχικά ελέγχει εάν ο χρήστης είναι συνδεδεμένος στην εφαρμογή με κάποιο λογαριασμό ή όχι. Σε περίπτωση που δεν είναι συνδεδεμένος χρησιμοποιούμε το όνομα χρήστη guest, μόνο που χρειάζεται να πάρουμε τα αποτελέσματα που αφορούν μόνο την δική μας συσκευή. Στέλνουμε στον SQL server ένα αίτημα για να παραλάβουμε όλες τις κρατήσεις των οποίων τον μοναδικό κωδικό παίρνουμε από τον PreferenceManager. Μέσω αυτού αποκτούμε πρόσβαση στην μνήμη του κινητού όπου αποθηκεύονται οι κρατήσεις σε περίπτωση που ο χρήστης δεν είναι συνδεδεμένος με κάποιο λογαριασμό. Τα αποτελέσματα που λαμβάνουμε μετά με την χρήση JSONObject τα εισάγουμε σε λίστα.

```
if (username.equals("guest")) {
    msg="guest";
    //User NOT logged in
    int count=0;
    String message="";
    boolean found=false;
    while( PreferenceManager.getDefaultSharedPreferences(getActivity()).getInt
            ("paymentid"+count,-1)!=-1)
    {
        found=true;
        message+=Integer.toString(PreferenceManager.getDefaultSharedPreferences
                (getActivity()).getInt("paymentid"+count,-1));
        message+=",";
        count++;
    }
    if(found) {
        visi=false;
        BOOKINGHSTR_URL = "http://nikkouk.zapto.org:1024/webservice/GuestBookingHistory.php";
        List<NameValuePair> params = new ArrayList<NameValuePair>();
        params.add(new BasicNameValuePair("username", username));
        message=message.substring(0, message.length() - 1);
        params.add(new BasicNameValuePair("count", Integer.toString(count)));
        params.add(new BasicNameValuePair("message", message));
        JSONObject jsonObject = jsonParser.makeHttpRequest(BOOKINGHSTR_URL, "POST", params);
        JSONArray jsonArray = jsonObject.getJSONArray("array");
        msg = jsonObject.getString("message");
        for (int i = 1; i < jsonArray.length(); i++) {

            list.add(new SingleBooking(jsonArray.getJSONObject(i).getInt("id"),
                    jsonArray.getJSONObject(i).getString("title"),
                    jsonArray.getJSONObject(i).getString("time"),
                    jsonArray.getJSONObject(i).getInt("seat"),
                    jsonArray.getJSONObject(i).getString("name")));
        }
    }
```

Σε περίπτωση που ο χρήστης είναι συνδεδεμένος στέλνουμε ένα αίτημα ανάκτησης των κρατήσεων που αντιστοιχούν στον συγκεκριμένο χρήστη και πάλι μέσω του JSONObject τυποποιούμε τα αποτελέσματα με τη μορφή λίστας.

```
else {
  //User logged in
  BOOKINGHSTR_URL = "http://nikkouk.zapto.org:1024/webservice/UserBookingHistory.php";
  List<NameValuePair> params = new ArrayList<NameValuePair>();
  params.add(new BasicNameValuePair("username", username));


  JSONObject jsonObject = jsonParser.makeHttpRequest(BOOKINGHSTR_URL, "POST", params);
  assert jsonObject != null;
  JSONArray jsonArray = jsonObject.getJSONArray("array");
  msg = jsonObject.getString("message");
  if(jsonArray.length()>1)
  {
      visi=false;
  }
  for (int i = 1; i < jsonArray.length(); i++) {

      list.add(new SingleBooking(jsonArray.getJSONObject(i).getInt("id"),
              jsonArray.getJSONObject(i).getString("title"),
              jsonArray.getJSONObject(i).getString("time"),
              jsonArray.getJSONObject(i).getInt("seat"),
              jsonArray.getJSONObject(i).getString("name")));
      Log.e("a", jsonArray.getJSONObject(i).toString());
  }
```

Η μέθοδος onPostExecute() διαγράφει το μήνυμα φόρτωσης του ιστορικού (καθώς έχει ήδη γίνει η φόρτωση). Ακόμη χρησιμοποιείται προκειμένου να ξεχωρίσει τις κρατήσεις που πήραμε κατά την εκτέλεση της doInBackground() σε δυο ξεχωριστές λίστες ανάλογα με το αν είναι ενεργές η όχι. Αυτό γίνεται χρησιμοποιώντας το ρολόι της συσκευής. Δημιουργούμε μια λίστα στην οποία εισάγονται αρχικά όλα τα αποτελέσματα και έπειτα τα συγκρίνουμε ένα προς ένα με την ημερομηνία της συσκευής. Σε περίπτωση που η ημερομηνία της κράτησης έχει παρέλθει το στοιχείο μπαίνει στην λίστα του ιστορικού ενώ στην αντίθετη περίπτωση εισάγεται στην λίστα των ενεργών κρατήσεων.

```
if (msg != "0") {
    final Set<String> setString = new LinkedHashSet<String>(list);
    setString.addAll(list);
    ArrayList hashlist = new ArrayList<SingleBooking>();
    hashlist.addAll(setString);
    ArrayList hashlistPending = new ArrayList<SingleBooking>();
    ArrayList hashlistOld = new ArrayList<SingleBooking>();
    String  date= new SimpleDateFormat("yyyy-MM-dd HH:mm").format(new Date());
    // group bookings to pending and old
    for(int i=0; i<hashlist.size(); i++)
    {
        SingleBooking s=(SingleBooking)hashlist.get(i);
        SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm");
        try {
            Date nowFate = format.parse(date);
            Date bookingDate = format.parse(s.getDate().toString());
            if(bookingDate.compareTo(nowFate)>0){hashlistPending.add(s);}
            else{hashlistOld.add(s);}
        }
        catch (ParseException e){e.printStackTrace();}
    }

    listView1.setAdapter(new HistoryAdapter(getActivity().getApplicationContext(), hashlistPending));
    listView2.setAdapter(new HistoryAdapter(getActivity().getApplicationContext(), hashlistOld));
```

Μια ακόμη λειτουργία της onPostExecute() είναι η δημιουργία του μηνύματος που εμφανίζεται όταν ο χρήστης επιλέξει κάποια κράτηση. Αυτό γίνεται με την χρήση της μεθόδου onItemClick().

```
public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
    SingleBooking singleBooking = (SingleBooking) listView1.getItemAtPosition(i);
    int resId = singleBooking.getResId();
    selectedCinema=singleBooking.getCinema();
    selectedMovie=singleBooking.getTitle();
    selectedDate=singleBooking.getDate();
    String seatsMessage = getResources().getString(R.string.seats);
    for (int j = 0; j < list.size(); j++) {
        SingleBooking tempBooking = ((SingleBooking) list.get(j));
        if (tempBooking.getResId() == resId) {
            seatsMessage += (tempBooking.getSeat() + ",");
        }
    }
    builder = new AlertDialog.Builder(getActivity());
    builder.setTitle(getResources().getString(R.string.reservation_number) + Integer.toString(singleBooking.getResId()));
    builder.setMessage(seatsMessage.substring(0, seatsMessage.length() - 1));
    String dialogString="Login with Facebook to share";
    if(session!=null) {
        if (session.isOpened()) dialogString = "Share To Facebook";
    }
    builder.setNeutralButton(dialogString,new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            if(session.isOpened())
            {
                Toast.makeText(getActivity(),getResources().getString(R.string.publish_req),Toast.LENGTH_SHORT);
                publishStory();
            }
            else if(session.isClosed()) dialog.dismiss();
        }
    });
    builder.show();
}
```

Σε περίπτωση που η κράτηση είναι στην λίστα των ενεργών κρατήσεων στον χρήστη εμφανίζεται ένα μήνυμα με τον κωδικό της κράτησης και τους αριθμούς των θέσεων στις οποίες αυτή αντιστοιχεί. Επιπλέον ελέγχεται εάν ο χρήστης είναι

συνδεδεμένος με λογαριασμό του facebook. Εάν ο χρήστης είναι συνδεδεμένος εμφανίζεται επίσης επιλογή ανάρτησης της κράτησης στο facebook, ενώ αν δεν είναι εμφανίζεται μήνυμα προτροπής εισόδου στο facebook για να μοιραστεί την κράτηση. Στην περίπτωση που η κράτηση είναι στην λίστα του ιστορικού και όχι στην λίστα των ενεργών κρατήσεων εμφανίζεται απλά μήνυμα με τον κωδικό της κράτησης και τις θέσεις στις οποίες αντιστοιχούσε. Όπως αναφέραμε εάν ο χρήστης είναι συνδεδεμένος στον λογαριασμό του στο facebook μπορεί να κάνει ανάρτηση της κράτησής του.

Εάν ο χρήστης επιλέξει να μοιραστεί την κράτηση του καλείται η μέθοδος publishStory(). Η μέθοδος αυτή ελέγχει εάν η εφαρμογή έχει τις απαιτούμενες άδειες από το API του facebook και σε περίπτωση που δεν τις έχει τις ζητά. Έπειτα δημιουργεί μια δέσμη με δεδομένα (το όνομα της ταινίας, του κινηματογράφου, της περιγραφής, του χρήστη που έκανε την κράτηση και μια εικόνα της ταινίας) και στέλνει ένα αίτημα για δημοσίευση στο facebook.

4.2.5) Movie Details

Βασική λειτουργία της κλάσης είναι να εμφανίσει στον χρήστη τις λεπτομέρειες της ταινίας που έχει επιλέξει, καθώς και το πρόγραμμα προβολών της ταινίας από το οποίο μπορεί να επιλέξει ο χρήστης πότε θέλει να παρακολουθήσει την ταινία. Η κλάση περιέχει τις υποκλάσεις RetrieveData , DownloadShowtime και CalendarDialog . Η κλάση RetrieveData είναι υπεύθυνη

για το κατέβασμα των φωτογραφιών από τον server.

```
class Retrievedata extends AsyncTask<String, Void, String> {
    Bitmap bitmap1;
    Bitmap bitmap2;
    Bitmap bitmap3;
    @Override
    protected String doInBackground(String... params) {
        try {
            URL url1 = new URL("http://nikkouk.zapto.org:1024/img/"+image+"1.png");
            URL url2 = new URL("http://nikkouk.zapto.org:1024/img/"+image+"2.png");
            URL url3 = new URL("http://nikkouk.zapto.org:1024/img/"+image+"3.png");
            HttpGet httpRequest = null;
            httpRequest = new HttpGet(url1.toURI());
            HttpClient httpclient = new DefaultHttpClient();
            HttpResponse response = httpclient.execute(httpRequest);
            HttpEntity entity = response.getEntity();
            BufferedHttpEntity b_entity = new BufferedHttpEntity(entity);
            InputStream input = b_entity.getContent();
            bitmap1 = BitmapFactory.decodeStream(input);
            httpRequest = new HttpGet(url2.toURI());
            response = httpclient.execute(httpRequest);
            entity = response.getEntity();
            b_entity = new BufferedHttpEntity(entity);
            input = b_entity.getContent();
            bitmap2 = BitmapFactory.decodeStream(input);
            httpRequest = new HttpGet(url3.toURI());
            response = httpclient.execute(httpRequest);
            entity = response.getEntity();
            b_entity = new BufferedHttpEntity(entity);
            input = b_entity.getContent();
            bitmap3 = BitmapFactory.decodeStream(input);
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (URISyntaxException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
```

Η υποκλάση DownloadShowtime όπως και η DownloadMovies που μελετήσαμε παραπάνω κληρονομεί την AsyncTask και περιέχει τις μεθόδους onPreExecute(), doInBackground() και onPostExecute(). Είναι υπεύθυνη για την εμφάνιση και απόκρυψη του μηνύματος φόρτωσης της λίστας προβολών της ταινίας και με την μέθοδο doInBackground() φορτώνονται στην λίστα οι

λεπτομέρειες της κάθε προβολής (κινηματογράφος, αίθουσα, ημερομηνία, ώρα προβολής).

```
protected String doInBackground(String... strings) {
    String msg="0";
    try {
        List<NameValuePair> params=new ArrayList<~>();
        params.add(new BasicNameValuePair("movieId",String.valueOf(id)));
        params.add(new BasicNameValuePair("price",String.valueOf(price)));
        JSONObject jsonObject=jsonParser.makeHttpRequest(
                "http://nikkouk.zapto.org:1024/webservice/Showtime.php", "POST",params);
        JSONArray jsonArray=jsonObject.getJSONArray("array");
        dateArray=new String[jsonArray.length()];
        roomArray=new String[jsonArray.length()];
        timeArray=new String[jsonArray.length()];
        showIdArray=new String[jsonArray.length()];
        cinemaArray=new String[jsonArray.length()];
        btnCalendar.setText(jsonArray.getJSONObject(0).getString("time").
                substring(0,10)+"(Change Date)");
        for(int i=0; i<jsonArray.length(); i++) {
            dateArray[i]=jsonArray.getJSONObject(i).getString("time").substring(0,10);
            timeArray[i]=jsonArray.getJSONObject(i).getString("time").substring(11,16);
            cinemaArray[i]=jsonArray.getJSONObject(i).getString("name");
            roomArray[i]=jsonArray.getJSONObject(i).getString("room_id");
            showIdArray[i]=jsonArray.getJSONObject(i).getString("id");
        }
        msg=jsonObject.getString("message");
    }
    catch (JSONException e){
        e.printStackTrace();
    }
    catch(Exception e){
        e.printStackTrace();
    }
    return msg;
}
```

Η κλάση CalendarDialog μας εμφανίζει ένα παράθυρο διαλόγου με μια λίστα με τις ημερομηνίες προβολής της ταινίας.

```
public static class CalendarDialog extends DialogFragment {
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        final AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
        try {
            String array[] = getArguments().getStringArray("array");
            ArrayList<String> arrayList= new ArrayList<~>(Arrays.asList(array));
            Set<String> setString = new LinkedHashSet<~>(arrayList);
            setString.addAll(arrayList);
            arrayList.clear();
            arrayList.addAll(setString);
            final String[] array2=  arrayList.toArray(new String[arrayList.size()]);
            // Use the Builder class for convenient dialog construction
            if (array2 != null) {
                builder.setItems(array2, (dialogInterface, i) -> {
                        dialogValue=array2[i];
                        btnCalendar.setText(array2[i].toString().substring(0,10)+"(Change Date)"
                        createShowtimeButtons(getActivity());
                        dialogInterface.dismiss();
                    });
            }
        catch (Exception exc)
        {builder.setMessage("Error!No showtime found!");}
        return builder.create();
    }
```

Η κλάση MovieDetails περιέχει τις μεθόδους onCreateView(), initiateViews() και InitiateListeners() που μελετήσαμε σε προηγούμενες κλάσεις. Μέσω αυτών των κλάσεων δημιουργείτε το περιβάλλον της καρτέλας, ενώ παίρνουν λειτουργικότητα και τα διάφορα κουμπιά. Εδώ θα πρέπει να αναφέρουμε την χρήση του YouTube Player API για την εμφάνιση του trailer κάθε ταινίας. Στην μέθοδο initiateViews() δημιουργούμε το fragment το οποίο περιέχει τον YouTube Player. Η onInitializationSuccess() εκτελείται κατά την επιτυχή αρχικοποίηση του player, ενώ η onInitializationFailure() σε περίπτωση αποτυχίας.

```
btnYoutube=(Button) view.findViewById(R.id.btnYoutube);
childFragment = YouTubePlayerSupportFragment.newInstance();
childFragment.initialize(key,new YouTubePlayer.OnInitializedListener() {

    @Override
    public void onInitializationSuccess(YouTubePlayer.Provider provider,
    YouTubePlayer youTubePlayer, boolean b)
    {youTubePlayer.loadVideo(trailer); }
    @Override
    public void onInitializationFailure(YouTubePlayer.Provider provider,
    YouTubeInitializationResult youTubeInitializationResult)
    {Toast.makeText(finalView.getContext(),
                "Error loading youtube video",Toast.LENGTH_SHORT).show();}
});
FragmentTransaction transaction = getChildFragmentManager().beginTransaction();
transaction.replace(R.id.frame2, childFragment).commit();
```

Με την InitiateListeners() δημιουργούμε ένα κουμπί που μπορεί να χρησιμοποιήσει ο χρήστης σε περίπτωση που ο player αποτύχει να φορτώσει το βίντεο. Αυτό το κουμπί μας παραπέμπει στην εφαρμογή του YouTube και εμφανίζει τα αποτελέσματα αναζήτησης για την ταινία που επιλέξαμε.

```
btnYoutube.setOnClickListener((view) -> {
        Intent intent = new Intent(Intent.ACTION_VIEW);
        intent.setData(Uri.parse("https://www.youtube.com/results?search_query="+title));
        startActivity(intent);
});
```

Για να σιγουρευτούμε πως ο player του YouTube παραμένει μέσα στα πλαίσια της οθόνης χρησιμοποιήσαμε τον ακόλουθο κώδικα.

```
int height;
int width;
if (Build.VERSION.SDK_INT >= 13){
    display.getSize(size);
    height=size.y;
    width= size.x;
}
else
{
    height=display.getHeight();
    width=display.getWidth();
}
if(width>height)
{
    height=height +300;
}
trailerLayout.setLayoutParams(new LinearLayout.LayoutParams
        (ViewGroup.LayoutParams.MATCH_PARENT, height / 2));
```

4.2.6) Time Picker

Η κλάση αυτή χρησιμοποιείται προκειμένου να επιλέξει ο χρήστης ώρα προβολή αφού έχει ήδη επιλέξει ημερομηνία από την προηγούμενη καρτέλα.

```
public class TimePicker extends Fragment {
    private OnFragmentInteractionListener mListener;
    public static TimePicker newInstance(  Bundle args ) {...}
    public TimePicker() {...}

    @Override
    public void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {...}

    public void onButtonPressed(Uri uri) {...}

    @Override
    public void onAttach(Activity activity) {...}

    @Override
    public void onDetach() {...}
    public interface OnFragmentInteractionListener {...}
}
```

Η μέθοδος onCreateView() δημιουργεί το περιεχόμενο της καρτέλας. Εμφανίζει στον χρήστη τον κινηματογράφο και την ημερομηνία που επέλεξε καθώς και μια λίστα με τις διαθέσιμες ώρες προβολής και τις αντίστοιχες αίθουσες. Στην συγκεκριμένη κλάση δεν υπάρχει αλληλεπίδραση με τον SQL server, όλες οι απαραίτητες πληροφορίες υπάρχουν ήδη από την προηγούμενη κλάση.

4.2.7) Seat Chart

Η κλάση αυτή είναι υπεύθυνη για την εμφάνιση του διαγράμματος με τις θέσεις της αίθουσας στον χρήστη. Εμφανίζει τις διαθέσιμες θέσεις με λευκό χρώμα ενώ οι ήδη κρατημένες εμφανίζονται με κίτρινο. Ο χρήστης μπορεί να επιλέξει έναν αριθμό θέσεων και να πραγματοποιήσει μια κράτηση.

Υποκλάση της SeatChart είναι η DownloadSeatChart. Περιέχει τις μεθόδους onPreExecute(), doInBackground() και onPostExecute() . Όπως και σε προηγούμενες κλάσεις που μελετήσαμε, οι μέθοδοι onPre-PostExecute() είναι υπεύθυνες για τον χειρισμό των μηνυμάτων που εμφανίζονται στον χρήστη όταν κατεβάζουμε το διάγραμμα, ενώ η doInBackground() χρησιμοποιείται για να κατεβάσουμε τις πληροφορίες που είναι απαραίτητες για την δημιουργία του(αριθμό θέσεων, αριθμό διαδρόμων, σειρές) .

Η μέθοδος createSeatButtons() χρησιμοποιείται προκειμένου να δημιουργήσουμε το διάγραμμα των θέσεων με τις πληροφορίες που πήραμε παραπάνω. Το διάγραμμα δημιουργείται δυναμικά, καθώς η δομή του είναι

αποθηκευμένη στον server. Επίσης η μέθοδος είναι υπεύθυνη για την λειτουργικότητα των θέσεων. Κάθε φορά που ο χρήστης επιλέγει μια θέση που είναι ελεύθερη αυτή αλλάζει χρώμα από λευκό σε πράσινο, ενώ ενημερώνεται ο μετρητής θέσεων με το σύνολο των θέσεων.

```
private static  void createSeatButtons(final Context context) {
    int i=0;
    int div=Integer.parseInt(axisY)/(Integer.parseInt(aisles)+1);
    final ArrayList<String> arrayList = new ArrayList<>(Arrays.asList(redSeats));
    for (int rows = 0; rows < Integer.parseInt(axisX); rows++) {
        TableRow tr = new TableRow(context);
        for (int cols= 0; cols < Integer.parseInt(axisY); cols++) {
            i++;
            final ImageButton chairBtn = new ImageButton(context);
            TableRow.LayoutParams rowParams=new TableRow.LayoutParams
                    (ViewGroup.LayoutParams.WRAP_CONTENT,
                            ViewGroup.LayoutParams.WRAP_CONTENT);
            chairBtn.setLayoutParams(rowParams);
            final int ia = i;
            if (arrayList.contains(Integer.toString(i))) {
                chairBtn.setBackgroundResource(R.drawable.r_chair);
                chairBtn.setEnabled(false);
            } else { chairBtn.setBackgroundResource(R.drawable.w_chair);}
            chairBtn.setOnClickListener((v) -> {
                    if (chairBtn.getBackground().getConstantState().
                            equals(context.getResources().getDrawable(R.drawable.w_chair).
                                    getConstantState())) {
                        chairBtn.setBackgroundResource(R.drawable.g_chair);
                        int oldCount=Integer.parseInt(seatsCounter.getText().toString());
                        oldCount++;
                        selectedSeats.add(ia);
                        seatsCounter.setText(Integer.toString(oldCount));
                    } else {
                        chairBtn.setBackgroundResource(R.drawable.w_chair);
                        int oldCount=Integer.parseInt(seatsCounter.getText().toString());
                        oldCount--;
                        selectedSeats.remove((Integer)ia);
                        seatsCounter.setText(Integer.toString(oldCount));
                    }
            });
            if(i%div==0 && (cols!=Integer.parseInt(axisY)-1))
            {
                TableRow.LayoutParams rowParamsWithMargin=new TableRow.LayoutParams
                        (ViewGroup.LayoutParams.WRAP_CONTENT,
                                ViewGroup.LayoutParams.WRAP_CONTENT);
                rowParamsWithMargin.setMargins(0,0,30,0);
                chairBtn.setLayoutParams(rowParamsWithMargin);
            }
            tr.addView(chairBtn);
        }
        seatPlanTable.addView(tr, new TableLayout.LayoutParams
                (TableLayout.LayoutParams.WRAP_CONTENT,
                        TableLayout.LayoutParams.WRAP_CONTENT));
    }
```

Τέλος δίνουμε λειτουργικότητα στο κουμπί Proceed μέσα στην μέθοδο onCreateView() της κύριας κλάσης. Το κουμπί παίρνει τον αριθμό των θέσεων καθώς και την ταυτότητα τους, τις τοποθετεί σε μια δέσμη δεδομένων που στη συνέχεια στέλνει στο fragment Checkout το οποίο δημιουργεί. Σε περίπτωση που ο χρήστης δεν έχει επιλέξει θέσεις εμφανίζεται κατάλληλο μήνυμα.

4.2.8) Checkout

Η κλάση checkout είναι υπεύθυνη για την ολοκλήρωση της κράτησης. Δίνει στον χρήστη την δυνατότητα επιλογής μεταξύ πληρωμής με πιστωτική κάρτα ή με μετρητά και μετά το πέρας της κράτησης ενημερώνει την βάση δεδομένων με τις λεπτομέρειες. Για την αποστολή των λεπτομερειών της κράτησης στην βάση χρησιμοποιείται η υποκλάση AtemptBooking. Η υποκλάση περιέχει την μέθοδο onPreExecute() που μας ενημερώνει με κατάλληλο μήνυμα για την δημιουργία της κράτησης και την doInBackground() η οποία παίρνει της πληροφορίες της κράτησης (τον κωδικό της ταινίας που αντιστοιχεί, το πλήθος των θέσεων, τους κωδικούς της κάθε θέσης και το όνομα του χρήστη που κάνει την κράτηση) και στέλνει στην βάση ένα αίτημα εισαγωγής νέας κράτησης. Ο χρήστης ενημερώνεται με μήνυμα για την επιτυχία δημιουργίας κράτησης. Τέλος περιέχει την μέθοδο onPostExecute() που αναλαμβάνει μετά το πέρας της δημιουργίας μιας κράτησης να μας πάει στην καρτέλα του ιστορικού κρατήσεων η οποία ενημερώνεται με την νέα κράτηση.

```
public class AttemptBooking extends AsyncTask<String, String, String> {

    @Override
    protected void onPreExecute() [...]

    @Override
    protected String doInBackground(String... strings) {
        String msg=null;
        paymentId=-1;
        try {
            List<NameValuePair> params=new ArrayList<~>();
            params.add(new BasicNameValuePair("showId", showId));
            params.add(new BasicNameValuePair("size",
                    Integer.toString(selectedSeats.size())));
            params.add(new BasicNameValuePair("username",username));
            for(int i=0; i<selectedSeats.size(); i++) {
                params.add(new BasicNameValuePair("seatNo"+Integer.toString(i),
                        Integer.toString(selectedSeats.get(i))));
            }
            JSONObject jsonObject=jsonParser.makeHttpRequest( LOGIN_URL, "POST", params);
            msg = jsonObject.getString(TAG_MESSAGE);
            paymentId=jsonObject.getInt("success");
        }
        catch (JSONException e){
            e.printStackTrace();
        }
        return msg;
    }

    protected void onPostExecute(String msg) {...}
}
```

Επιστρέφοντας στην κύρια κλάση checkout έχουμε τις μεθόδους onCreateView() και initiateViews(). Η πρώτη είναι υπεύθυνη για την δημιουργία της καρτέλας. Δημιουργεί μια λίστα με όλες τις λεπτομέρειες της κράτησης του χρήστη, έτσι ώστε ο χρήστης να επιβεβαιώσει την ορθότητα της κράτησης του. Η μέθοδος initiateViews() δίνει λειτουργικότητα στα κουμπιά της καρτέλας. Σε περίπτωση που ο χρήστης επιλέξει να πληρώσει με πιστωτική και πατήσει το αντίστοιχο κουμπί (btnCreditCard) του εμφανίζεται ένα layout στο οποίο μπορεί είτε χειροκίνητα ή αυτόματα (btnAutoFill) να συμπληρώσει τα στοιχεία της κάρτας του. Σε περίπτωση που επιλέξει μετρητά εμφανίζεται το ανάλογο layout. Τέλος υπάρχει το κουμπί αποστολής της κράτησης. Το κουμπί ανάλογα με τον τρόπο πληρωμής που επέλεξε ο χρήστης στέλνει στην βάση με χρήστη κατάλληλου php script αίτημα

δημιουργίας νέας εγγραφής στον πίνακα των κρατήσεων.

```
        btnAutoFill.setOnClickListener((v) -> {
                creditNo.setText(PreferenceManager.getDefaultSharedPreferences
                        (getActivity()).getString("creditNo", ""));
                cv2.setText(PreferenceManager.getDefaultSharedPreferences
                        (getActivity()).getString("cv2", ""));
                expDate.setText(PreferenceManager.getDefaultSharedPreferences
                        (getActivity()).getString("expDate", ""));
                cardName.setText(PreferenceManager.getDefaultSharedPreferences
                        (getActivity()).getString("cardName", ""));
        });
        btnCash.setOnClickListener((v) -> {
                creditLayout.setVisibility(view.GONE);
                cashLayout.setVisibility( View.VISIBLE);
        });
        btnCreditCard.setOnClickListener((v) -> {
                cashLayout.setVisibility(view.GONE);
                creditLayout.setVisibility( View.VISIBLE);
        });
        submitBtn=(Button)view.findViewById(R.id.btnSubmit);
        submitBtn.setOnClickListener((v) -> {
                if (cashLayout.getVisibility() == View.VISIBLE) {
                    LOGIN_URL = "http://nikkouk.zapto.org:1024/webservice/NewCashBooking.php";
                    new AttemptBooking().execute();
                }
                else if (creditLayout.getVisibility() == View.VISIBLE) {
                    LOGIN_URL = "http://nikkouk.zapto.org:1024/webservice/NewCreditBooking.php";
                    new AttemptBooking().execute();
                }
                else
                {
                    Toast.makeText(getActivity(), "Choose a payment method first!",
                            Toast.LENGTH_LONG).show();
                }

        });
```

4.3) Βοηθητικές κλάσεις

Εκτός από τις κλάσεις που παρέχουν περιεχόμενο στον χρήστη υπάρχουν ορισμένες των οποίων αν και η λειτουργικότητα δεν είναι εμφανής, παίζουν κρίσιμο ρόλο στην λειτουργία του προγράμματος. Οι κλάσεις αυτές είναι οι JSONParser και SlidePagerAdapter.

4.3.1) JSONParser

Η κλάση αυτή χρησιμοποιείται προκειμένου να ελέγξουμε την μέθοδο αποστολής των πληροφοριών από την βάση μας. Ανάλογα κωδικοποιεί τα δεδομένα με την κατάλληλη μορφή (Get ή Post). Επίσης κατά την λήψη δεδομένων είναι υπεύθυνη για τον τρόπο με τον οποίο κωδικοποιούνται τα αποτελέσματα που παίρνουμε τρέχοντας ερωτήματα στην βάση δεδομένων.

4.3.2) SlidePagerAdapter

Η κλάση SlidePagerAdapter χρησιμοποιείται προκειμένου να ενσωματώσουμε στην εφαρμογή μας την λειτουργικότητα ενός ViewPager. Με αυτό τον τρόπο ο χρήστης μπορεί να μετακινηθεί από καρτέλα σε καρτέλα μέσω κάνοντας swipe προς την κατεύθυνση της καρτέλας που επιθυμεί να εμφανίσει.

4.4) XML

Η XML είναι μια γλώσσα σήμανσης, που περιέχει ένα σύνολο κανόνων για την ηλεκτρονική κωδικοποίηση κειμένων. Χρησιμοποιείται δηλαδή προκειμένου να περιγράψει τα δεδομένα. Στην περίπτωση μας η XML είναι υπεύθυνη για το πώς θα φαίνονται τα πάντα στην εφαρμογή μας, από το μέγεθος μιας λίστας μέχρι την εμφάνιση ενός κουμπιού και τον χώρο που καταλαμβάνει ένα αντικείμενο στην οθόνη μας. Επίσης μπορούμε να ορίσουμε διαφορετικό τρόπο απεικόνισης για την ίδια διεπαφή ανάλογα με την συσκευή στην οποία αυτή εμφανίζεται.

Το σημαντικότερο xml αρχείο της εφαρμογής είναι το AndroidManifest. Το αρχείο αυτό είναι απαραίτητο για κάθε εφαρμογή και έχει πάντα τη συγκεκριμένη ονομασία. Μπορεί κανείς να το σκεφτεί σαν ην ταυτότητα της εφαρμογής. Εδώ καθορίζονται τα διάφορα χαρακτηριστικά των οθονών της εφαρμογής καθώς και τα δικαιώματα χρήσης που η εφαρμογή ζητάει για να λειτουργήσει σωστά, όπως την πρόσβαση στο Internet την πρόσβαση στην εσωτερική μνήμη κ.α. . Επίσης στο Android Manifest ορίζουμε όλες της δραστηριότητες(Activities) που περιέχει το πρόγραμμα. Επίσης πρέπει να ορίσουμε ποια είναι η κύρια δραστηριότητα μας. Αυτό το ορίζουμε χρησιμοποιώντας ένα intent-filter που εμπεριέχει τα main action και launcher category. Όλες οι υπόλοιπες δραστηριότητες δηλώνονται ως παιδία της κύριας δραστηριότητας. Ορίζουμε ακόμη τα κλειδιά των facebook και Google Maps API τα οποία είναι αναγκαία για να τα ενσωματώσουμε στην εφαρμογή μας.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.nick.tabsstrip" >
    <uses-permission android:name="com.vogella.android.locationapi.maps.permission.MAPS_RECEIVE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <permission
        android:name="com.vogella.android.locationapi.maps.permission.MAPS_RECEIVE"
        android:protectionLevel="signature" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="MyCinema"
        android:theme="@style/Theme.AppCompat" >
        <activity
            android:name=".MainActivity"
            android:configChanges="keyboardHidden|orientation|screenSize"
            android:label="MyCinema" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".ShowtimeActivity"
            android:configChanges="keyboardHidden|orientation|screenSize"
            android:label="@string/app_name"
            android:parentActivityName=".MainActivity" >
        </activity>
        <activity
            android:name="com.facebook.LoginActivity"
            android:configChanges="keyboardHidden|orientation|screenSize"
            android:parentActivityName=".MainActivity" >
        </activity>
        <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="AIzaSyBa9ebwmY3Jm2SuxzMij_ZiMsWUVNH_KrI" />
        <meta-data
            android:name="com.facebook.sdk.ApplicationId"
            android:value="@string/facebook_app_id" />
        <activity
            android:name=".HelpActivity"
            android:label="Help" >
        </activity>
    </application>
</manifest>
```

4.5) PHP

Η PHP χρησιμοποιείται ως ο ενδιάμεσος μεταξύ του Android Studio και του MySQL server. Δημιουργεί μια σύνδεση μεταξύ της βάσης και του προγράμματος μέσω της οποίας εκτελεί SQL queries. Έπειτα τα αποτελέσματα των queries κωδικοποιούνται σε JSON αντικείμενα. Τα script που χρησιμοποιήσαμε έχουν την παρακάτω μορφή:

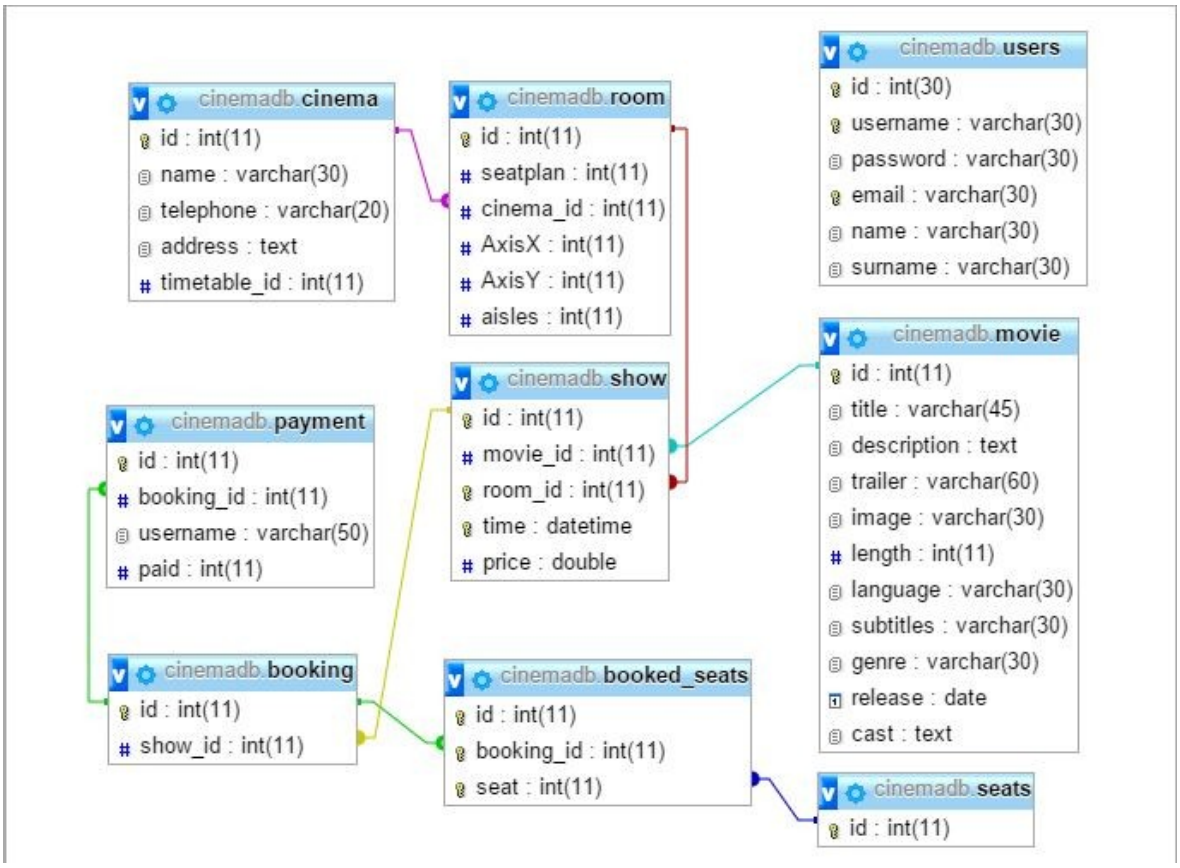
```php
<?php header("content-type: text/html;charset=utf-8") ;

$user="ptuxiakh";
$pass="ptuxiakh";
$host ="192.168.2.2";
$dbname="cinemadb";
    #open connection
    try{
        $DBH = new PDO("mysql:host=$host;dbname=$dbname;charset=utf8", $user, $pass);
        $DBH->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION );
    }
    catch (PDOException $e){
        $response["success"] = 0;
        $response["message"] = "Database Error1. Please Try Again!";
        die(json_encode($response));
    }
    try{
        $STH = $DBH->prepare("SELECT m.*,s.price from `show` s,`movie` m
        where  m.id=s.movie_id and time>=now() GROUP BY title,price ");
        $STH->execute();
        $STH->setFetchMode(PDO::FETCH_ASSOC);
    }
    catch (PDOException $e){
                $response["success"] = 0;
                $response["message"] = $e->getMessage();
                die(json_encode($response));
    }
            while ($row = $STH->fetch()){
            $data[] = $row;
        }
        $response["success"] = 1;
        $response["array"] = $data;
           $response["message"] = "successful!";
           die(json_encode($response));
    # close the connection
    $DBH = null;
?>
```

Χρησιμοποιούμε το επίπεδο πρόσβασης βάσεων δεδομένων PDO που παρέχει μια ενιαία μέθοδο πρόσβασης σε πολλαπλές βάσεις δεδομένων προκειμένου να στείλουμε τα στοιχεία μας στην βάση (την τοποθεσία της βάσης, το όνομα της, το όνομα του χρήστη που προσπαθεί να συνδεθεί και τον κωδικό του). Έπειτα δημιουργούμε και στέλνουμε ένα ερώτημα στην βάση. Τέλος λαμβάνουμε τα αποτελέσματα του ερωτήματος και τα κωδικοποιούμε με την μορφή ενός JSON αντικειμένου. Ένα php script μπορεί να τρέξει και περισσότερα από ένα SQL statements, ενημερώνοντας πολλούς πίνακες της βάσης ταυτόχρονα..

4.6) Βάση δεδομένων SQL

Στην βάση δεδομένων αποθηκεύονται όλες οι πληροφορίες του κινηματογράφου αλλά και των χρηστών. Ο διαχειριστής είναι υπεύθυνος για την ανανέωση της βάσης με νέες ταινίες και τις ώρες προβολών τους, ενώ στην βάση δημιουργούνται αυτόματα νέες κρατήσεις από τους χρήστες. Παράλληλα εδώ αποθηκεύονται και οι λογαριασμοί των χρηστών καθώς και οι κρατήσεις που αντιστοιχούν στον καθένα τους. Παρακάτω παραθέτουμε μια συνολική εικόνα της βάσης, των πινάκων και των πεδίων που αντιστοιχούν σε καθένα καθώς και τις σχέσεις που διέπουν τους πίνακες.



Εισαγωγή, ενημέρωση ή διαγραφή δεδομένων από έναν πίνακα της βάσης έχουν ως αποτέλεσμα αντίστοιχες ενέργειες και σε άλλους πίνακες της βάσης.

5)Μελλοντικές επεκτάσεις

Τελειώνοντας την ανάπτυξη της εφαρμογής μας προσπαθήσαμε να σκεφτούμε με ποιον τρόπο θα ήταν δυνατό στο μέλλον να την εμπλουτίσουμε, να προσθέσουμε περιεχόμενο επιπλέον από αυτό που ήδη είναι διαθέσιμο στους χρήστες. Ως συνέπεια δημιουργήσαμε μια λίστα από λειτουργίες που θα μπορούσαμε να ενσωματώσουμε σε επόμενες εκδόσεις της εφαρμογής, οι οποίες θα εξυπηρετούν τόσο τους πελάτες της εφαρμογής, όσο και την εταιρία κινηματογράφου που διαχειρίζεται το πρόγραμμα. Αρχικά σκεφτήκαμε να εισάγουμε σε μια νέα λίστα τις ταινίες που θα παίζει προσεχώς ο κινηματογράφος. Έπειτα σκεφτήκαμε να δώσουμε στον χρήστη την δυνατότητα να επιλέγει κινηματογράφο προτού επιλέξει την ταινία που θέλει να παρακολουθήσει. Το σκεπτικό πίσω από αυτή την αλλαγή είναι να γνωρίζει ο χρήστης εκ των προτέρων εάν η ταινία που θέλει να παρακολουθήσει είναι διαθέσιμη στον κινηματογράφο της επιλογής του, γλιτώνοντας έτσι χρόνο σε περίπτωση που δεν είναι και ο χρήστης θα πρέπει να επιλέξει εκ νέου μια διαφορετική ταινία. Μια ακόμη προσθήκη που θα μπορούσαμε να κάνουμε είναι να ενσωματώσουμε επιπλέον επιλογές στον τομέα των κοινωνικών δικτύων. Ήδη υπάρχει η επιλογή να μοιραστεί ο χρήστης την κράτηση του στην προσωπική του σελίδα στο Facebook, θα μπορούσαμε όμως να έχουμε επιπλέον επιλογές όπως το Google+ ή το Twitter. Νέα χαρακτηριστικά θα μπορούσαν να κάνουν και ευκολότερη την αλληλεπίδραση των διαχειριστών με το πρόγραμμα. Αυτή τη στιγμή η διαδικασία εισαγωγής - επεξεργασίας – διαγραφής ταινιών, προβολών, λογαριασμών, γενικότερα όλη η αλληλεπίδραση μεταξύ διαχειριστή και server γίνεται μέσω του phpMyAdmin σε κάποιον περιηγητή. Θα μπορούσαμε στο μέλλον να αναπτύξουμε μια ξεχωριστή εφαρμογή η οποία θα έκανε την διαδικασία αυτή πιο απλή για τους διαχειριστές του συστήματος. Επιπλέον η εφαρμογή θα μπορούσε να τροποποιηθεί προκειμένου η εταιρία κινηματογράφου που την χειρίζεται να μπορεί να εισάγει διαφημίσεις για άλλες ταινίες ή για τον κινηματογράφο. Επιπλέον θα μπορούσε να προσφέρει στους χρήστες προνόμια χρησιμοποιώντας την εφαρμογή, όπως έκπτωση σε συγκεκριμένες ταινίες ή ένα σύστημα ανταμοιβών για τις ταινίες που έχει παρακολουθήσει ο χρήστης, π.χ. παρακολουθώντας δέκα ταινίες να δικαιούται ένα δωρεάν εισιτήριο. Προφανώς απαιτείται περαιτέρω μελέτη και σκέψη προκειμένου τα νέα χαρακτηριστικά που μπορούν να ενσωματωθούν στην εφαρμογή να εναρμονίζονται με τα ήδη υπάρχοντα και να μην δημιουργούν πρόβλημα στους χρήστες. Παραδείγματος χάριν θα εισάγοντας μια λίστα με τις ταινίες που θα παίζουν προσεχώς θα πρέπει να καθορίσουμε το αν θα πρέπει να αλλάξουμε την υπάρχουσα διάταξη της εφαρμογής εισάγοντας μια νέα καρτέλα η ενσωματώνοντας τη λίστα στην ήδη υπάρχουσα αρχική. Είναι όμως σαφές πως υπάρχουν περιθώρια ανάπτυξης για την εφαρμογή.

ΒΙΒΛΙΟΓΡΑΦΙΑ

- **Rebecca Scudder.** An Overview of Structured Query Language: What is SQL?
- **Michael J. Miller Jr.** Overview of the MySQL Database Management System
- **Marc Delisle**. Mastering phpMyAdmin for Effective MySQL Management
- **Jonathan Simon**. Head First Android Development.
- **Neil Smyth**. Android Studio Essentials.
- **Reto Meier**. Professional Android 4 Application Development
- **Αλέξανδρος Χατζηγεωργίου**. Αντικειμενοστραφής Σχεδίαση UML, Αρχές, Πρότυπα και Ευρετικοί κανόνες.
- **Paul Deitel, Harvey Deitel, Abbey Deitel, Michael Morgano**. Android for Programmers, an App-Driven Approach. 1st Edition, Prentice Hall, USA.
- **Zigurd Mednieks, Laird Dornin, G. Blake Meike, MasumiNakamura** Programming Android: Java Programming for the New Generation of Mobile Devices
- **Jason Morris**. Android User Interface Development: Beginner's Guide.

Διαδικτυακές πηγές:

- www.Android-developers.blogspot.gr
- Techterms.com
- www.Stackoverflow.com
- www.developer.android.com/training

ΕΠΙΛΟΓΟΣ

Η παρούσα πτυχιακή είχε ως αντικείμενο την ανάπτυξη εφαρμογής για την κράτηση εισιτηρίων σε κινηματογράφο για την πλατφόρμα Android καθώς και η εις βάθος κατανόηση τόσο του συστήματος στο οποίο απευθυνόμαστε, όσο και των εργαλείων ανάπτυξης που χρησιμοποιήσαμε κατά την διάρκεια της ανάπτυξης.

Στο σημείο αυτό θα θέλαμε να ευχαριστήσουμε τον καθηγητή μας Δρ. Δ.Ν Κλεφτούρη που μας εμπιστεύθηκε με την ανάθεση της εργασίας καθώς και για την μεγάλη υπομονή του καθώς ξεπεράσαμε αρκετά τον αρχικό προγραμματισμό μας αναφορικά με το χρονοδιάγραμμα της εργασίας. Ακόμη θα θέλαμε να ευχαριστήσουμε τους καθηγητές του τμήματος πληροφορικής του ΤΕΙ Θεσσαλονίκης που προσπάθησαν για πολλά χρόνια να μας διδάξουν και να μας καθοδηγήσουν έτσι ώστε να γίνουμε σωστοί προγραμματιστές. Τέλος θα θέλαμε να πούμε ένα μεγάλο ευχαριστώ στις οικογένειες μας που μας ανέχθηκαν όλο αυτό το διάστημα που τους κάναμε δύσκολη τη ζωή, καθώς και στους φίλους μας που μας συμπαραστάθηκαν με τον έναν ή τον άλλο τρόπο.