

**ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΑ**

**ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ**

**ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ**

**ΜΕΤΡΗΣΕΙΣ ΛΟΓΙΣΜΙΚΟΥ ΣΕ ANDROID ΜΕ ΤΗΝ ΧΡΗΣΗ ΤΩΝ DESIGN**

**PATTERNS**

***ΟΝΟΜΑΤΕΠΩΝΥΜΟ ΦΟΙΤΗΤΗ: ΓΕΩΡΓΙΟΣ ΧΡΙΣΤΟΠΟΥΛΟΣ***

***ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ :Π14198***

***ΕΠΙΒΛΕΠΟΝΤΑΣ ΚΑΘΗΓΗΤΗΣ:ΕΥΘΥΜΗΣ ΑΛΕΠΗΣ***

**Πειραιάς,2019**

**Περίληψη**

Σκοπός της παρούσας πτυχιακής εργασίας είναι η δοκιμή και η κατασκευή μιας mobile εφαρμογής ώστε να πάρουμε μετρήσεις για τις επιδόσεις της. Η εφαρμογή αυτή υλοποιήθηκε στο προγραμματιστικό περιβάλλον Android Studio.Με την χρήση 6 design pattern καταλήξαμε σε συμπεράσματα τα οποία μας δίνουν μία καθαρή εικόνα για τους καλύτερους τρόπους δόμησης ενός προγράμματος. Αυτοί οι τρόποι ασχολούνται με την αρχικοποίηση, αλλαγή συμπεριφοράς αλλά και με την δομή των αντικειμένων. Επίσης έγινε αναφορά και σε design pattern τα οποία δίνουν λύσεις στον τρόπο κατασκευής του προγράμματος. Τέλος, αυτές οι λύσεις και τα συμπεράσματα μπορούν να αποτελέσουν σημαντικό εργαλείο για προγραμματιστές που ασχολούνται με το Android Development αλλά και γενικότερα στον αντικειμενοστραφή προγραμματισμό.

**Περιεχόμενα**

[1.Εισαγωγή 6](#_Toc62562)

[1.1 Κίνητρο για την διεξαγωγή της εργασίας 6](#_Toc62563)

[1.2 Σκοπός και στόχοι εργασίας 6](#_Toc62564)

[1.3 Δομή εργασίας 6](#_Toc62565)

[2.Ανασκόπηση πεδίου 8](#_Toc62566)

[2.1 Splitwise 8](#_Toc62567)

[2.2 Spending Tracker 12](#_Toc62568)

[3.Παρουσίαση της εφαρμογής 16](#_Toc62569)

[3.1 Χρηση της εφαρμογής 16](#_Toc62570)

[3.2 Η Δραστηριότητα Login 16](#_Toc62571)

[3.3 Η Δραστηριότητα Register 19](#_Toc62572)

[3.4 Η Δραστηριότητα Reset 20](#_Toc62573)

[3.5 Η Δραστηριότητα Home 20](#_Toc62574)

[4. ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ 32](#_Toc62575)

[4.1. ΜΕΤΡΗΣΕΙΣ ΛΟΓΙΣΜΙΚΟΥ ΣΤΟ DEPENDENCY INJECTION PATTERN 32](#_Toc62576)

[4.1.1.Εισαγωγή 32](#_Toc62577)

[4.1.2.Πως λειτουργεί? 32](#_Toc62578)

[4.1.3.Διάγραμμα κλάσης 33](#_Toc62579)

[4.1.4.Οφέλη από το DI 33](#_Toc62580)

[4.1.5.Annotation Processor 34](#_Toc62581)

[4.1.6.Profiling της εφαρμογής με την χρήση του Android Profiler 34](#_Toc62582)

[4.2. ΜΕΤΡΗΣΕΙΣ ΛΟΓΙΣΜΙΚΟΥ ΣΤΟ ΜVP PATTERN 39](#_Toc62583)

[4.2.1.Εισαγωγή 39](#_Toc62584)

[4.2.2.Πως λειτουργεί? 39](#_Toc62585)

[4.2.3.Διάγραμμα κλάσης 42](#_Toc62586)

[4.2.4.Οφέλη από το MVP 43](#_Toc62587)

[4.2.5.Profiling της εφαρμογής με την χρήση του Android Profiler 44](#_Toc62588)

[4.3 ΜΕΤΡΗΣΕΙΣ ΛΟΓΙΣΜΙΚΟΥ ΣΤΟ PROXY PATTERN 48](#_Toc62589)

[4.3.1.Εισαγωγή 48](#_Toc62590)

[4.3.2.Πως λειτουργεί? 48](#_Toc62591)

[4.3.3.Διάγραμμα κλάσης 49](#_Toc62592)

[4.3.4.Οφέλη του Proxy Pattern 50](#_Toc62593)

[4.3.5.Profiling της εφαρμογής με την χρήση του Android Profiler 50](#_Toc62594)

[4.4 ΜΕΤΡΗΣΕΙΣ ΛΟΓΙΣΜΙΚΟΥ ΣΤΟ TEMPLATE PATTERN 53](#_Toc62595)

[4.4.1.Εισαγωγή 53](#_Toc62596)

[4.4.2.Πως λειτουργεί? 53](#_Toc62597)

[4.4.3.Διάγραμμα κλάσης 54](#_Toc62598)

[4.4.4.Οφέλη του Template Pattern 55](#_Toc62599)

[4.4.5.Profiling της εφαρμογής με την χρήση του Android Profiler 56](#_Toc62600)

[4.5 ΜΕΤΡΗΣΕΙΣ ΛΟΓΙΣΜΙΚΟΥ ΣΤΟ BUILDER PATTERN 60](#_Toc62601)

[4.5.1.Εισαγωγή 60](#_Toc62602)

[4.5.2.Πώς λειτουργεί? 60](#_Toc62603)

[4.5.3.Διάγραμμα κλάσης 62](#_Toc62604)

[4.5.4.Οφέλη του Builder Pattern 62](#_Toc62605)

[4.5.5.Profiling της εφαρμογής με την χρήση του Android Profiler 63](#_Toc62606)

[4.6 ΜΕΤΡΗΣΕΙΣ ΛΟΓΙΣΜΙΚΟΥ ΣΤΟ MVC PATTERN 67](#_Toc62607)

[4.6.1 Εισαγωγή 67](#_Toc62608)

[4.6.2.Πως λειτουργεί? 67](#_Toc62609)

[4.6.3.Διάγραμμα κλάσης 68](#_Toc62610)

[4.6.4.Οφέλη του MVC Pattern 69](#_Toc62611)

[4.6.5.Android Profiler on Expense Manager 70](#_Toc62612)

[4.7 Ενέργειες χρήστη μέσα στην εφαρμογή 74](#_Toc62613)

[4.8 Στατιστικα στοιχεια των Design Pattern και συγκριση 75](#_Toc62614)

[4.8.1.Στατιστικά στοιχεία 76](#_Toc62615)

[4.8.2. Σύγκριση των Design Pattern 78](#_Toc62616)

[5.Εργαλεια και Τεχνολογίες 82](#_Toc62617)

[5.1 Γλώσσα Προγραμματισμού 82](#_Toc62618)

[5.2 Βάση Δεδομένων 83](#_Toc62619)

[5.3 Περιβάλλον 83](#_Toc62620)

[5.4 Εφαρμογή για Μετρήσεις 84](#_Toc62621)

[5.5 Plugins 84](#_Toc62622)

[6.Συμπεράσματα και Μελλοντικές επεκτάσεις 85](#_Toc62623)

[7. Βιβλιογραφία 86](#_Toc62624)

[8.Παράρτημα 87](#_Toc62625)

[8.1 Γλωσσάριο 88](#_Toc62626)

# 1.Εισαγωγή

### 1.1 Κίνητρο για την διεξαγωγή της εργασίας

Μεγάλο μέρος των κινητών τηλεφώνων χρησιμοποιεί το λειτουργικό σύστημα Android. Στην σύγχρονη εποχή οι εφαρμογές ολοένα και μεγαλώνουν σε μέγεθος έχοντας αντίκτυπο στα τεχνικά χαρακτηριστικά των τηλεφώνων. Αυτό συμβαίνει γιατί οι σύγχρονες εφαρμογές απαιτούν πολλούς υπολογισμούς αλλά και αρκετή μνήμη προκειμένου να διεκπεραιώσουν κάποιες λειτουργίες. Για αυτό το λόγο χρειάζεται να γίνει σύγκριση των διαφορετικών design pattern προκειμένου να βρούμε το πιο αποτελεσματικό και βέλτιστο που εξυπηρετεί τις σύγχρονες ανάγκες.

### 1.2 Σκοπός και στόχοι εργασίας

Σκοπός της παρούσας εργασίας είναι η μετρήσεις λογισμικού, πάνω στα διαφορετικά design patterns σε Android με την χρήση Firebase, ώστε να δούμε στο κάθε pattern τι διαχείριση πόρων επιτυγχάνεται αλλά και την επικοινωνία μεταξύ των κλάσεων και των αντικειμένων του προγράμματος .

### 1.3 Δομή εργασίας

**Κεφάλαιο 1- Εισαγωγή:**

Στο πρώτο κεφάλαιο, γίνεται μια εισαγωγή για την εφαρμογή που υλοποιήθηκε καθώς και ποιες ανάγκες εξυπηρετεί ή πρόκειται να εξυπηρετήσει στο μέλλον.

**Κεφάλαιο 2- Ανασκόπηση πεδίου:**

Στο δεύτερο κεφάλαιο, γίνεται μια παρουσίαση και αναφορά σε παρόμοιες εφαρμογές που υπάρχουν στην Ελλαδα και στο εξωτερικό.

**Κεφάλαιο 3- Παρουσίαση και χρήση της εφαρμογής:**

Στο τρίτο κεφάλαιο, το οποίο αποτελεί το εγχειρίδιο χρήσης της εφαρμογής καθώς παρουσιάζει επεξηγηματικά με την χρήση εικόνων τις λειτουργίες της εφαρμογής.

**Κεφάλαιο 4- Αρχιτεκτονική συστήματος:**

Στο τέταρτο κεφάλαιο, παρουσιάζεται η αρχιτεκτονική του συστήματος καθώς και κάποια σχεδιαγράμματα για την πλήρη κατανόηση του συστήματος. Επιπλέον γίνεται αναφορά στις τεχνολογίες που χρησιμοποιήθηκαν όπως η γλώσσα, η βάση καθώς και το πως φτιάχτηκε.

**Κεφάλαιο 5- Εργαλεία και Τεχνολογίες:**

Στο πέμπτο κεφάλαιο, γίνεται αναφορά σε όλες τις τεχνολογίες και τα εργαλεία που χρησιμοποιήσαμε όπως βάση δεδομένων, frameworks, προγραμματιστικό περιβάλλον , plugins.

**Κεφάλαιο 6- Συμπεράσματα και μελλοντικές επεκτάσεις:**

Τέλος στο έκτο κεφάλαιο έχουμε τα συμπεράσματα και τις μελλοντικές επεκτάσεις της εφαρμογής.

**Κεφάλαιο 7- Βιβλιογραφία:**

Αναφορά στις πήγες που χρησιμοποιήθηκαν για την εκπόνηση της εργασίας.

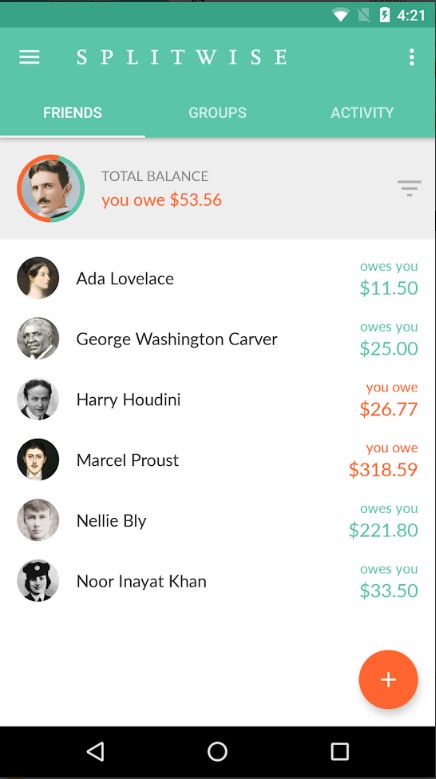
**Κεφάλαιο 8- Παραρτήματα:** Περιέχει το Γλωσσάριο.

# 2.Ανασκόπηση πεδίου

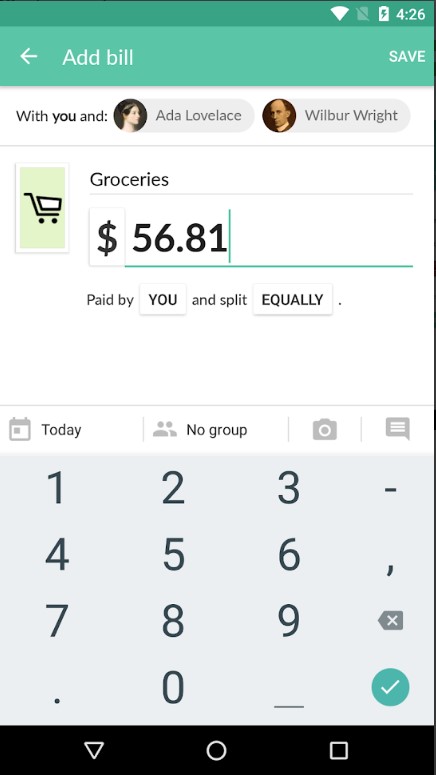
Οι εφαρμογές στις οποίες έχουμε την δυνατότητα να καταγράψουμε τα έξοδα μας, είναι ευρέως διαδεδομένες, αφού πολλές φορές οι άνθρωποι θέλουν να έχουν μια εικόνα για τα λεφτά που ξοδεύουν. Επίσης υπάρχουν και παρόμοιες εφαρμογές με τις οποίες κάποιος καταχωρεί τα χρήματα που δανείστηκε από κάποιον άλλον ή και το αντίστροφο.

### 2.1 Splitwise

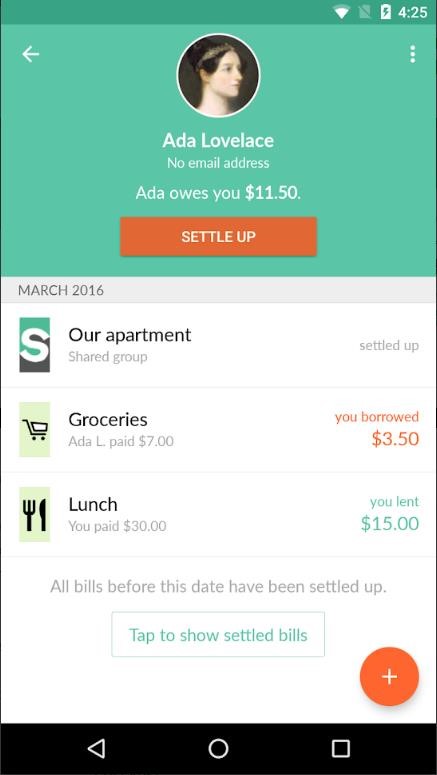
Μία πολύ γνωστή εφαρμογή που έχει αυτού του είδους την λειτουργία είναι το **Splitwise.**Το **Splitwise** υπάρχει διαθέσιμο στο  **Google Play store** και είναι από τις καλύτερες εφαρμογές για το μοίρασμα λογαριασμών μεταξύ χρηστών. Στην παρακάτω εικόνα μπορούμε να δούμε την κύρια οθόνη της εφαρμογής η οποία μας δίνει έναν πίνακα με πληροφορίες για το μοίρασμα των εξόδων.



Με τα πορτοκαλί γράμματα είναι τα λεφτά που χρωστάει ο χρήστης ενώ τα πράσινα γράμματα είναι τα λεφτά που πρέπει να εισπράξει ο χρήστης. Με το πορτοκαλί κουμπί μπορούμε να κάνουμε καταχώρηση νέου εσόδου/εξόδου όπως φαίνεται στην παρακάτω εικόνα. Όπως βλέπουμε υπάρχει και η δυνατότητα μοιράσματος των εξόδων σε παραπάνω από δύο άτομα.



Σε περίπτωση που ο χρήστης θέλει να δει τους λογαριασμούς του με κάποιον άλλον χρήστη συγκεκριμένα, τότε μπορεί να επιλέξει το εικονίδιο του και να ανοίξει η καρτέλα με τους λογαριασμούς όπως στην παρακάτω εικόνα.Επίσης δίνεται και η επιλογή settle up ώστε με ένα κουμπί να εξοφληθούν τα χρέοι μεταξύ των χρηστών.



### 2.2 Spending Tracker

Μία άλλη εφαρμογή της ίδιας κατηγορίας είναι το **Spending Tracker.**Το **Spending Tracker** είναι μια δωρεάν εφαρμογή η οποία είναι διαθέσιμη στο **Google Play store.**Η συγκεκριμένη εφαρμογή είναι προσωπικός καταχωρητής εξόδων ο οποίος δείχνει με διάφορα διαγράμματα στοιχεία για το που ξοδεύονται τα περισσότερα χρήματα.Στην παρακάτω είκονα βλέπουμε την κατηγοριοποίηση τον εξόδων.



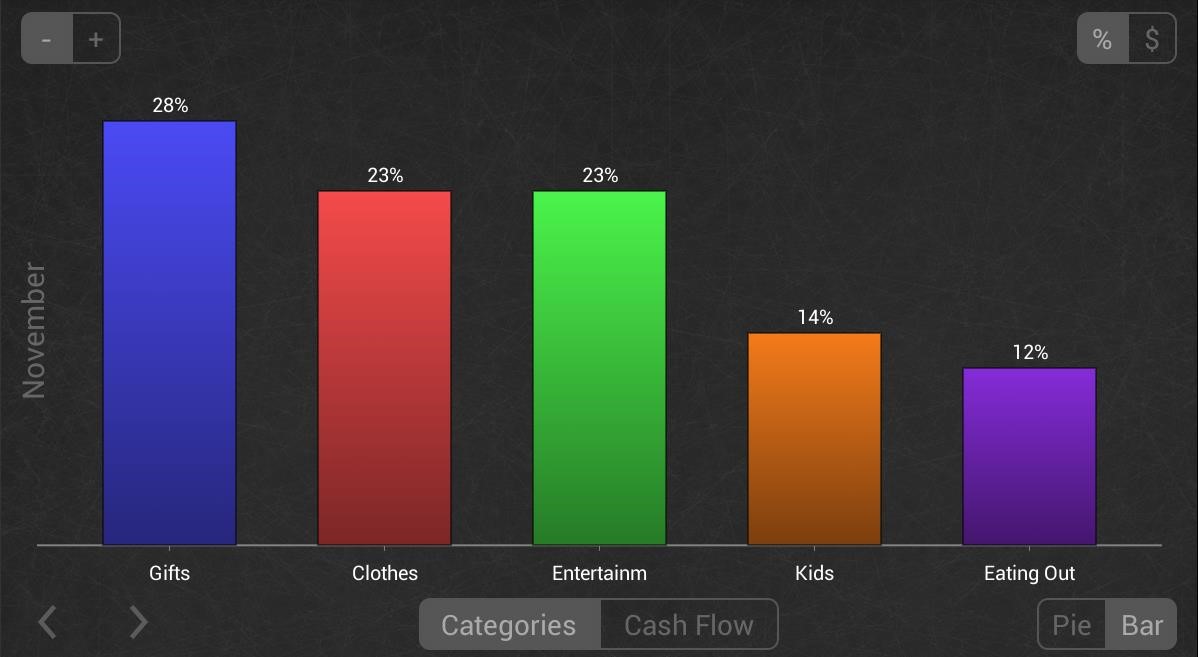
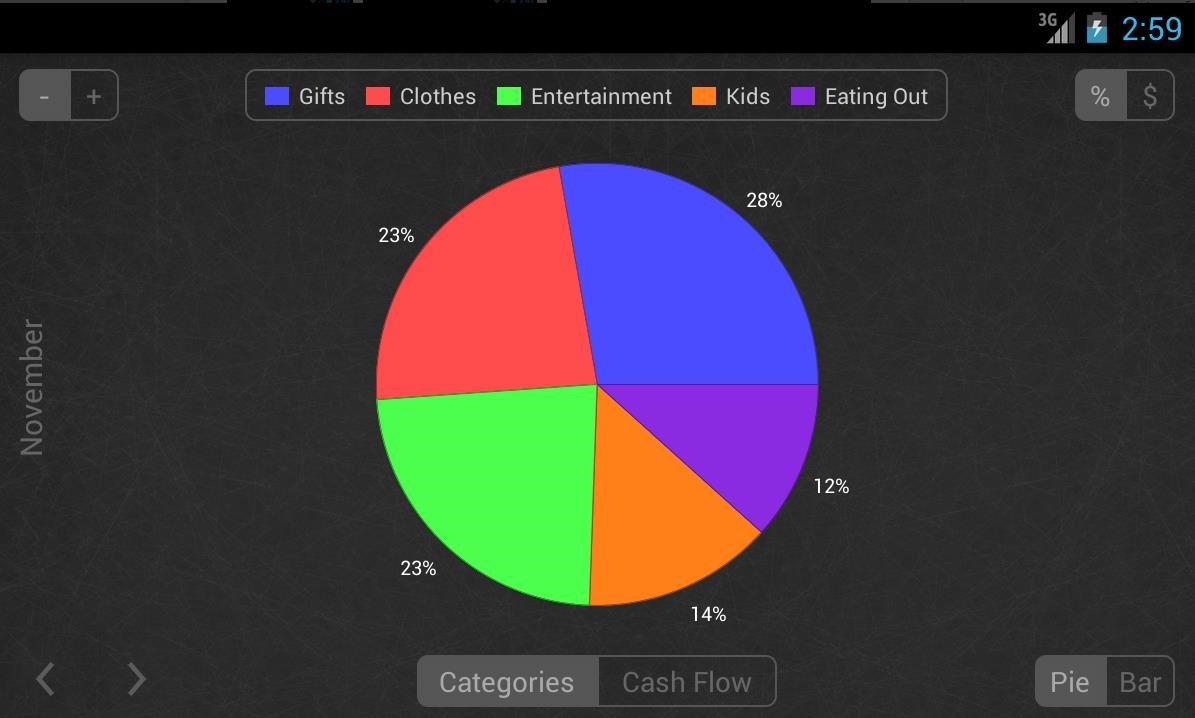
Επιπλέον η εφαρμογή παρέχει και ιστορικό των συναλλαγών ώστε ο χρήστης να έχει ξεκάθαρη εικόνα για τα έξοδα του.Στην παρακάτω εικόνα μπορούμε να δούμε την καρτέλα με το ιστορικό του χρήστη.



Ο κάθε χρήστης μπορεί να δεί τα έξοδα του στην παρακάτω καρτέλα η οποία παρέχει πληροφορίες για το κάθε έξοδο μεμονομένα όπως φαίνεται παρακάτω.



Ο χρήστης έχει την δυνατότητα να δεί και διαγράμματα ώστε να έχει μία οπτική εικόνα για τα έξοδα του.



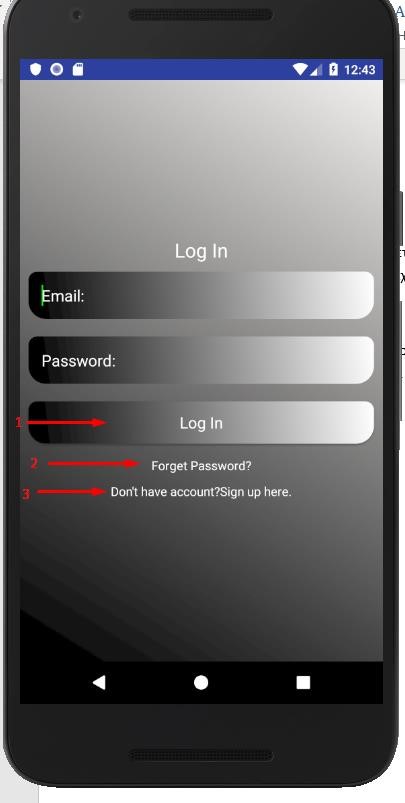
# 3.Παρουσίαση της εφαρμογής

### 3.1 Χρηση της εφαρμογής

Η χρηση της εφαρμογης προοριζεται για χρήστες οι οποίοι θέλουν να καταχωρούν τα έξοδα τους για να είναι σε θέση να διαχειρίζονται τα χρήματα τους πιο αποδοτικά.

### 3.2 Η Δραστηριότητα Login

Η πρώτη οθόνη της εφαρμογής που συναντάμε είναι η δραστηριότητα Login.Σύμφωνα με την παρακάτω εικόνα μας παρέχονται κάποιες επιλογές, οι οποίες υπογραμμίζονται με τα κόκκινα βέλη.

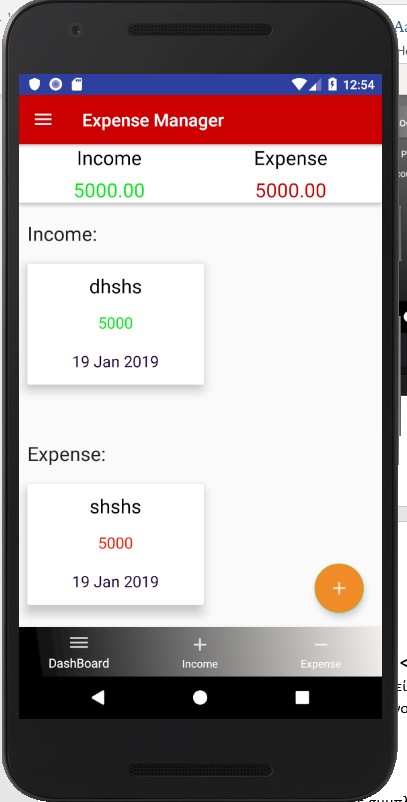


**Βήμα 1)**

Στο πεδίο τα οποία αναγράφεται **<<Email>>** συμπληρώνουμε την ηλεκτρονική διεύθυνση με την οποία έχουμε εγγραφεί στην εφαρμογή.Στο επόμενο πεδίο στο οποίο αναγράφεται **<<Password>>** συμπληρώνουμε τον κωδικό που δηλώσαμε κατά την εγγραφή μας στην εφαρμογή.

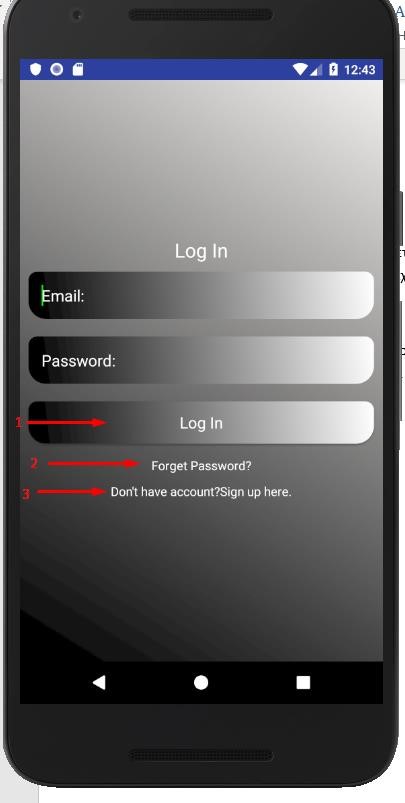
**Βήμα 2)**

Στην συνέχεια αφού έχουμε συμπληρώσει τα προσωπικά μας στοιχεία σωστά, πατάμε το κουμπί **Log In**.Με αυτην την ενέργεια η εφαρμογή εκτελεί τους κατάλληλους ελέγχους για την είσοδο μας.Εαν η σύνδεση είναι επιτυχής τότε θα εμφανιστεί το μήνυμα **“Login Successful,,** και στην συνέχεια θα οδηγηθούμε στην κύρια οθόνη της εφαρμογής μας.Στην παρακάτω εικόνα φαίνεται η κυρίως οθόνη της εφαρμογής μας.



**Περίπτωση 1)**

Σε περίπτωση που ο χρήστης έχει ξεχάσει τον κωδικό του μπορεί να πατήσει το βελάκι με τον αριθμό 2.Με αυτόν τον τρόπο θα μεταφερθεί σε μία νέα φόρμα στην οποία θα υποβάλλει ξανά τα στοιχεία του ώστε να μπορέσει να αλλάξει τον κωδικό του.



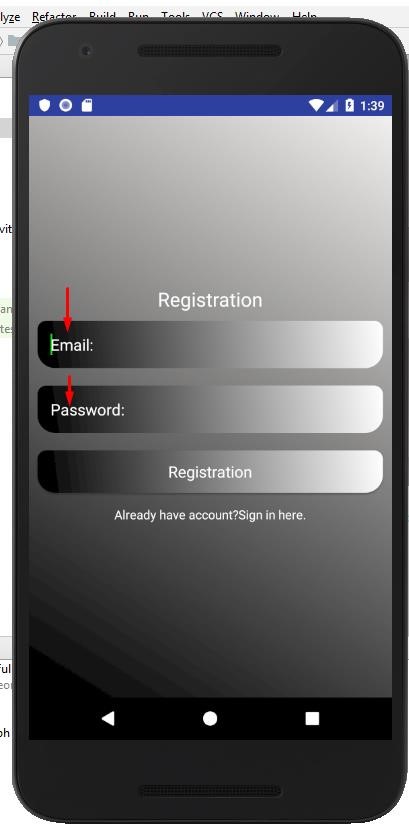
**Περίπτωση 2)**

Εάν ο χρήστης δεν έχει λογαριασμό στην εφαρμογή,θα πρέπει να πατήσει το βελάκι με τον αριθμό 3 όπως φαίνεται στην παραπάνω εικόνα.Στην συνέχεια ο χρήστης θα μεταφερθεί σε μία νέα φόρμα της οποίας η λειτουργία θα αναλυθέι στην επόμενη υποενότητα.

### 3.3 Η Δραστηριότητα Register

Στην παρακάτω δραστηριότητα ο χρήστης έχει την δυνατότητα να εγγραφεί πρώτη φορά στην εφαρμογή ώστε να είναι σε θέση να την χρησιμοποιήσει.Στην παρακάτω φόρμα ο χρήστης συμπληρώνει τα στοιχεία του αντίστοιχα όπως θα έκανε και στην φόρμα του Login.

Τα κόκκινα βελάκια υποδεικνείουν τα πεδία που θα πρέπει να τοποθετηθεί ο κωδικός και το Email.



### 3.4 Η Δραστηριότητα Reset

Σε αυτήν την δραστηριότητα βρίσκουμε την ίδια φόρμα όπως και στην δραστηριότητα **Register.**Ο χρήστης με αυτόν τον τρόπο κάνει εισαγωγή διαφορετικών στοιχείων από αυτά που είχε δηλώσει αρχικά.

### 3.5 Η Δραστηριότητα Home

Αυτή η δραστηριότητα αποτελεί τον πυρήνα της εφαρμογής μας,αφού όλες οι ενέργειες καταγραφής εξόδων πραγματοποιούνται σε αυτήν την δραστηριότητα.Στην παρακάτω εικόνα φαίνονται οι καρτέλες εσόδων/εξόδων και τα κουμπιά πλοήγησης.



**Λειτουργία αριθμημένων εικονοστοιχείων:**

**1)Καρτέλα εσόδων**

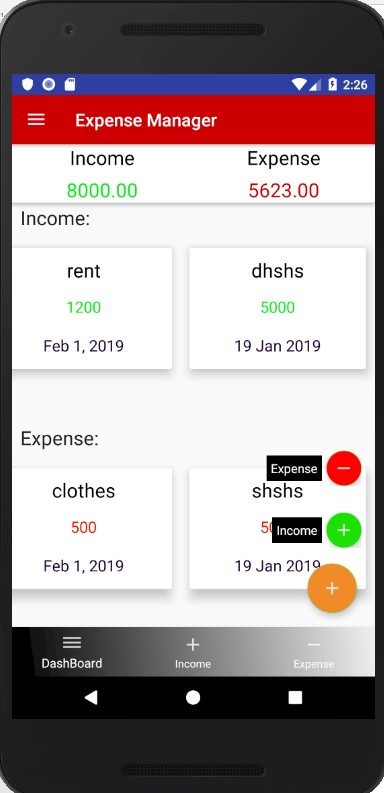
Σε αυτήν την ταμπέλα αναγράφονται περιληπτικα τα μεμονωμένα έξοδα.Δηλαδή κάθε κουτάκι περιλμβάνει ένα έσοδο περιληπτικά(ποσό,επικεφαλίδα).

**2)Καρτέλα εξόδων**

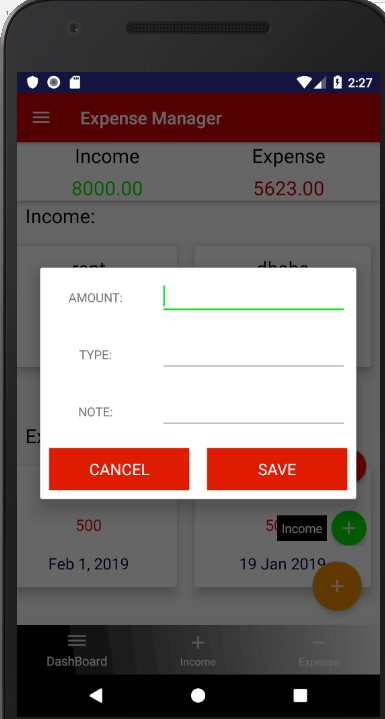
Σε αυτήν την ταμπέλα αναγράφονται περιληπτικα τα μεμονωμένα έξοδα.Δηλαδή κάθε κουτάκι περιλμβάνει ένα έξοδο περιληπτικά(ποσό,επικεφαλίδα).

**3)Εισαγωγή έσοδου/έξοδου**

Με το πάτημα του κουμπιού εμφανίζονται 2 επιλογές.Εxpense και Income.



Με το πάτημα του Expense είμαστε σε θέση να καταχωρήσουμε έξοδο ενώ αντίθετα με το πάτημα του Income καταχωρούμε έσοδο.Στην παρακάτω εικόνα γίνεται η εισαγωγή στοιχείων.



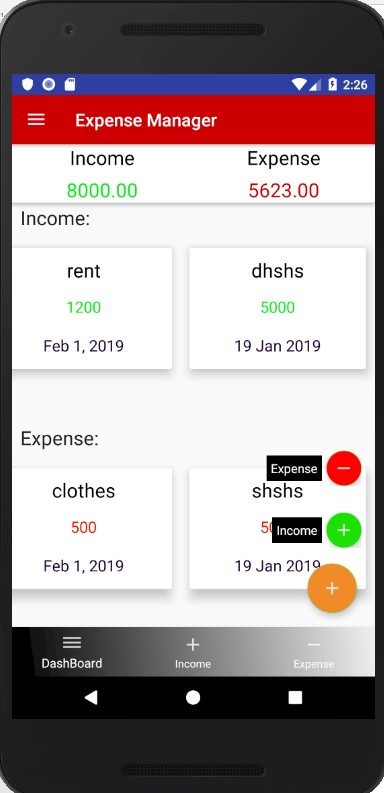
Στο πεδίο **Amount** εισάγουμε το πόσο του εξόδου/εσόδου.

Στο πεδίο **Type** εισάγουμε την κατηγορία του εξόδου/εσόδου.

Στο πεδίο **Note** εισάγουμε την περιγραφή του εξόδου/εσόδου.

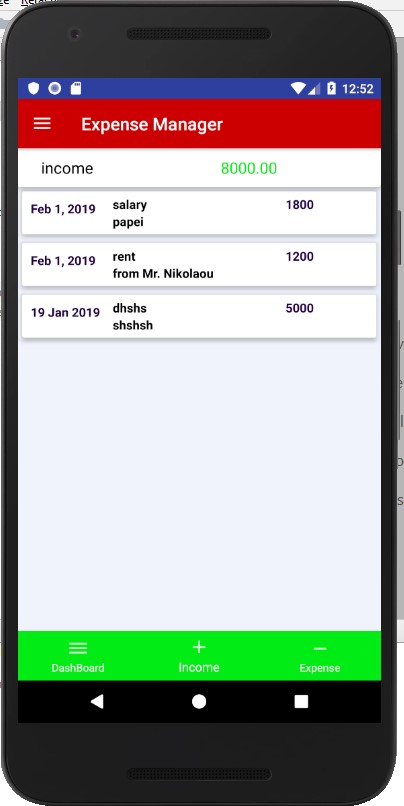
**4)Ταμπλό**

Το ταμπλό είναι η κυρίως οθόνη με όλα τα εικονοστοιχεία και τα ταμπλό εσόδων/εξόδων όπως φαίνονται στην παρακάτων εικόνα.



**5)Αναλυτική καρτέλα εσόδων**

Στην αναλυτική καρτέλα εισόδων ο χρήστης έχει στην διάθεση του όλα τα έσοδα αναλυτικά σε μορφή λίστας όπως φαίνεται παρακάτω.



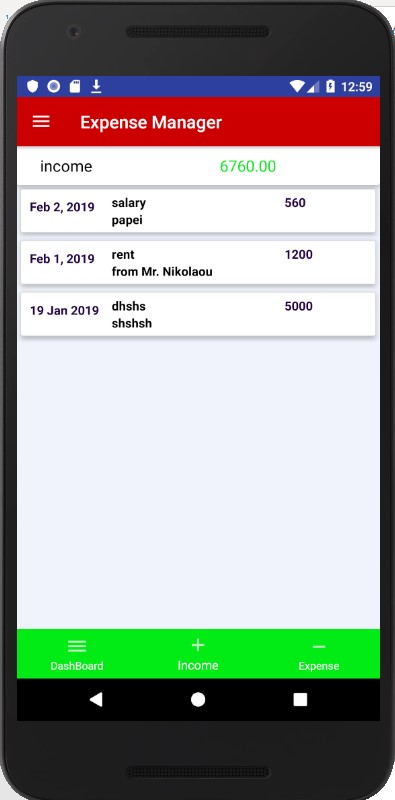
Με το πάτημα κάθε στοιχείου εμφανίζεται το παρακάτω παράθυρο.Με την επιλογή

**“Delete,,** διαγράφεται το στοιχείο.



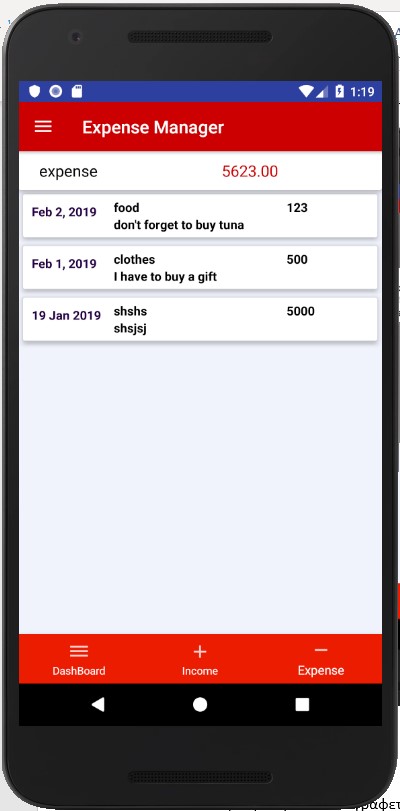
Ενώ στην περίπτωση του update καταχωρούνται οι αλλαγές που πραγματοποιήθηκαν όπως στο παρακάτω στιγμιότυπο.

*Το πρώτο στοιχείο έχει διαφορετική τιμή από πρίν καθώς πραγματοποιήθηκε αλλαγή από τον χρήστη και στην συνέχεια update.*

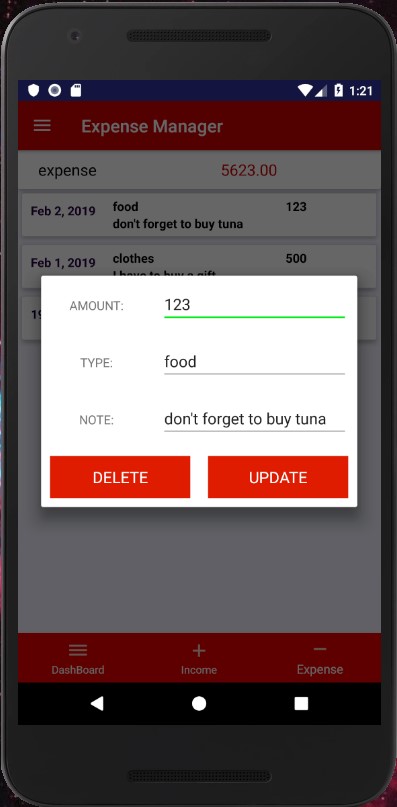


**6) Αναλυτική καρτέλα εξόδων**

Στην αναλυτική καρτέλα εισόδων ο χρήστης έχει στην διάθεση του όλα τα έσοδα αναλυτικά σε μορφή λίστας όπως φαίνεται παρακάτω.

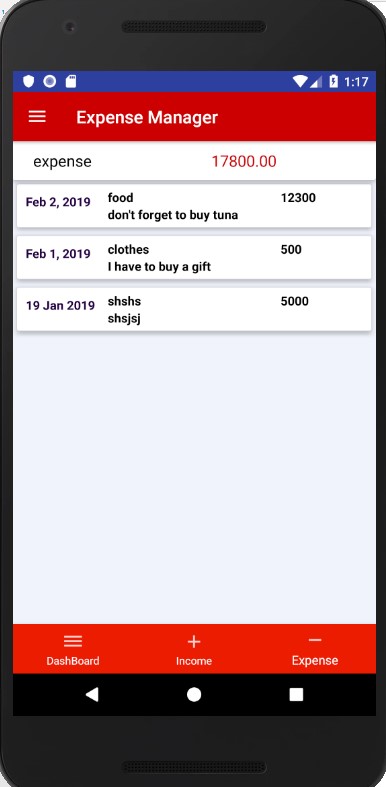


Με το πάτημα κάθε στοιχείου εμφανίζεται το παρακάτω παράθυρο.Με την επιλογή **“Delete,,**διαγράφεται το στοιχείο.



Ενώ στην περίπτωση του update καταχωρούνται οι αλλαγές που πραγματοποιήθηκαν όπως στο παρακάτω στιγμιότυπο.

*Το πρώτο στοιχείο έχει διαφορετική τιμή από πρίν καθώς πραγματοποιήθηκε αλλαγή από τον χρήστη και στην συνέχεια update.*



**7)Ταμπέλα εσόδων**

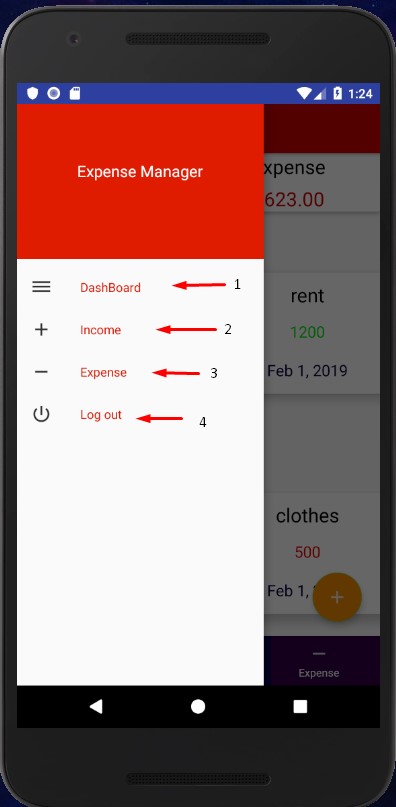
Σε αυτήν την ταμπέλα αναγράφεται το συνολικό ποσό τον εσόδων.

**8)Ταμπέλα εξόδων**

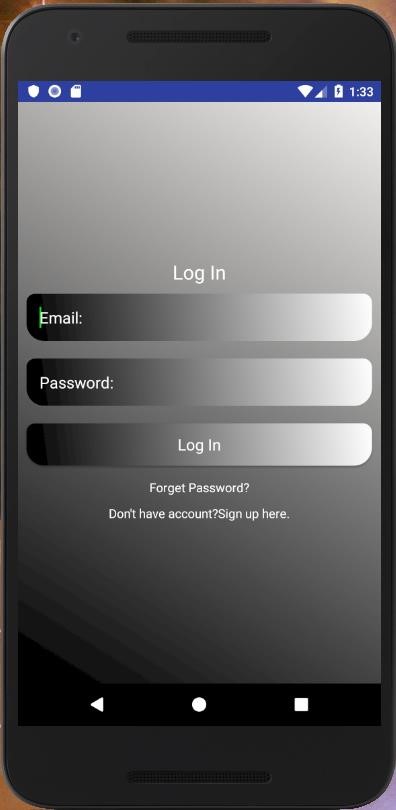
Σε αυτήν την ταμπέλα αναγράφεται το συνολικό ποσό τον εξόδων.

**9)Πλαινό μενού**

Με το πάτημα του κουμπιού θα εμφανιστεί ένα μενού επιλογών όπως φαίνεται στο παρακάτω στιγμιότυπο.



Οι επιλογές 1,2 και 3 κάνουν ανακατεύθυνση στο Ταμπλό,στην αναλυτική λίστα εσόδων και στην αναλυτική λίστα εξόδων αντίστοιχα,οι οποίες αναλύθηκαν προηγουμένως.Η 4η επιλογή δίνει την δυνατότητα στον χρήστη να αποσυνδεθεί από την εφαρμογή.Αυτό μπορεί να συμβεί σε περίπτωση που κάποιος χρήστης θέλει να χρησιμοποιήσει διαφορετικό προφίλ.Με την αποσύνδεση μας από την εφαρμογή γίνεται ανακατεύθυνση στην δραστηριότητα Login.



# 4. ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ

### 4.1. ΜΕΤΡΗΣΕΙΣ ΛΟΓΙΣΜΙΚΟΥ ΣΤΟ DEPENDENCY INJECTION PATTERN

##### 4.1.1.Εισαγωγή

Η ενότητα αυτή θα ασχοληθεί με το Dependency Injection pattern σε

Android.Για την υλοποίηση αυτού του pattern χρησιμοποιήθηκε το framework **Dagger2** της Google.

Κάθε εφαρμογή αποτελείται από αρκετές κλάσεις οι οποίες επικοινωνούν και συνεργάζονται μεταξύ τους ώστε να διεξάγουν κάποιες διαδικασίες. Συνήθως, ο τρόπος επικοινωνίας επιτυγχάνεται δημιουργώντας ένα instance ενός object. Όταν εφαρμόζουμε dependency injection τα objects αποκτούν τις μεταβλητές και τα dependencies που θα χρειαστούν από κάποια εξωτερική οντότητα η οποία κατευθύνει κάθε object στο σύστημα. Δηλαδή τα dependencies παρέχονται στο σύστημα(inject).

Βασίζεται στην έννοια της Ανάστροφης του ελέγχου( Inversion of Control), η οποία λέει ότι αντί ο κώδικας χαμηλού επιπέδου να καλεί τον κώδικα στο υψηλό επίπεδο , ο κώδικας υψηλού επιπέδου μπορεί να λάβει κώδικα χαμηλότερου επιπέδου τον οποίον μπορεί να τον καλέσει. Αυτό αναστρέφει το τυπικό μοτίβο ελέγχου που παρατηρείται στον διαδικαστικό προγραμματισμό .

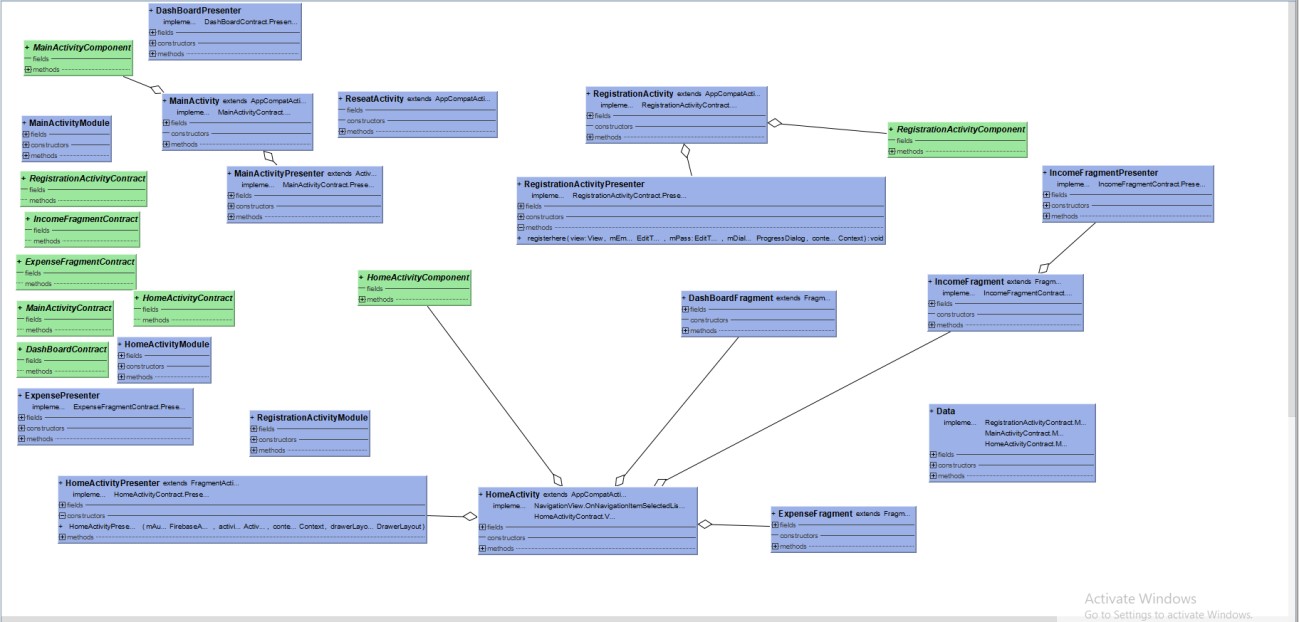
##### 4.1.2.Πως λειτουργεί?

Οι περισσότεροι dependency injectors βασίζονται στo reflection για τη δημιουργία και την παροχή των εξαρτήσεων. Αυτό συνήθως είναι πολύ αργό και χρονοβόρο. Επίσης εκτελεί ανάλυση των εξαρτήσεων κατά το χρόνο εκτέλεσης η οποία προκαλεί απροσδόκητα σφάλματα και διακόπτει την εφαρμογή.

Από την άλλη πλευρά, μερικοί injectors χρησιμοποιούν ένα Pre-compiler που δημιουργεί όλες τις κλάσεις (γράφημα αντικειμένων) που πρέπει να λειτουργήσει χρησιμοποιώντας τον annotation processor. Ένας annotation processor είναι ένας τρόπος για να διαβάστουν τα compiled αρχεία κατά τη διάρκεια του χρόνου κατασκευής ώστε να δημιουργήθουν αρχεία κώδικα πηγής που θα χρησιμοποιηθούν στο πρόγραμμα. Επομένως, εκτελούν την ανάλυση εξαρτήσεων πριν την εκτέλεση της εφαρμογής για την αποφυγή απροσδόκητων σφάλματων.

##### 4.1.3.Διάγραμμα κλάσης

Στο παρακάτω διάγραμμα κλάσεων μπορούμε να δούμε τον τρόπο με τον οποίο είναι δομημένο το πρόγραμμα μας. Επίσης, απεικονίζονται και οι σχέσεις μεταξύ των κλάσεων. Οι κλάσεις οι οποίες έχουν πράσινο ανοικτό χρώμα είναι εκείνες οι οποίες γίνονται inject ώστε να παρέχουν τα dependencies στις κλάσεις που τα χρειάζονται.



##### 4.1.4.Οφέλη από το DI

Το Dependency Injection(DI) μειώνει τον περιττό κώδικα και κάνει την διαδικασία ανάπτυξης ομαλή. Η σωστή εφαρμογή του dependency injection στα προγράμματα μας, επιτρέπει να έχουμε:

* Χαλαρή σύζευξη μεταξύ των οντοτήτων μας (loose coupling).
* Εύκολη διαδικασία testing του προγράμματος μας.
* Επαναχρησιμοποιούμενο κώδικα.

##### 4.1.5 Annotation Processor

O annotation processor βοηθάει στην εύκολη κατανόηση και ανάπτυξη των προγραμμάτων μας. Στον κώδικα χρησιμοποιήσαμε τα παρακάτω annotations:

* **@Inject:**Κάνει αίτημα ώστε να έχει την παροχή των εξαρτήσεων. Μπορεί να χρησιμοποιηθεί σε constructors,fields και methods.
* **@Module και Provides:**Προσδιορίζουν κλάσεις και μεθόδους οι οποίες παρέχουν τις εξαρτήσεις.
* **@Component:**Ενεργοποιεί τα παρεχόμενα modules και χρησιμοποιείτε για την εκτέλεση της παροχής των εξαρτήσεων.

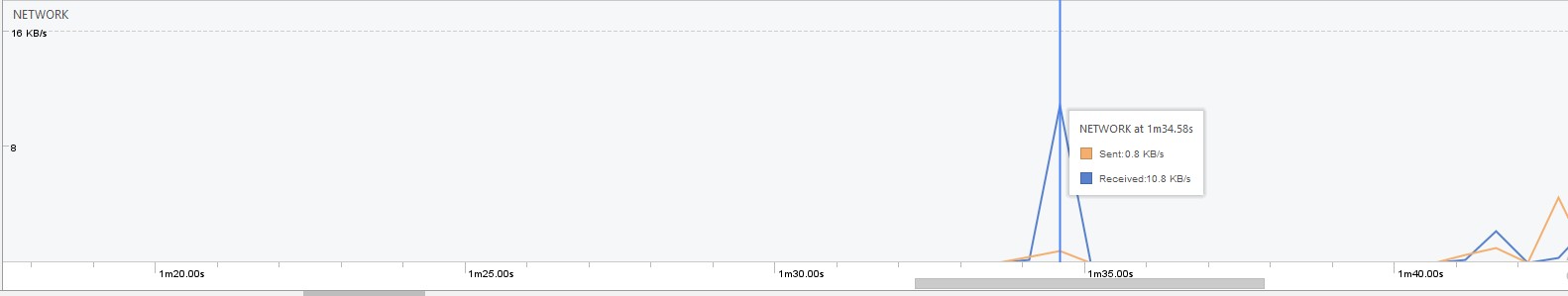
##### 4.1.6.Profiling της εφαρμογής με την χρήση του Android Profiler.

Με την χρήση του Android Profiler παίρνουμε τις παρακάτω τιμές όταν εκτελούμε τις βασικές ενέργειες στην εφαρμογή μας. Για παράδειγμα:

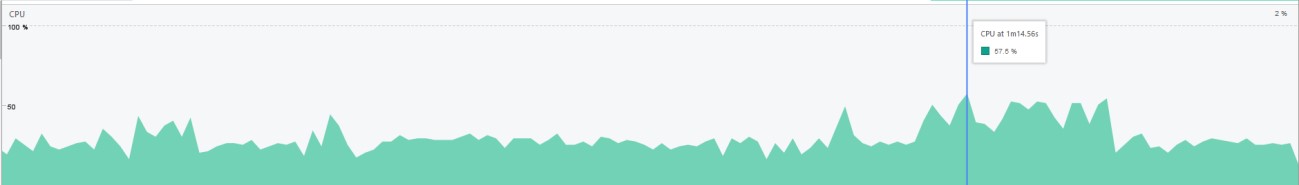
***Login:***

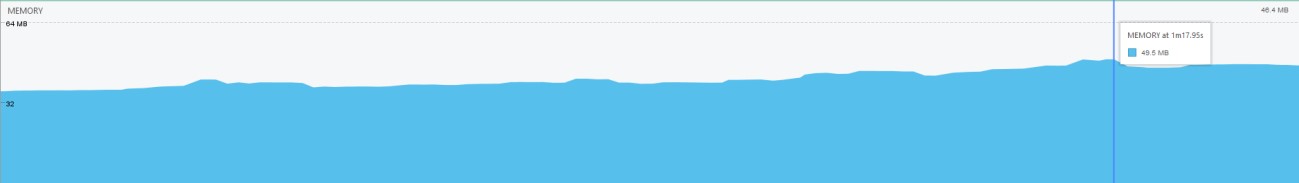






***Register:***



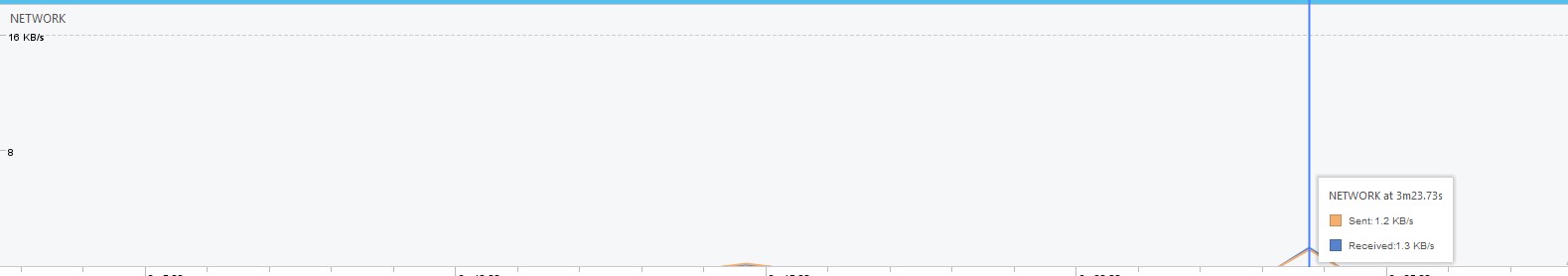




***Insert:***



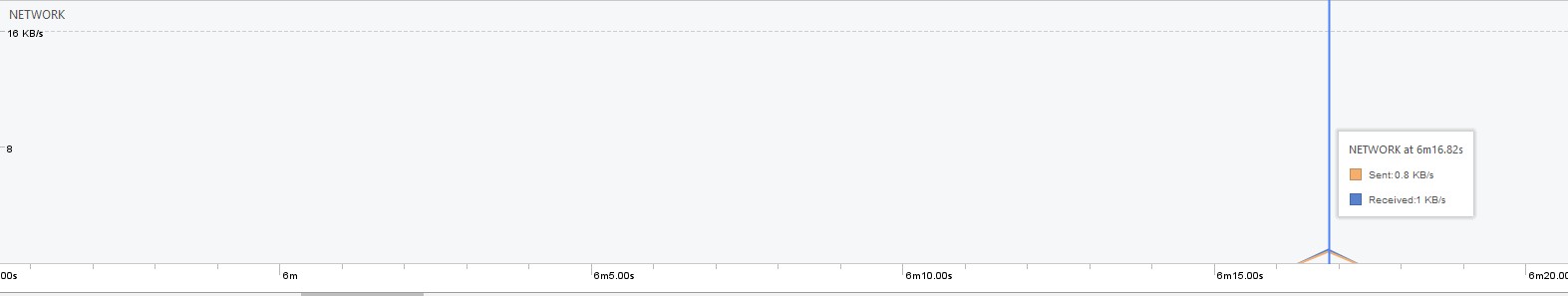




***Delete:***



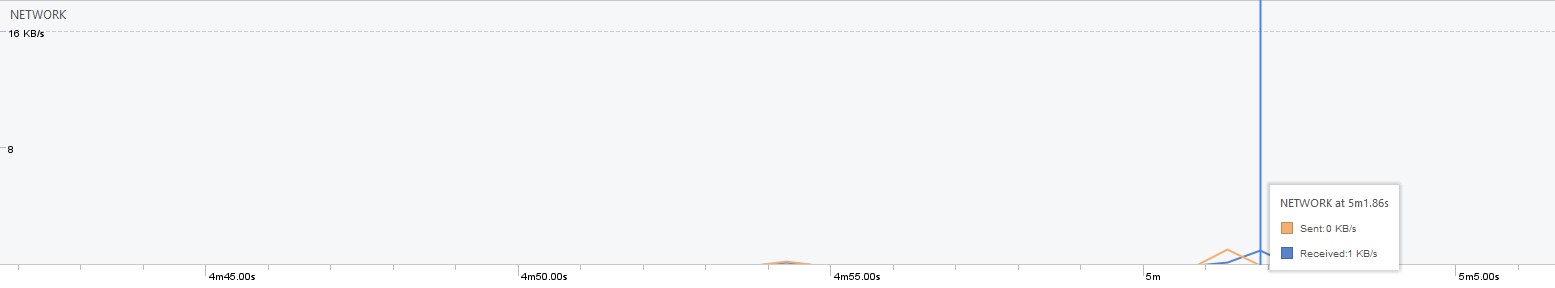




***Update:***







### 4.2. ΜΕΤΡΗΣΕΙΣ ΛΟΓΙΣΜΙΚΟΥ ΣΤΟ ΜVP PATTERN

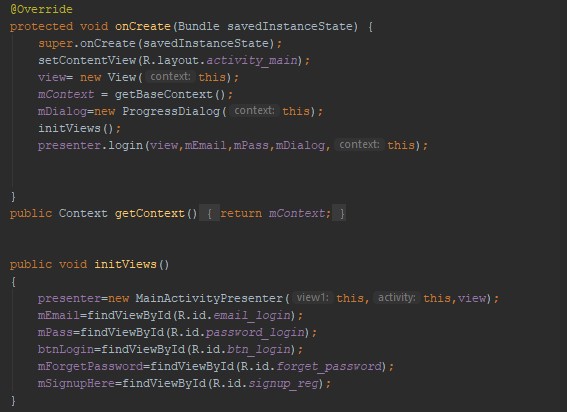
##### 4.2.1.Εισαγωγή

Η ενότητα αυτή θα ασχοληθεί με το MVP pattern σε Android.To pattern που θα ασχοληθούμε σε αυτή την ενότητα ασχολείται με 3 οντότητες.Το Model(data),Το View(γραφικά στοιχεία), τον Presenter(middle-man)

##### 4.2.2.Πως λειτουργεί?

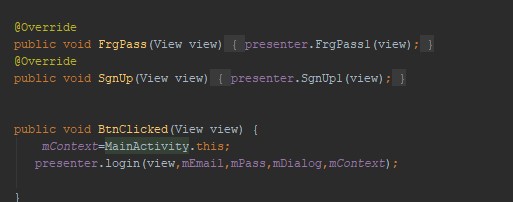
Ο λόγος που γίνεται αυτός ο διαχωρισμός είναι για καλύτερο testing. Θέλουμε να δημιουργήσουμε ένα View με όσο το δυνατόν λιγότερη λειτουργικότητα. Βέβαια αυτό είναι μια πρόκληση καθώς στο περιβάλλον Android έχουμε τα Activities τα οποία συνδυάζουν view και λειτουργικότητα σε ένα. Επομένως με το MVP στέλνουμε όλη την λογική του προγράμματος μας στον Presenter ετσι ωστε να διαχειριστεί αυτός,την επιχειρησιακή λογική αλλά και το update των views , την σύνδεση με τα API και τις βάσεις δεδομένων. Στην περίπτωση μας χρησιμοποιούμε το Firebase.

*Για παράδειγμα στην κυρίως κλάση δημιουργούμε ένα object της κλάσης Presenter:*

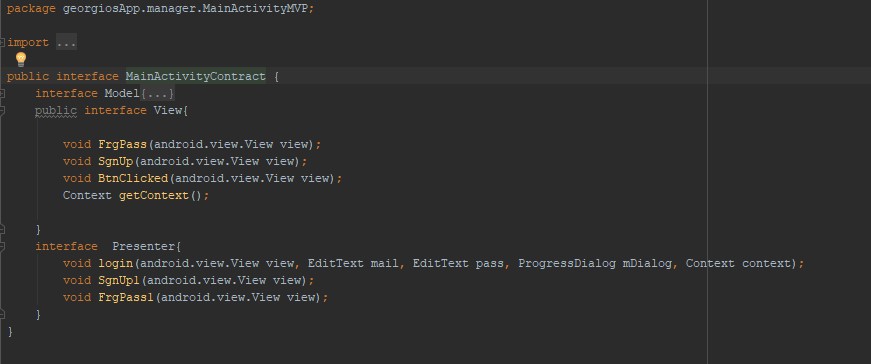


Στην μέθοδο InitViews() γίνεται η αρχικοποίηση του.

Και στις παρακάτω μεθόδους καλούμε την αντίστοιχη μέθοδο του Presenter.

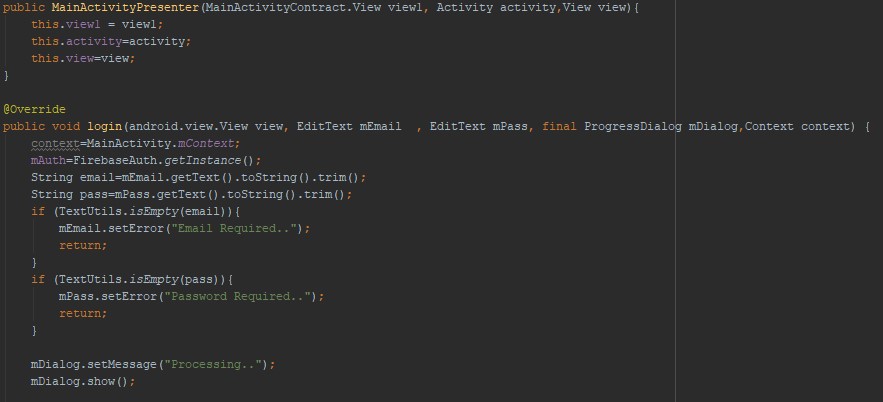


Στην συνέχεια όλες αυτές οι μέθοδοι είναι σε ένα Interface που περιέχει το Interface του Model,View και του Presenter:



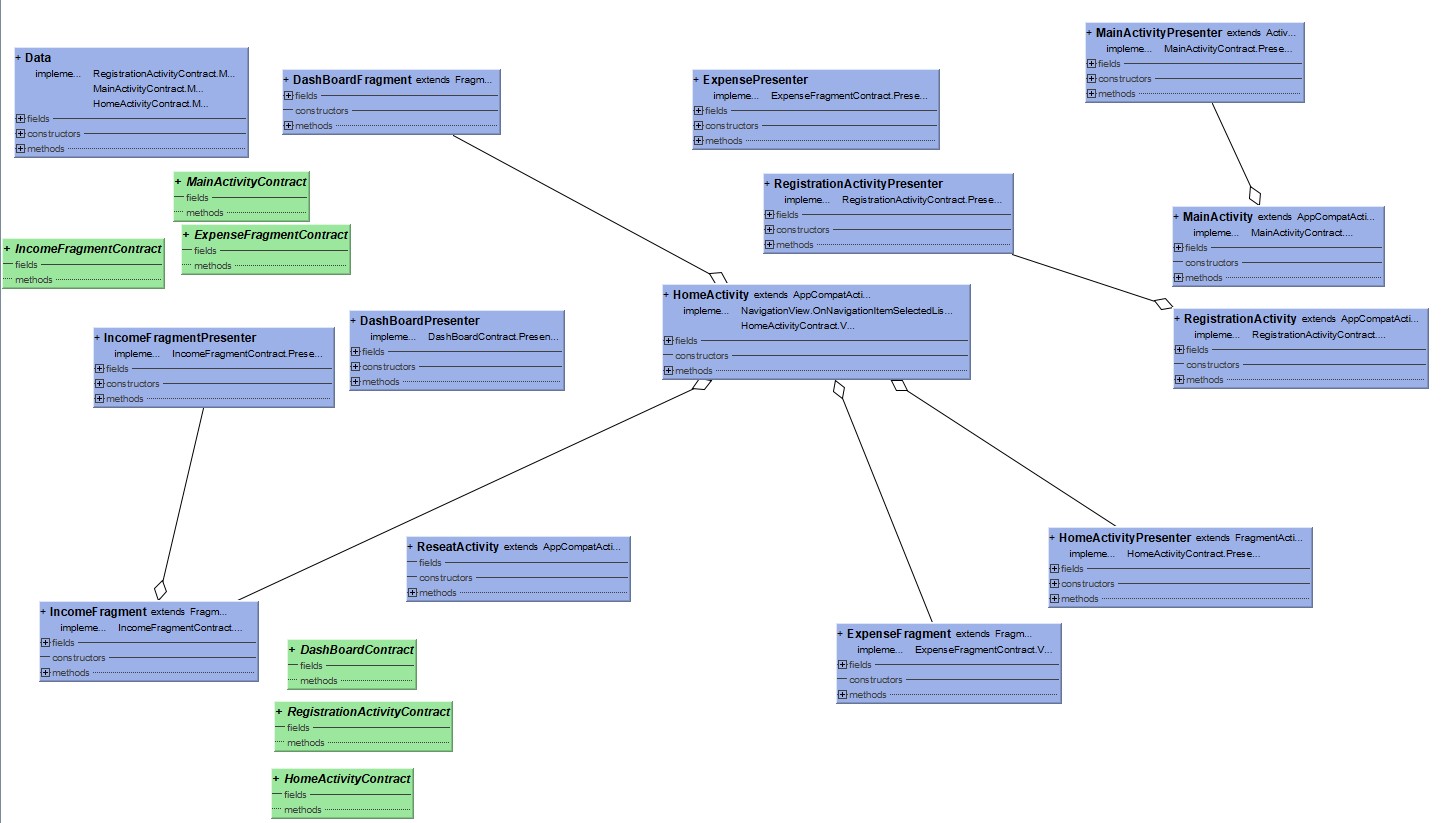
Το activity κάνει Implement τo View interface και ο Presenter το Presenter Interface.

*Στο παρακάτω στιγμιότυπο μπορούμε να δούμε την λογική τον κλάσεων που υλοποιείτε στον Presenter:*



### 4.2.3.Διάγραμμα κλάσης

Στο παρακάτω διάγραμμα κλάσης απεικονίζεται το MVP Pattern. Από το διάγραμμα μπορούμε να δούμε ότι κάθε **Activity** έχει έναν Presenter ο οποίος εκτελεί την επιχειρησιακή λογική, αφήνοντας το Activity σαν μία διάταξη εικονοστοιχείων. Ο λόγος για τον οποίο γίνεται αυτό είναι για τον διαχωρισμό και το testing των διαφορετικών layers του προγράμματος μας.



### 4.2.4 Οφέλη από το MVP

* Σαφής διαχωρισμός των ευθυνών μεταξύ των στοιχείων. Αυτός ο διαχωρισμός επιτρέπει την ευκολότερη κατανόηση και συντήρηση της βάσης κώδικα.

* Modularity. To modularity μάς επιτρέπει να μεταβούμε σε μια διαφορετική εφαρμογή της συνιστώσας προβολής για να αλλάξουμε εντελώς το interface εφαρμογής, ενώ όλα τα υπόλοιπα στοιχεία παραμένουν άθικτα.

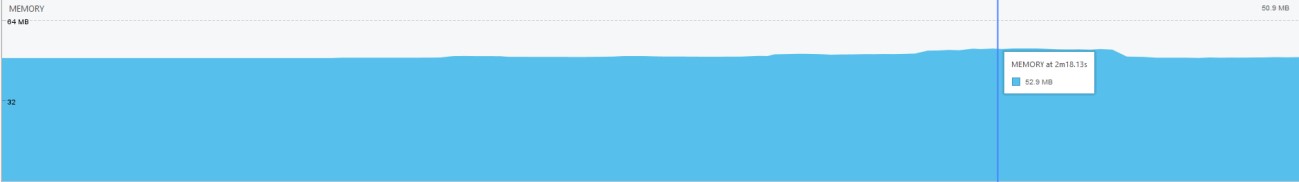
* Ευκολότερο testing. Δεδομένου ότι υπάρχουν καλά καθορισμένα όρια μεταξύ των συστατικών, γίνεται πολύ πιο εύκολο να δοκιμαστεί κάθε συστατικό μεμονωμένα

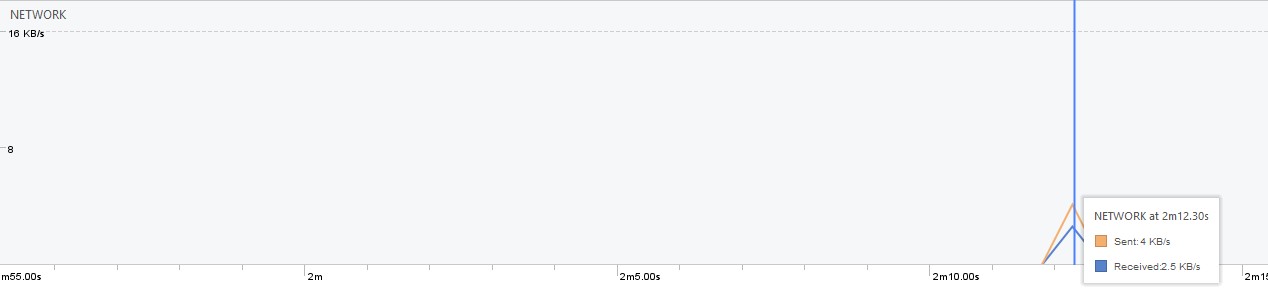
### 4.2.5.Profiling της εφαρμογής με την χρήση του Android Profiler.

Από τα παρακάτω screenshot του android profiler παίρνουμε κάποια στοιχεία για την χρήση της CPU,το δίκτυο και την μνήμη.

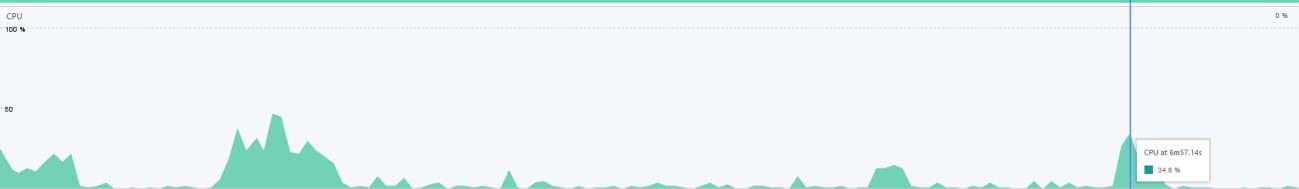
Για παράδειγμα στο από κάτω στιγμιότυπο έχουμε το **Login** στην εφαρμογή . **Login**

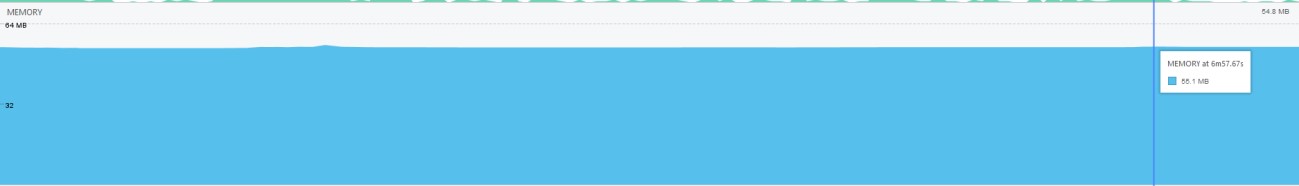


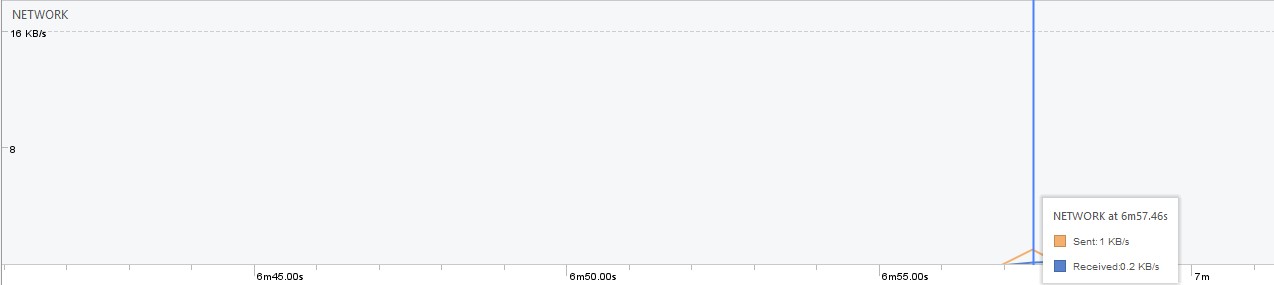




Στα παρακάτω screenshot βλέπουμε την επίδοση της εφαρμογής στις βασικές τις ενέργειες όπως **Update**

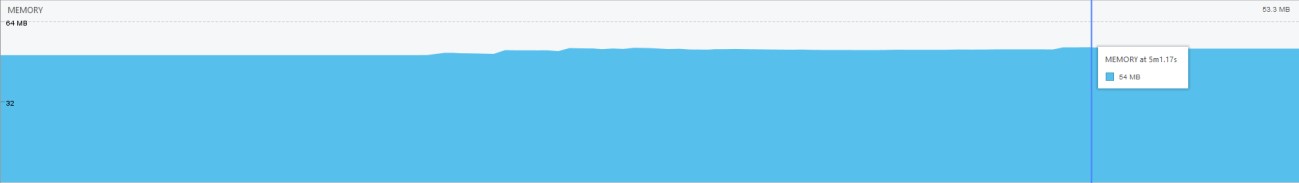






**Insert**







**Delete**







## 4.3 ΜΕΤΡΗΣΕΙΣ ΛΟΓΙΣΜΙΚΟΥ ΣΤΟ PROXY PATTERN

### 4.3.1.Εισαγωγή

Η ενότητα αυτή θα ασχοληθεί με το Proxy Pattern σε Android.To Pattern αυτό κατηγοριοποιείται στα structural design patterns καθώς μια άλλη κλάση παρέχει την λειτουργικότητα της βασικής κλάσης που στην περίπτωση μας είναι τα activities.

### 4.3.2.Πως λειτουργεί?

Το Proxy pattern χρησιμοποιείται σαν μια πύλη για την ανάκτηση πληροφορίας. Για παράδειγμα, στην καθημερινή μας ζωή η πιστωτική κάρτα είναι μια δίοδος για τον προσωπικό μας λογαριασμό ο οποίος προστατεύεται από τον οργανισμό της τράπεζας .

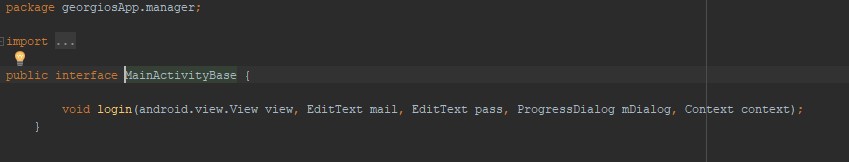
Με την πιστωτική κάρτα είμαστε σε θέση να δούμε τα περιεχόμενα του τραπεζικού λογαριασμού. Έτσι αντίστοιχα, το pattern αυτό ελέγχει την είσοδο στα αντικείμενα που προστατεύει ώστε να κρατάει τις ευαίσθητες πληροφορίες.

Στην εφαρμογή μας χρησιμοποιήσαμε proxy στη method onClick() ώστε να στείλουμε τις ευαίσθητες σε μια άλλη κλάση που προστατεύει τις πληροφορίες. Με την εκτέλεση του κώδικα η κλάση αυτή θα δώσει πρόσβαση ή όχι.

*Με τον παρακάτω κώδικα δίνεται πρόσβαση ή όχι στην εφαρμογή.*



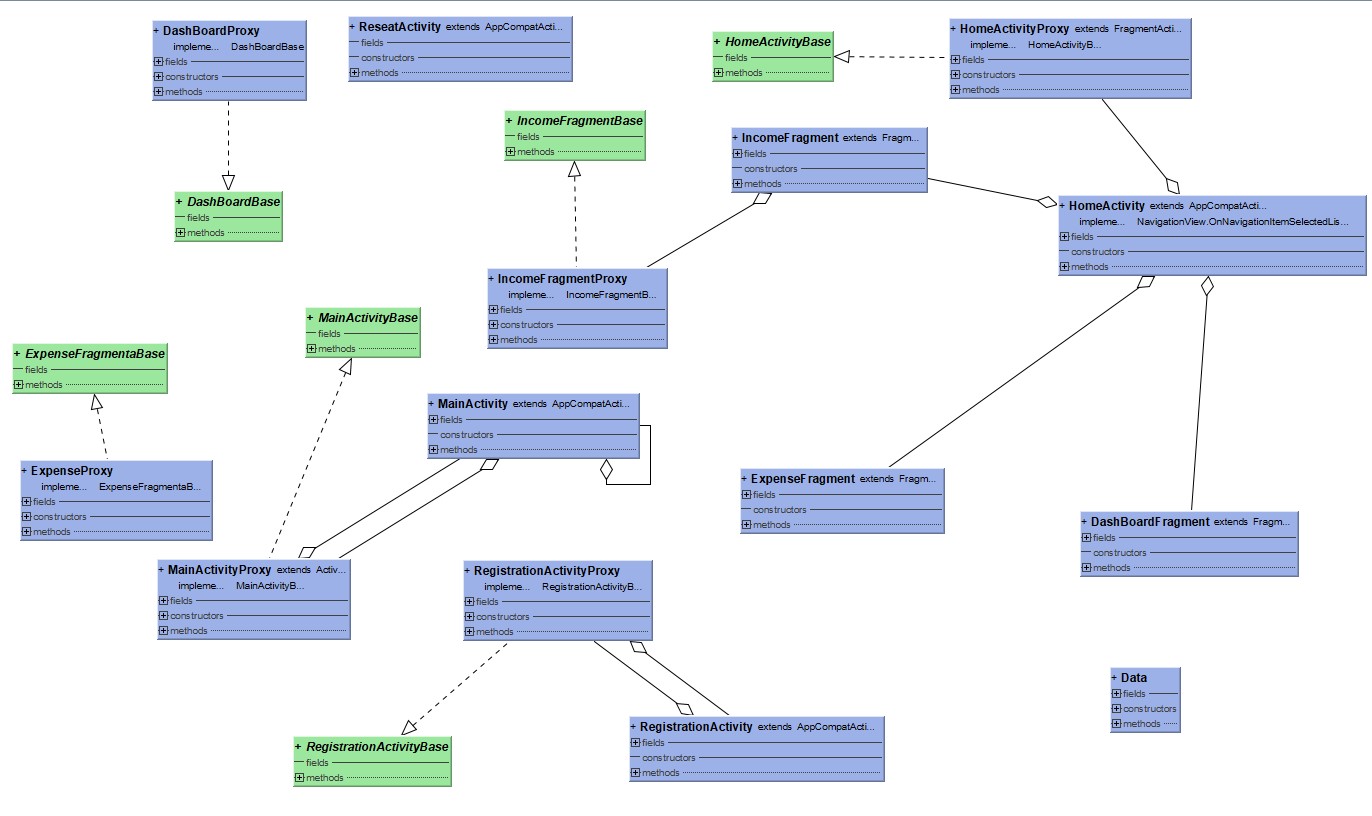
Για να μπορέσουμε να δώσουμε την λειτουργικότητα μιας κλάσης σε μια άλλη χρειαστήκαμε ένα interface όπως το παρακάτω.



Έπειτα οι δυο μας κλάσεις κάνουν implement το interface ώστε η proxy κλάση να χρησιμοποιήσει αυτό το method.

### 4.3.3.Διάγραμμα κλάσης

Στο παρακάτω διάγραμμα κλάσεων απεικονίζονται οι σχέσεις μεταξύ των κλάσεων στο Proxy Pattern. Στο Proxy Pattern οι κλάσεις λειτουργούν σαν δικλείδα ασφαλείας ώστε να μην εκθέτουν ευαίσθητες πληροφορίες.



### 4.3.4.Οφέλη του Proxy Pattern

Ένα από τα πλεονεκτήματα του Proxy Pattern είναι η ασφάλεια. Αυτό το μοτίβο αποφεύγει την επικάλυψη αντικειμένων που μπορεί να έχουν τεράστιο μέγεθος και αρκετή μνήμη. Αυτό με τη σειρά του αυξάνει την απόδοση της εφαρμογής. Ο απομακρυσμένος διακομιστής μεσολάβησης διασφαλίζει επίσης την ασφάλεια εγκαθιστώντας το τοπικό πληρεξούσιο κώδικα στο μηχάνημα-πελάτη και στη συνέχεια, αποκτώντας πρόσβαση στον διακομιστή με τη βοήθεια του απομακρυσμένου κώδικα.

### 4.3.5.Profiling της εφαρμογής με την χρήση του Android Profiler.

Από τα παρακάτω screenshot του android profiler παίρνουμε κάποια στοιχεία για την χρήση της CPU,το δίκτυο και την μνήμη.

Για παράδειγμα στο παρακάτω στιγμιότυπο έχουμε το Login στην εφαρμογή . **Login**



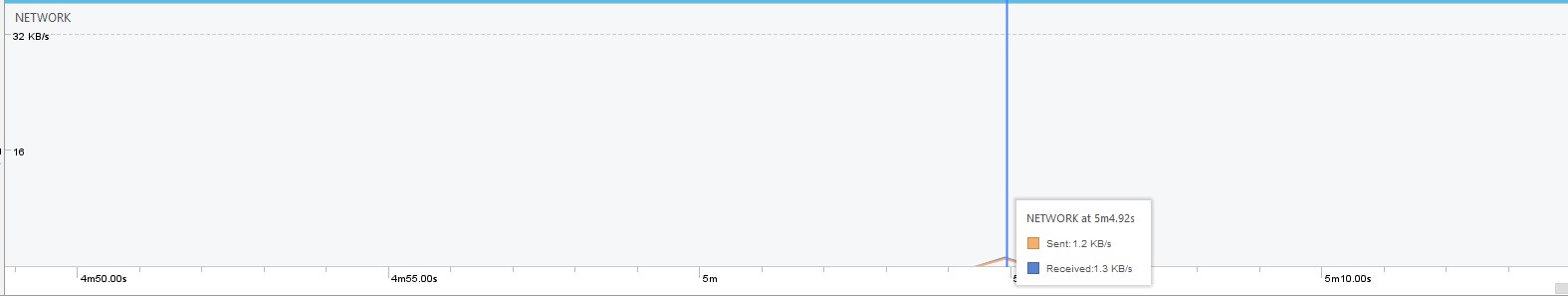




Στα παρακάτω screenshot βλέπουμε την επίδοση της εφαρμογής στις βασικές τις ενέργειες όπως **Update**



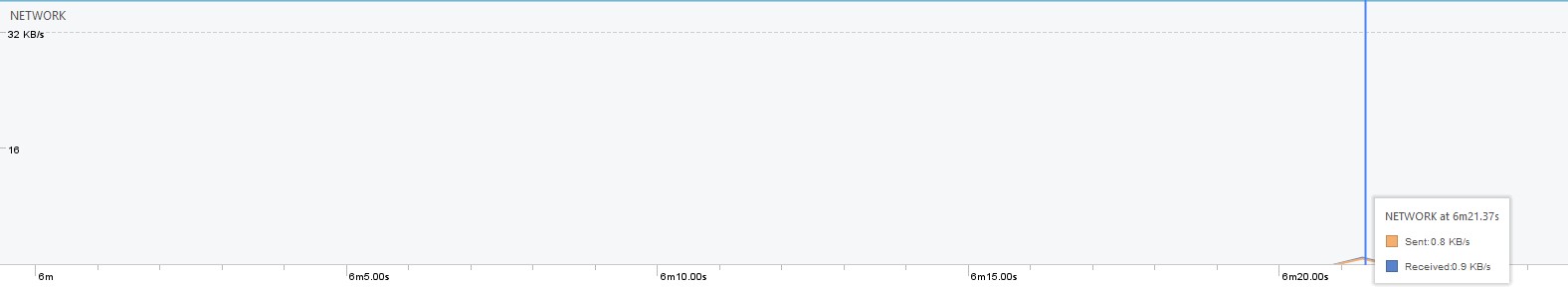




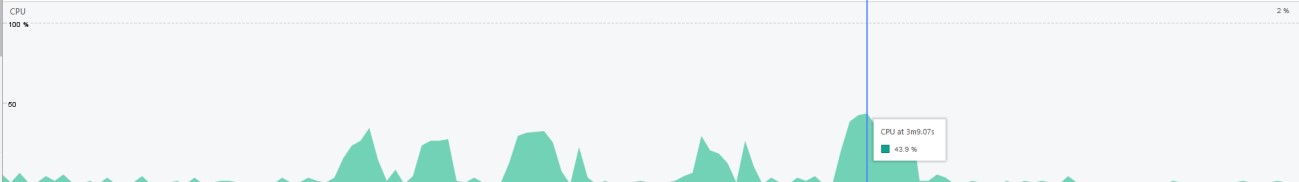
**Delete**

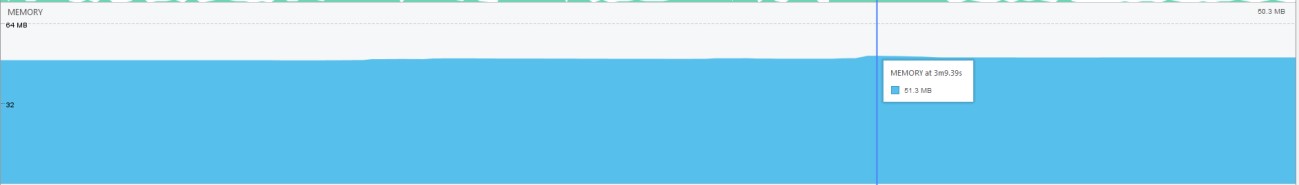






**Insert**







## 4.4 ΜΕΤΡΗΣΕΙΣ ΛΟΓΙΣΜΙΚΟΥ ΣΤΟ TEMPLATE PATTERN

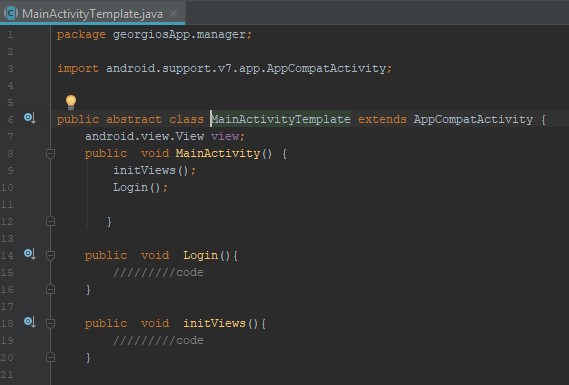
### 4.4.1.Εισαγωγή

Η ενότητα αυτή θα ασχοληθεί με το Template pattern σε Android.To pattern που θα ασχοληθούμε σε αυτή την ενότητα κατηγοριοποιείται στα Behavioral Design Pattern καθώς ανάλογα με τις ανάγκες του προγράμματος τα αντικείμενα υιοθετούν διαφορετική συμπεριφορά κάνοντας override μεθόδους από το abstract class.

### 4.4.2.Πως λειτουργεί?

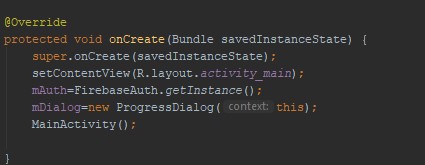
To Template χρησιμοποιεί abstract classes ώστε να ορίζει καθορισμένους τρόπους εκτέλεσης των μεθόδων ενός προγράμματος . Ουσιαστικά, αυτό το pattern ορίζει τον σκελετό ενός προγράμματος το οποίο εκτελείται βήμα βήμα αφήνοντας κάποιες ενέργειες στις υποκλάσεις του κυρίως προγράμματος.

*Όπως για παράδειγμα στην παρακάτω abstract κλάση:*



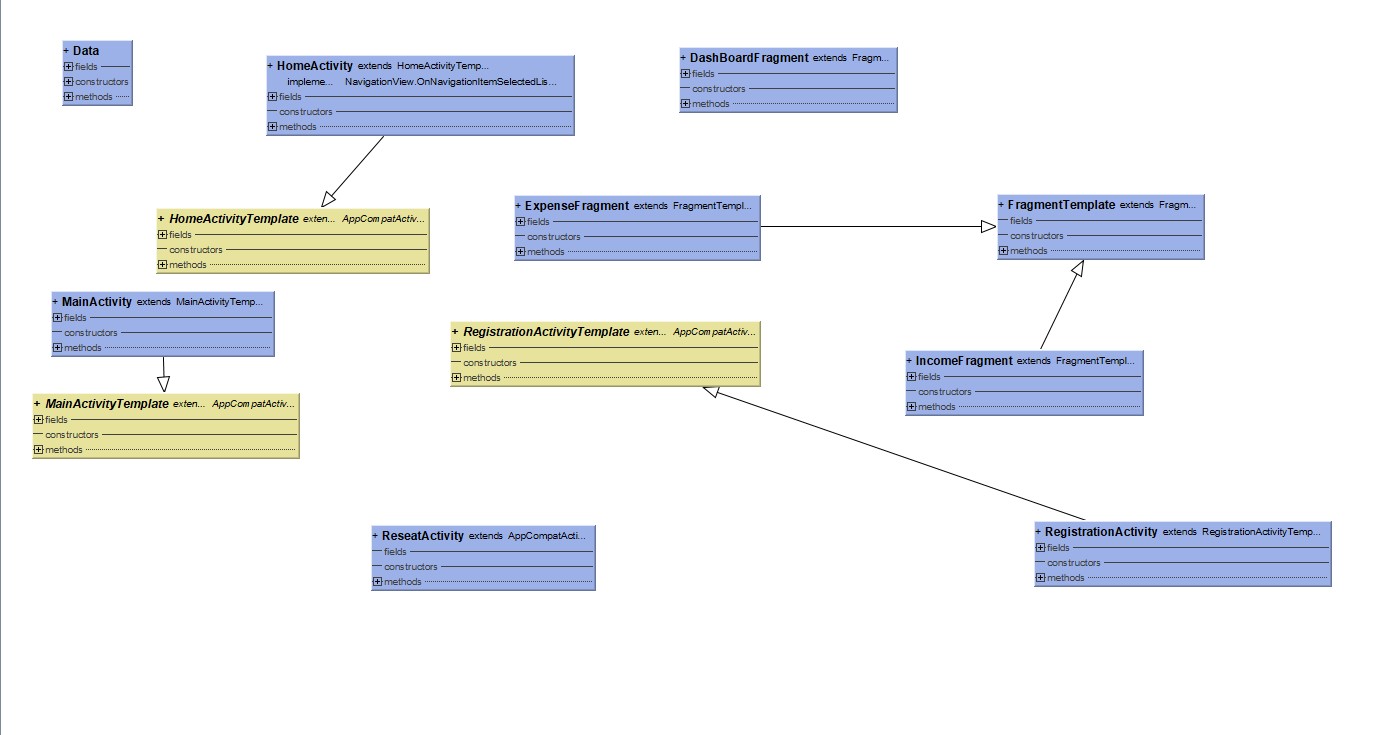
Οι μέθοδοι Login() και InitViews() θα γίνουν override από την κυρίως κλάση. Στην μέθοδο MainActivity() ορίζουμε την σειρά τον βημάτων με την οποία πρέπει να εκτελεστούν οι μέθοδοι όπως θα κάναμε και σε έναν αλγόριθμο. Στην συνέχεια, καλούμε την μέθοδο MainActivity() από το Activity μας .

*Μπορούμε να δούμε αυτό το παράδειγμα στο παρακάτω στιγμιότυπο*



### 4.4.3.Διάγραμμα κλάσης

Στο παρακάτω διάγραμμα κλάσεων απεικονίζονται οι σχέσεις μεταξύ των κλάσεων στο Template Pattern. Στο Template Pattern οι κλάσεις με χρώμα μπεζ είναι τα templates δηλαδή ένας αλγόριθμος εκτέλεσης συναρτήσεων του προγράμματος μας ώστε να έχει λογική συνέχεια. Παρατηρούμε ότι το Expense Fragment και το Income **Fragment** κάνουν **implement** το Fragment Template. Αυτό συμβαίνει διότι τα 2 Fragment είναι ίδια σε αλγοριθμική δομή. Επομένως, γίνεται επαναχρησιμοποίηση κώδικα.



### 4.4.4 Οφέλη του Template Pattern

* Όταν η συμπεριφορά ενός αλγορίθμου μπορεί να ποικίλει, αφήνουμε τις υποκατηγορίες να εφαρμόζουν τη συμπεριφορά μέσω του override

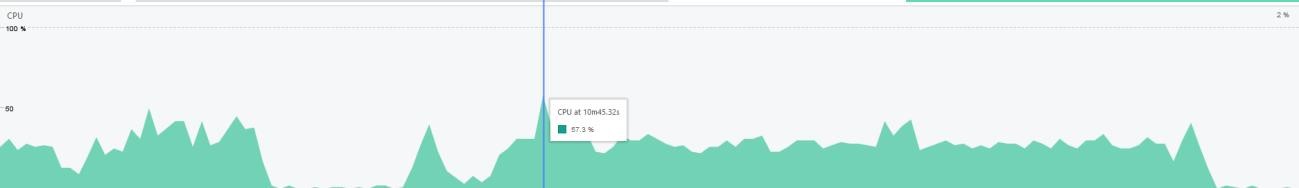
* Όταν θέλουμε να αποφύγετε την αλληλοεπικάλυψη κώδικα, εφαρμόζοντας παραλλαγές του αλγορίθμου σε υποκατηγορίες

* Όταν θέλουμε να ελέγξουμε το σημείο που επιτρέπεται η υποκατηγορία.

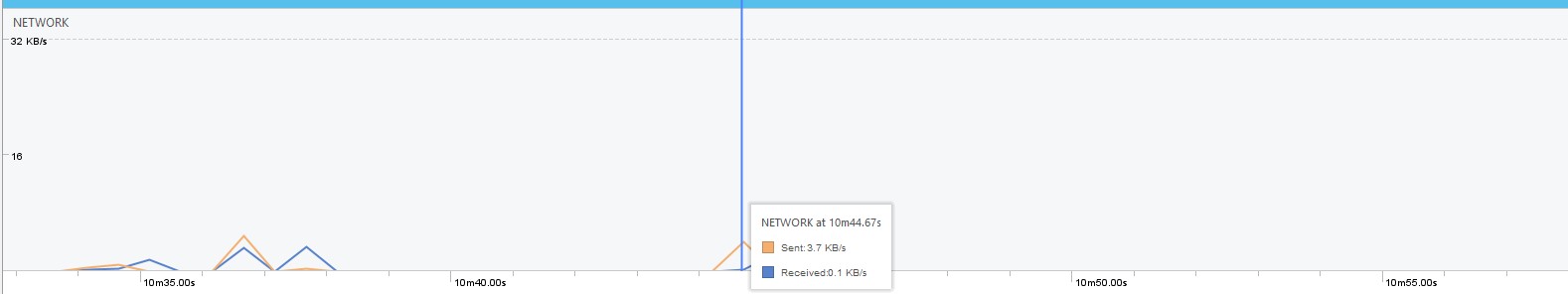
### 4.4.5.Profiling της εφαρμογής με την χρήση του Android Profiler.

Από τα παρακάτω screenshot του android profiler παίρνουμε κάποια στοιχεία για την χρήση της CPU,το δίκτυο και την μνήμη.

Για παράδειγμα στο από κάτω στιγμιότυπο έχουμε το **Login** στην εφαρμογή . **Login**

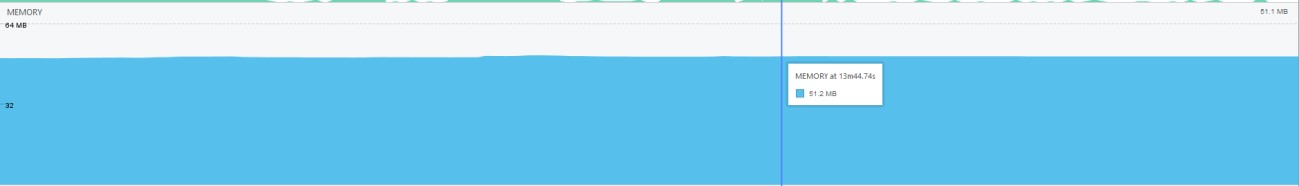


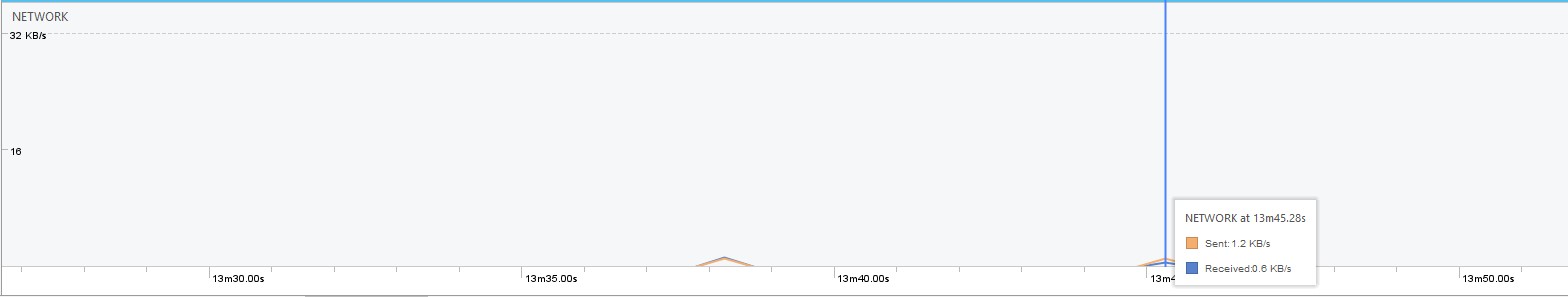




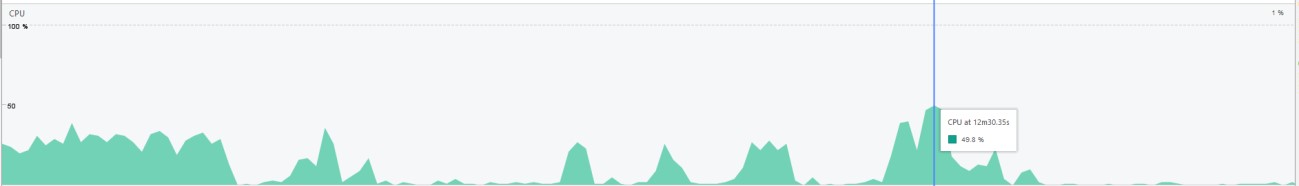
Στα παρακάτω screenshot βλέπουμε την επίδοση της εφαρμογής στις βασικές τις ενέργειες όπως **Update**

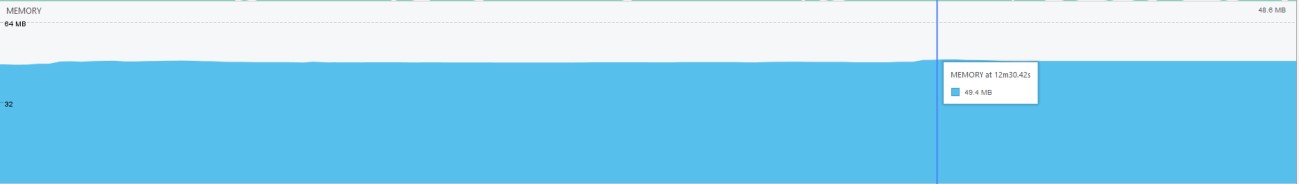


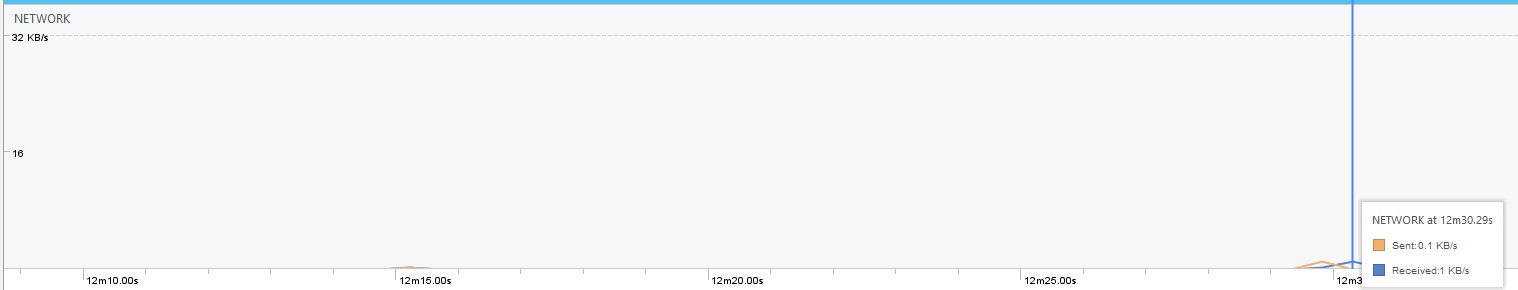




**Insert**



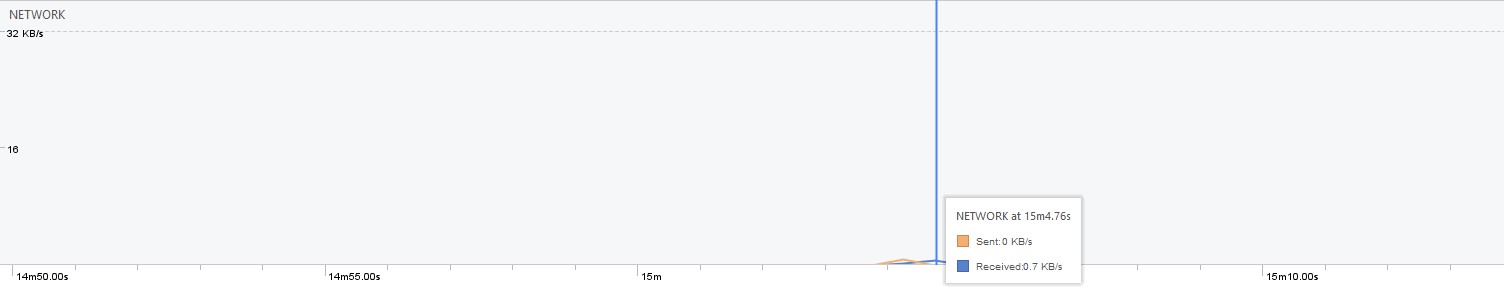




**Delete**







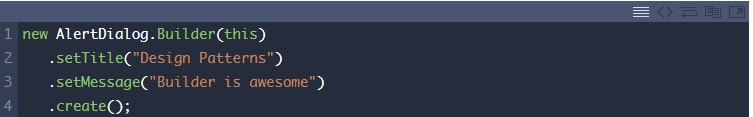
## 4.5 ΜΕΤΡΗΣΕΙΣ ΛΟΓΙΣΜΙΚΟΥ ΣΤΟ BUILDER PATTERN

### 4.5.1.Εισαγωγή

Η ενότητα αυτή θα ασχοληθεί με το Builder pattern σε Android.To pattern που θα ασχοληθούμε σε αυτή την ενότητα κατηγοριοποιείται στα Creational Design Pattern καθώς παρέχουν λύσεις για την δημιουργία κλάσεων και αντικειμένων(objects).

### 4.5.2.Πώς λειτουργεί?

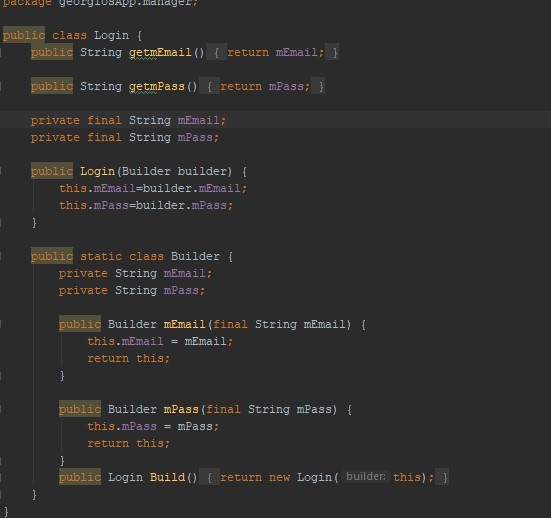
To Builder Pattern την δημιουργία αντικειμένων με πολύ απλό και καθαρό τρόπο. Είναι πολύ εύχρηστο όταν έχουμε κάποιες κλάσεις οι οποίες δέχονται πολλές παραμέτρους. Χρησιμοποιώντας το Βuilder Pattern μπορούμε να ορίσουμε αυτές τις παραμέτρους με μια σειρά από μεθόδους. Ένα πολύ απλό παράδειγμα που συναντάμε συχνά στο Android είναι η κλάση **AlertDialog.Builder()** Για παράδειγμα:



Στην εφαρμογή μας χρησιμοποιήσαμε Builder ώστε να πάρουμε τις πληροφορίες από τα Textbox όσο αναφορά την εγγραφή και την είσοδο του χρήστη. Στην παρακάτω εικόνα μπορούμε να δούμε το παράδειγμα του κώδικα

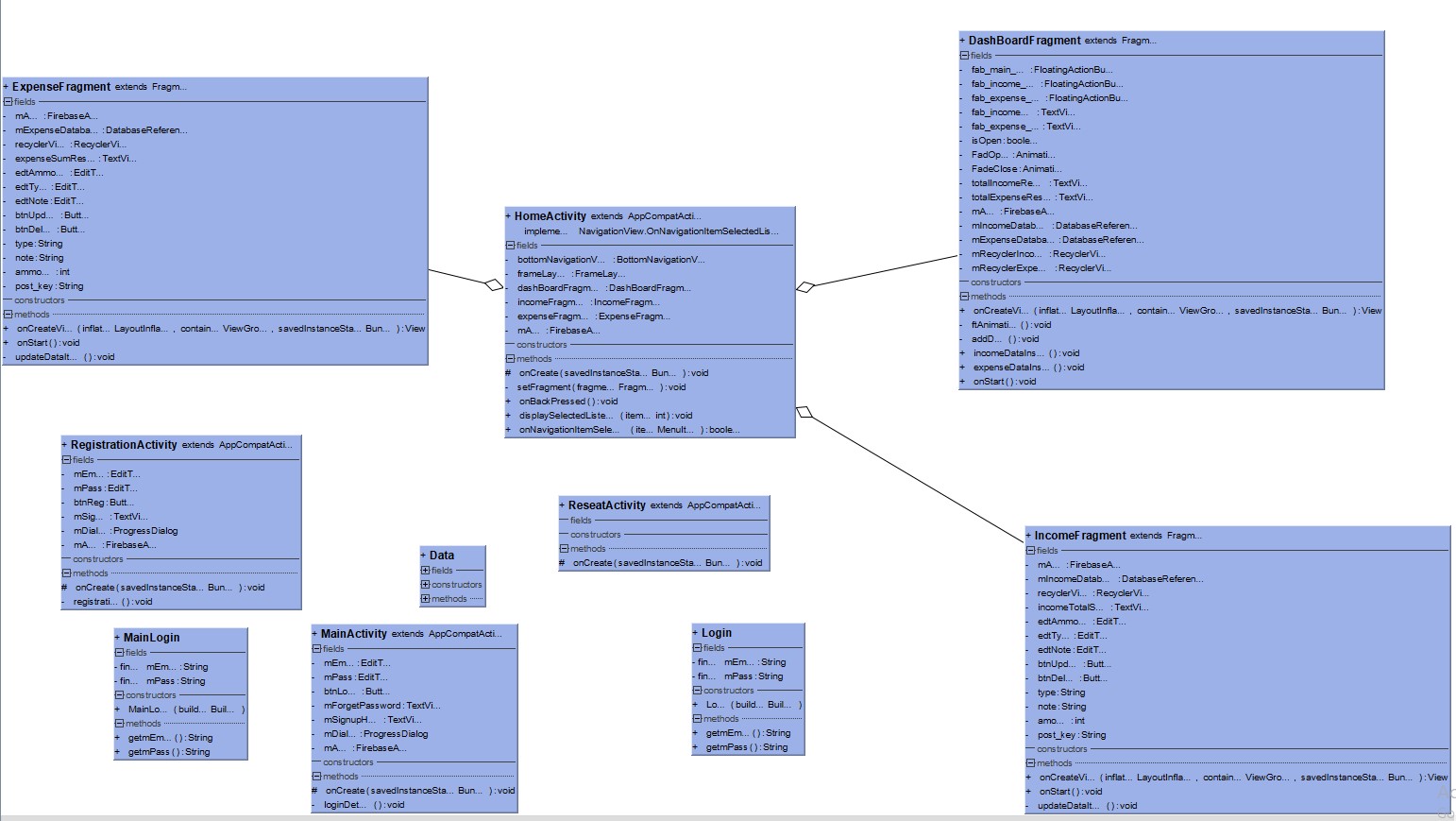


Για να μπορέσουμε να εισάγουμε την πληροφορία με τέτοιο τρόπο χρειάστηκε η δημιουργία καινούργιων κλάσεων σύμφωνα με τον παρακάτω τρόπο.



### 4.5.3.Διάγραμμα κλάσης

Στο παρακάτω διάγραμμα κλάσεων απεικονίζονται οι σχέσεις μεταξύ των κλάσεων στο Builder Pattern. Στο Builder Pattern δημιουργούμε τις κλάσεις Main Login και Login ώστε να διευκολύνουμε την διαδικασία αρχικοποίησης αντικειμένων τα οποία έχουν constructors με πάρα πολλά πεδία.



### 4.5.4 Οφέλη του Builder Pattern

* Σύνθετος Constructor. Πολλαπλός constructor ο οποίος έχει συνδυασμούς πολλαπλών παραμέτρων με εμφωλευμένα αντικείμενα.

* Μεγάλος αριθμός παραμέτρων. Έχοντας μεγάλο αριθμό παραμέτρων πεδίου είναι επίσης το βασικό σημείο που πρέπει να ληφθεί υπόψη.

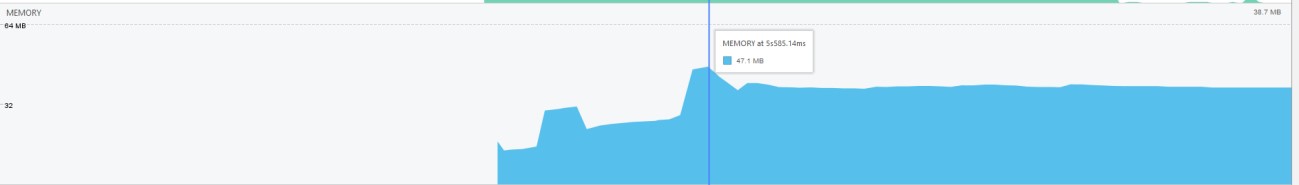
* Αμετάβλητότητα. Μπορούμε να αναγκάσουμε την αντικατάσταση του αντικειμένου μόλις τελειώσουμε με τη δημιουργία αντικειμένου

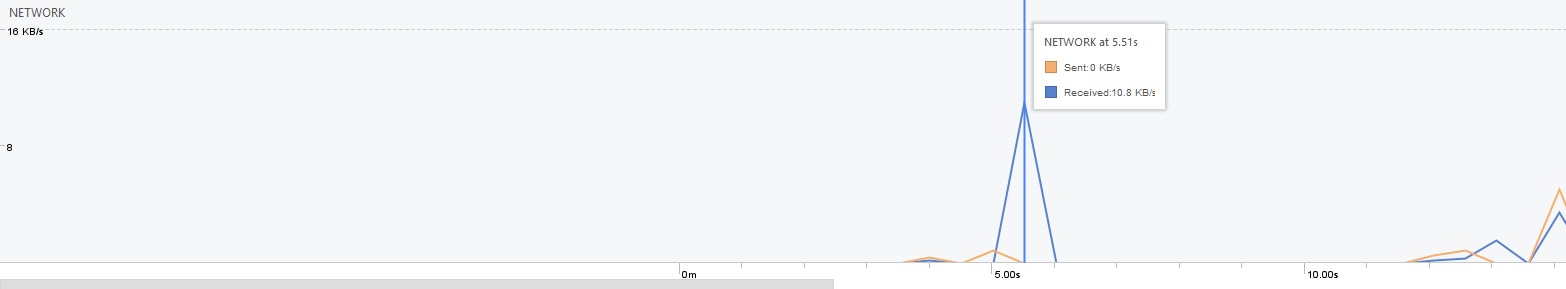
### 4.5.5.Profiling της εφαρμογής με την χρήση του Android Profiler.

Από τα παρακάτω screenshot του android profiler παίρνουμε κάποια στοιχεία για την χρήση της CPU,το δίκτυο και την μνήμη.

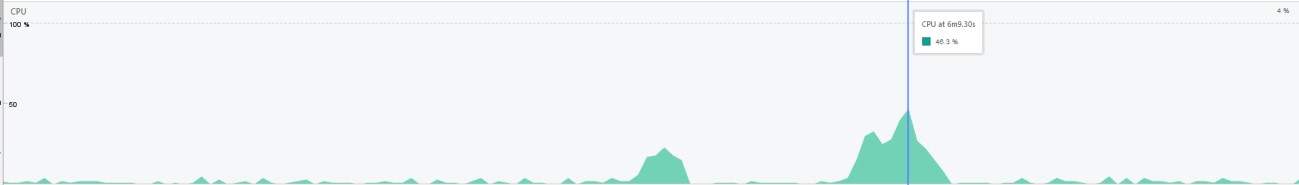
Για παράδειγμα στο από κάτω στιγμιότυπο έχουμε το Login στην εφαρμογή . **Login**

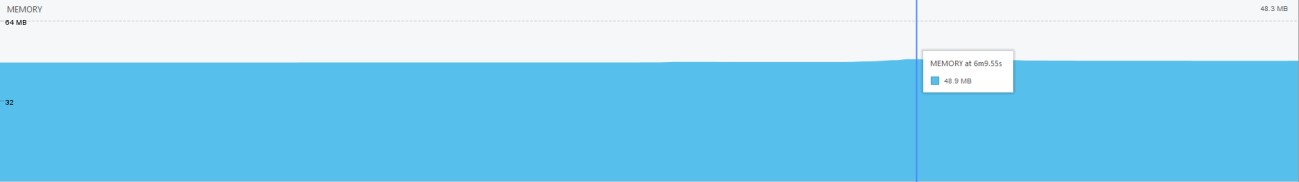






Στα παρακάτω screenshot βλέπουμε την επίδοση της εφαρμογής στις βασικές τις ενέργειες όπως **Update**







**Insert**

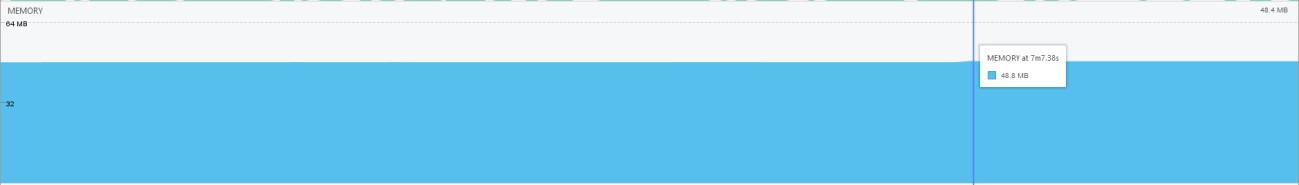


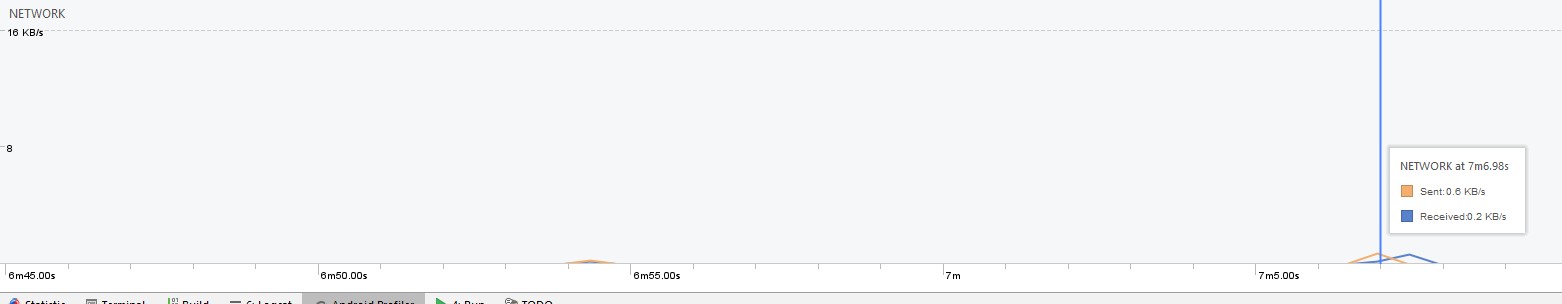




**Delete**







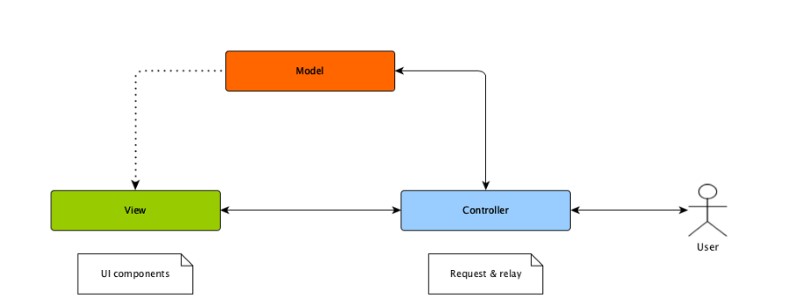
## 4.6 ΜΕΤΡΗΣΕΙΣ ΛΟΓΙΣΜΙΚΟΥ ΣΤΟ MVC PATTERN

#### 4.6.1 Εισαγωγή

Η ενότητα αυτή θα ασχοληθεί με τα software metrics του pattern MVC pattern στο Android.

Στην παρακάτω εικόνα μπορούμε να παρατηρήσουμε την λειτουργία του MVC pattern στο android δηλαδή το πως ορίζουμε το Model,το View και το Controller.

### 4.6.2.Πως λειτουργεί?



**Model**

Το Model αντιπροσωπεύει ένα σύνολο κατηγοριών που περιγράφουν την επιχειρησιακή λογική, δηλαδή το επιχειρηματικό μοντέλο καθώς και λειτουργίες πρόσβασης δεδομένων, δηλαδή το Μοντέλο δεδομένων. Ορίζει επίσης τους επιχειρηματικούς κανόνες για τα δεδομένα που αφορούν τον τρόπο αλλαγής και χειρισμού των δεδομένων.

**View**

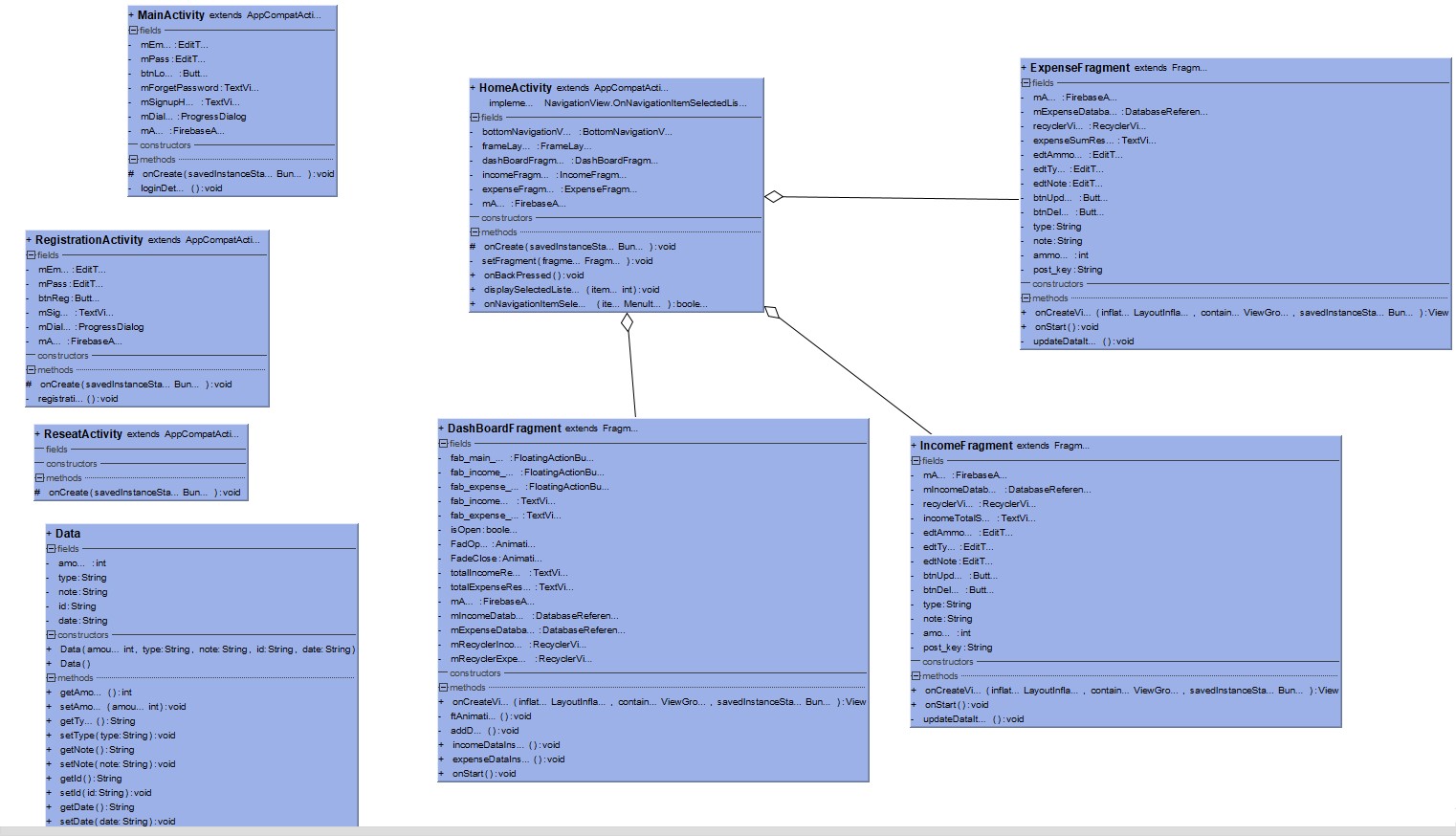
To View αντιπροσωπεύει τα στοιχεία UI. Είναι υπεύθυνο μόνο για την εμφάνιση των δεδομένων που λαμβάνονται από τον Controller ως αποτέλεσμα. Αυτό μετατρέπει επίσης τα μοντέλα σε UI.

**Controller**

Ο Controller είναι υπεύθυνος για την επεξεργασία των εισερχόμενων αιτημάτων. Λαμβάνει πληροφορίες από τους χρήστες μέσω του View, στη συνέχεια επεξεργάζεται τα δεδομένα του χρήστη με τη βοήθεια του Model και διαβιβάζει τα αποτελέσματα πίσω στο view. Συνήθως, ενεργεί ως controller μεταξύ του view και του model.

### 4.6.3.Διάγραμμα κλάσης

Στο παρακάτω διάγραμμα κλάσεων απεικονίζονται οι σχέσεις μεταξύ των κλάσεων στο MVC Pattern. Στο MVC Pattern οι κλάσεις διαχωρίζονται σε Model, View και Controller. Με αυτόν τον τρόπο επιτυγχάνεται διαχωρισμός των οντοτήτων αφού οι σχέσεις μεταξύ των κλάσεων μειώνονται δραματικά.



### 4.6.4 Οφέλη του MVC Pattern

* Ταχύτερη διαδικασία ανάπτυξης: το MVC υποστηρίζει ταχεία και παράλληλη ανάπτυξη. Με το MVC, ένας προγραμματιστής μπορεί να δουλέψει στο View, ενώ άλλοι μπορούν να εργαστούν στον Controller για να δημιουργήσουν την επιχειρηματική λογική της εφαρμογής. Η εφαρμογή που αναπτύχθηκε χρησιμοποιώντας MVC μπορεί να είναι τρεις φορές πιο γρήγορη από την εφαρμογή που αναπτύχθηκε χρησιμοποιώντας άλλα design patterns.

* Δυνατότητα παροχής πολλαπλών Views: Στο μοντέλο MVC, μπορείτε να δημιουργήσετε πολλαπλά Views για ένα μοντέλο. Η αλληλοεπικάλυψη κώδικα είναι πολύ περιορισμένη στο MVC επειδή διαχωρίζει δεδομένα και επιχειρησιακή λογική από την οθόνη.

* Υποστήριξη ασύγχρονης τεχνικής: Το MVC υποστηρίζει επίσης ασύγχρονη τεχνική, η οποία βοηθά τους προγραμματιστές να αναπτύξουν μια εφαρμογή η οποία φορτώνεται πολύ γρήγορα.

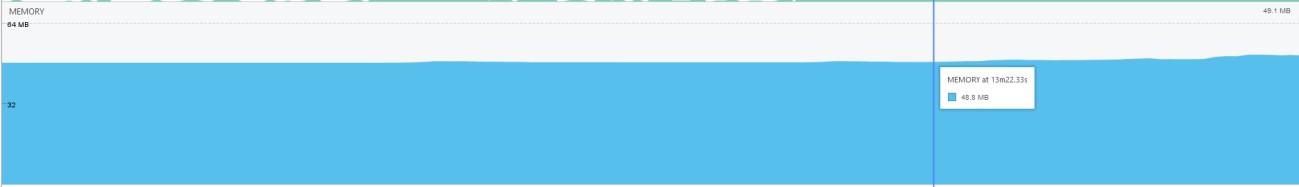
### 4.6.5.Android Profiler on Expense Manager

Από τα παρακάτω screenshot του android profiler παίρνουμε κάποια στοιχεία για την χρήση της CPU,το δίκτυο και την μνήμη.

Για παράδειγμα στο από κάτω στιγμιότυπο έχουμε το **Login** στην εφαρμογή .

Login

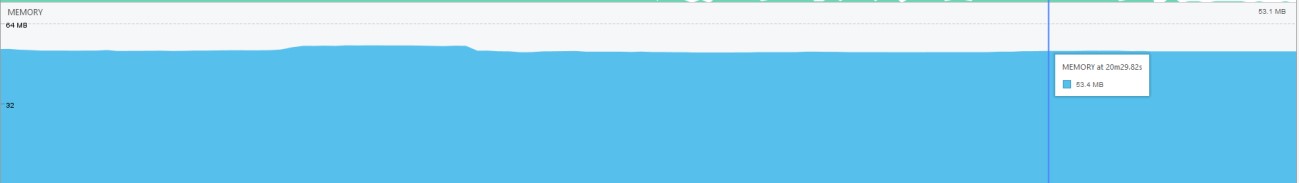


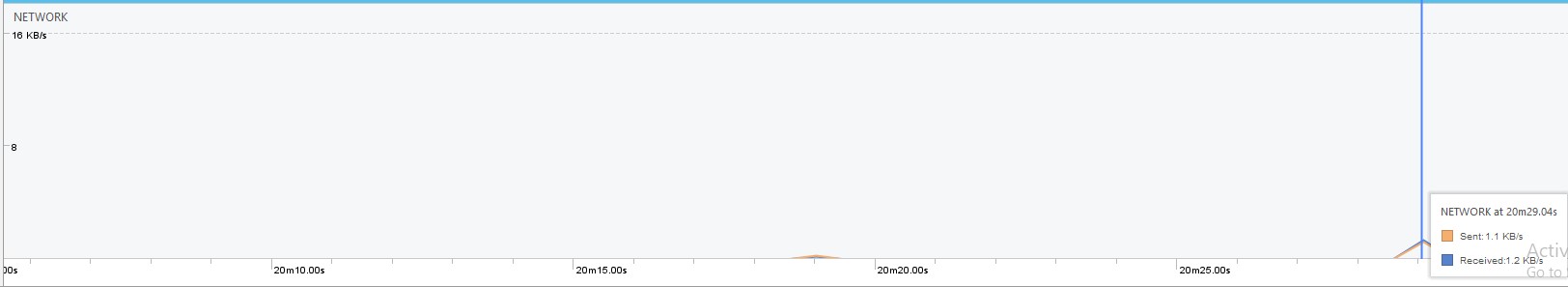




Update

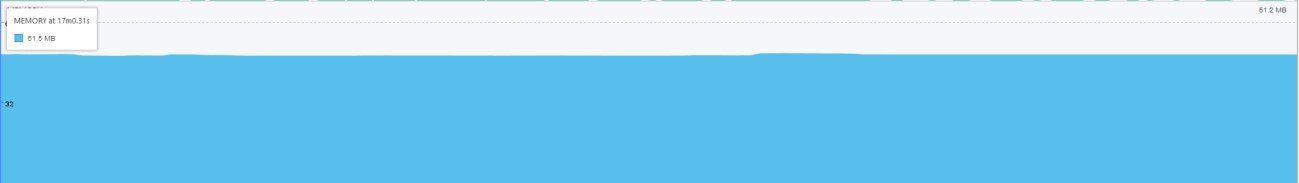


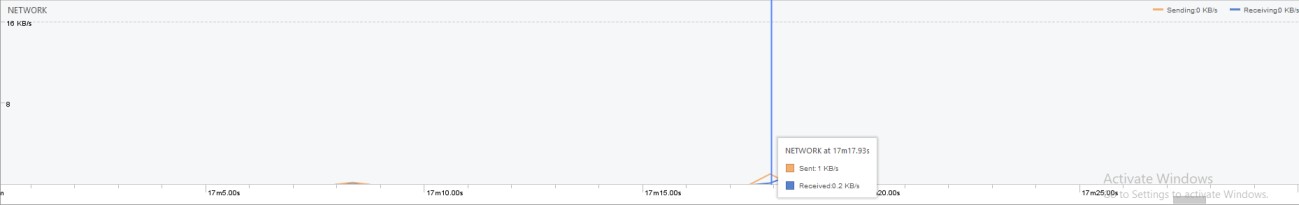




Insert



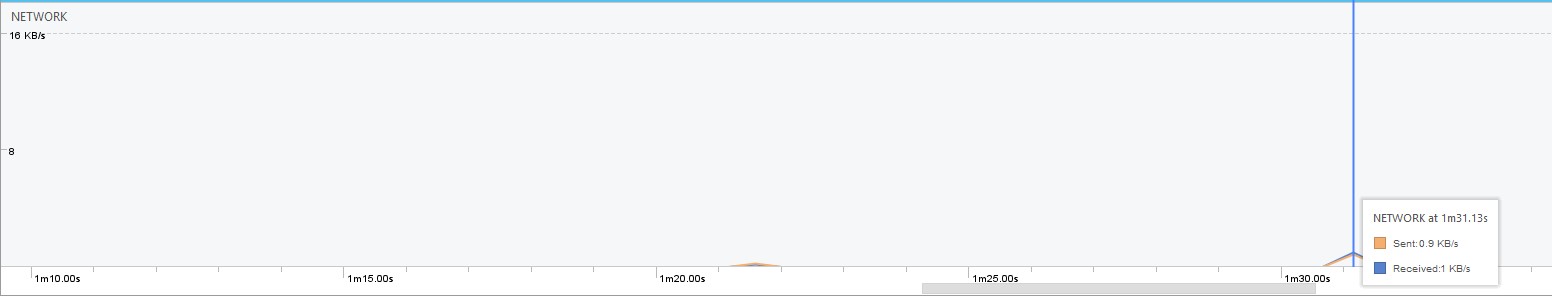




Delete

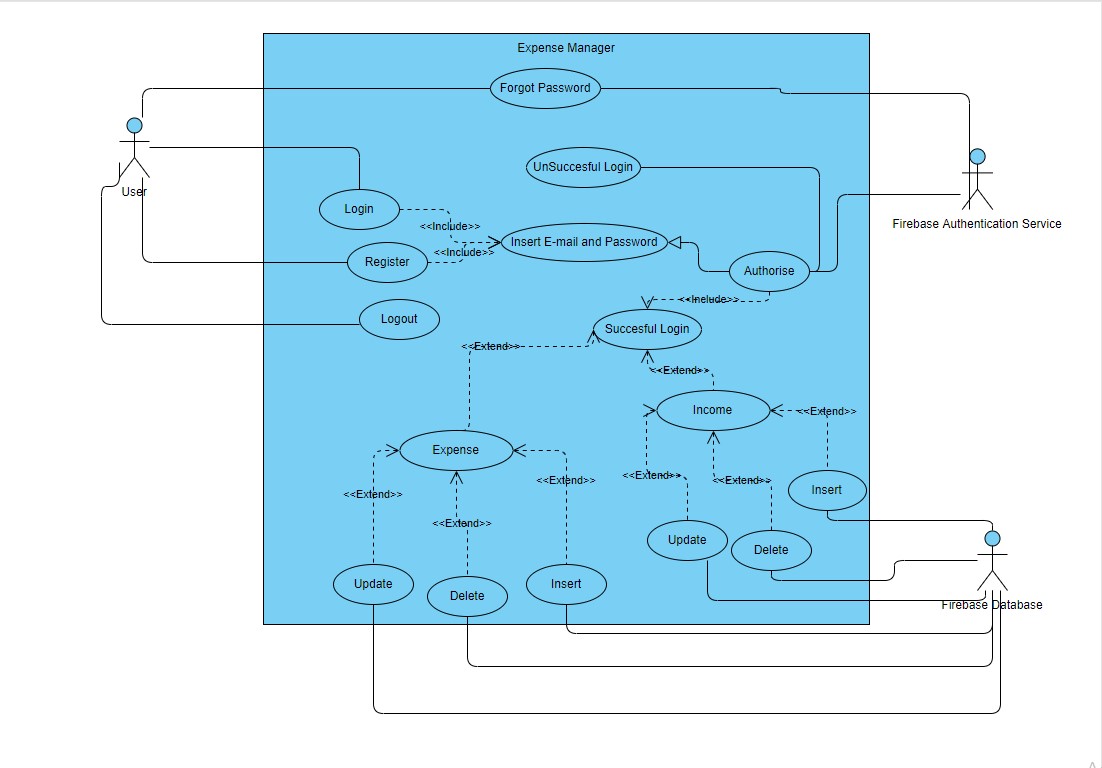






### 4.7 Ενέργειες χρήστη μέσα στην εφαρμογή

Στο παρακάτω Use case διαγραμμα μπορούμε να δούμε την αλληλεπίδραση των οντοτήτων μεταξύ τους αλλά και με το σύστημα.



Από το παραπάνω διάγραμμα παίρνουμε τις εξής πληροφορίες,

Ο χρήστης(**User**) μπορεί:

* Να κάνει **Login**
* Να κάνει **Register**
* Να κάνει **Logout**
* Να κάνει **Forgot Password**

Το σύστημα ταυτοποίησης(**Firebase Authentication Service**) μπορεί:

* Να κάνει έλεγχο στοιχείων όταν κάποιος χρήστης προσπαθεί να κάνει **Login**
* Να κάνει έλεγχο στοιχείων όταν κάποιος χρήστης προσπαθεί να κάνει **Register**
* Να κάνει έλεγχο για την επανέκδοση του κωδικού από το αίτημα του χρήστη
* Να δώσει πρόσβαση στην εφαρμογή όταν ο κωδικός είναι έγκυρος
* Να εμφανίσει μήνυμα ανεπιτυχούς εισόδου όταν ο κωδικός δεν είναι έγκυρος

Εφόσον ο χρήστης ταυτοποιηθεί από το σύστημα μπορεί:

* Να κάνει **insert** νέου εσόδου/ εξόδου
* Να κάνει **delete** ενός έσοδο/ εξόδου
* Να κάνει **update** ένα έσοδο/ εξόδου
* Να κάνει **Logout**

Η αντίστοιχη βάση δεδομένων της εφαρμογής(**Firebase Database**) μπορεί:

* Να κάνει **insert** νέου εσόδου/ εξόδου
* Να κάνει **delete** ενός έσοδο/ εξόδου
* Να κάνει **update** ένα έσοδο/ εξόδου

### 4.8 Στατιστικα στοιχεια των Design Pattern και συγκριση

##### 4.8.1.Στατιστικά στοιχεία

**CPU:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PATTERN** | **LOGIN** | **UPDATE** | **INSERT** | **DELETE** |
| *MVC* | 62.3% | 49.6% | 36.1% | 45.3% |
| *MVP* | 61.4% | 34.8% | 44% | 45.3% |
| *BUILDER* | 70.2% | 46.3% | 37.9% | 33% |
| *DEPENDENCY*  *INJECTION* | 62.6% | 47% | 41.5% | 32.7% |
| *PROXY* | 64.3% | 49.9% | 43.9% | 41.6% |
| *TEMPLATE* | 57.3% | 40% | 49.8% | 45.4% |

**MEMORY:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PATTERN** | **LOGIN** | **UPDATE** | **INSERT** | **DELETE** |
| *MVC* | 48.8 Mb | 53.4 Mb | 51.5 Mb | 47.1 Mb |
| *MVP* | 52.9 Mb | 55.1 Mb | 54 Mb | 56.7 Mb |
| *BUILDER* | 47.1 Mb | 48.9 Mb | 45.1 Mb | 48.8 Mb |
| *DEPENDENCY*  *INJECTION* | 52.8 Mb | 54.5 Mb | 52 Mb | 54.3 Mb |
| *PROXY* | 52.6 Mb | 55 Mb | 51.3 Mb | 56 Mb |
| *TEMPLATE* | 48.6 Mb | 51.2 Mb | 49.4 Mb | 54.4 Mb |

**NETWORK:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PATTERN** | **LOGIN** | **UPDATE** | **INSERT** | **DELETE** |
| *MVC* | **SENT:**4.2  Kb/s  **RECEIVED:**  2.7 Kb/s | **SENT:**1.1  Kb/s  **RECEIVED:**  1.2 Kb/s | **SENT:**1Kb/s **RECEIVED:**  0.2 Kb/s | **SENT:**0.9Kb/s **RECEIVED:** 1 Kb/s |
| *MVP* | **SENT:**4 Kb/s **RECEIVED:**  2.5 Kb/s | **SENT:**1 Kb/s **RECEIVED:**  0.2 Kb/s | **SENT:**1.2  Kb/s  **RECEIVED:**  1.3 Kb/s | **SENT:**0.8Kb/s **RECEIVED:**  0.9 Kb/s |
| *BUILDER* | **SENT:**0Kb/s **RECEIVED:**  10.8 Kb/s | **SENT:**1.1  Kb/s  **RECEIVED:**  0.1 Kb/s | **SENT:**1.1  Kb/s  **RECEIVED:**  0.2 Kb/s | **SENT:**0.6Kb/s **RECEIVED:**  0.2 Kb/s |
| *DEPENDENCY*  *INJECTION* | **SENT:**0.8  Kb/s  **RECEIVED:**  10.8 Kb/s | **SENT:**0  Kb/s  **RECEIVED:** 1 Kb/s | **SENT:**1.2  Kb/s  **RECEIVED:**  1.3 Kb/s | **SENT:**0.8  Kb/s  **RECEIVED:** 1 Kb/s |
| *PROXY* | **SENT:**1.5  Kb/s  **RECEIVED:**  11.1 Kb/s | **SENT:**1.2  Kb/s  **RECEIVED:**  1.3 Kb/s | **SENT:**1.2  Kb/s  **RECEIVED:**  1.3 Kb/s | **SENT:**0.8 Kb/s **RECEIVED:**  0.9 Kb/s |
| *TEMPLATE* | **SENT:**3.7  Kb/s  **RECEIVED:**  0.1 Kb/s | **SENT:**1.2  Kb/s  **RECEIVED:**  0.6 Kb/s | **SENT:**0.1  Kb/s  **RECEIVED:** 1 Kb/s | **SENT:**0  Kb/s  **RECEIVED:**  0.7 Kb/s |

###### 4.8.2. Σύγκριση των Design Pattern

Από τους παραπανω πινακες παιρνουμε τις μετρησεις των Design Pattern για τις 4 βασικες ενεργειες της εφαρμογης οι οποιες είναι Login, Update, Delete, Insert.Με βάση αυτά τα στοιχεία μπορούμε να συγκρίνουμε τα pattern ώστε να διεξάγουμε συμπεράσματα.

‘Οσο αναφορά την CPU για βα βρούμε το πιο αποδοτικό pattern θα αθροίσουμε τον πινακα CPU κατά γραμμή.Αυτο μας δίνει ως αποτέλεσμα τον παρακάτω πίνακα ***Αθροισμα κατά γραμμή στον πίνακα CPU:***

|  |  |
| --- | --- |
| ***Dependency Injection*** | *183,8* |
| ***MVP*** | *185,5* |
| ***Builder*** | *187,4* |
| ***Template*** | *192,5* |
| ***MVC*** | *193,3* |
| ***Proxy*** | *199,7* |

Από τον παραπάνω πίνακα φαίνονται τα πιο απόδοτικά Pattern ταξινομημένα κατά αύξουσα σειρά.Με την ίδια λογική θα βγεί και ο πίνακας αθροίσματος κατά γραμμή για τον πίνακα Memory.

***Αθροισμα κατά γραμμή στον πίνακα Memory:***

|  |  |
| --- | --- |
| ***Builder*** | *189,9* |
| ***MVC*** | *200,8* |
| ***Template*** | *203,6* |
| ***Dependency Injection*** | *213,6* |
| ***Proxy*** | *214,9* |
| ***MVP*** | *218.7* |

‘Οσο αναφορα την διαχείρηση της μνήμης ο παραπάνω πίνακας μας δίνει τα πιο απόδοτικά pattern ταξινομημένα επίσης κατά αύξουσα σειρά.Τελος αθροίζοντας κατά γραμμή τον πίνακα Network έχουμε τον εξής πίνακα

***Αθροισμα κατά γραμμή στον πίνακα Network:***

|  |  |
| --- | --- |
| ***Template*** | *Sent: 5*  *Received: 2.4* |
| ***MVP*** | *Sent:7*  *Received:4.9* |
| ***MVC*** | *Sent:7.2*  *Received:5.1* |
| ***Builder*** | *Sent:2,8*  *Received:11.3* |
| ***Dependency Injection*** | *Sent:2.8*  *Received:14.1* |
| ***Proxy*** | *Sent:4.7*  *Received:14.6* |

‘Οσο αναφορα την διαχείρηση των πόρων του δικτύου ο παραπάνω πίνακας μας δίνει τα πιο απόδοτικά pattern ταξινομημένα κατά αύξουσα σειρά.

Επίσης σημαντικό είναι να αναφέρουμε τον χρόνο που χρειάζεται το κάθε pattern να κάνει build αλλα και τις γραμμες κώδικα.Απο τον παρακάτω πίνακα παίρνουμε αυτά τα στοιχεία.

|  |  |  |
| --- | --- | --- |
| ***Pattern*** | ***Build Time*** | ***Lines Of Code*** |
| *MVP* | 17s 116ms | 23967 |
| *MVP* | 13s 59ms | 23311 |
| *Builder* | 12s 493ms | 23963 |
| *Proxy* | 12s 59ms | 21143 |
| *Dependency Injection* | 16s 892ms | 23913 |
| *Template* | 12s 435ms | 23948 |

Από τους παραπάνω πίνακες βγάζουμε τα εξής συμπεράσματα.

Καλύτερο Design Pattern για την CPU:***Dependency Injection,***

Καλύτερο Design Pattern για την Μνήμη(Memory):***Builder,***

Καλύτερο Design Pattern για την διαχείρηση πόρων δικτύου(Network):***Template,***

Καλύτερο Build Time:***Proxy,***

Λιγότερες Γραμμές Κώδικα: ***Proxy.***

Αντιθέτως έχουμε,

Χειρότερο Design Pattern για την CPU:***Proxy,***

Χειρότερο Design Pattern για την Μνήμη(Memory):***MVP,***

Χειρότερο Design Pattern για την διαχείρηση πόρων δικτύου(Network):***Proxy,***

Χείριστο Build Time:***MVP,***

Περισσότερες Γραμμές Κώδικα: ***MVP.***

Τα παραπάνω στοιχεία και αριθμοί μας δίνουν μία πολύ καλή εικόνα για τις μετρήσεις του λογισμικόυ σε περιβάλλον Android με χρήση Firebase.Για παράδειγμα, αμα κάποιος προγραμματιστής χρείαζεται να κατασκευάσει πολύ αποδοτικό πρόγραμμα σχετικά με την μνήμη θα επιλέξει το Builder Pattern και αντίστοιχα για τα υπόλοιπα (CPU,Network,κλπ.).Όμως η χρήση pattern μπορεί να μην έχει να κάνει πάντα με την αποδοτικότητα του προγράμματος.Ο διαχωρισμός των **layers** που γίνεται στο **MVP** Pattern χρησιμοποιείται για καλύτερο **testing.**Η διαδικασία που εκτελείται στο **Proxy** Pattern γίνεται για λόγους ασφαλείας κυρίως και σε κάποιες περιπτώσεις και για λόγους αποδοτικότητας.Τα Τemplates που κατασκευάζουμε στο Template Pattern, χρησιμοποιούνται ώστε να γράφουμε επαναχρησιμοποιούμενο κώδικα.Τα **dependencies** που χρησιμοποιούμε στο Dependency Injection pattern παρέχονται στο πρόγραμμα και γίνονται **Inject** στις κλάσεις που τα χρειάζονται.Επίσης, βοηθούν στον διαχωρισμό των layers, στην εύκολη ανάπτυξη του προγράμματος αλλα επίσης και στην εύκολη αρχικοποίηση των αντικειμένων(object).Επομένως μπορούμε μόνο να ορίσουμε για συγκεκριμένες σκοπιές ποια pattern είναι τα καλύτερα.

# 5.Εργαλεια και Τεχνολογίες

### 5.1 Γλώσσα Προγραμματισμού

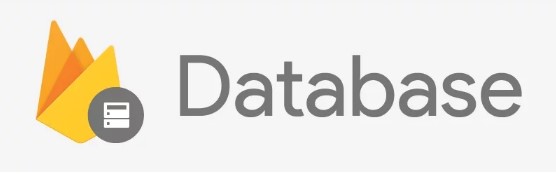
Η **γλώσσα προγραμματισμού** που χρησιμοποιήθηκε για την ανάπτυξη αυτού του Project είναι η **Java Standard Edition 8**. Αυτή η γλώσσα παρέχεται από την **Oracle**. Η έκδοση αυτή προσφέρει πολλές λειτουργίες οι οποίες ήταν χρήσιμες για την διεκπεραίωση αυτής της εργασίας. Περισσότερες πληροφορίες για αυτήν την έκδοση μπορούν να βρεθούν στο παρακάτω link:

<https://www.oracle.com/technetwork/java/javase/8-whats-new-2157071.html>



### 5.2 Βάση Δεδομένων

Η βάση δεδομένων που χρησιμοποιήθηκε για την ανάπτυξη αυτού του Project είναι το **Firebase Database**. Το προιόν διατίθεται από την **Firebase,Inc** η οποία είναι μία θυγατρική της **Google.** Αυτή η βάση δεδομένων προσφέρει μία πολύ γρήγορη και εύκολη λύση για την εφαρμογή Android που υλοποιήθηκε.Η έκδοση που χρησιμοποιήθηκε στην εφαρμογή είναι η **10.0.1.**



### 5.3 Περιβάλλον

Το περιβάλλον που χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής είναι το **Android Studio**.To Android Studio έιναι ένα λογισμικό βασισμένο στο **JetBrains** και το **IntelliJ IDEA**. Το λογισμικό αυτό παρέχεται από την **Google** και είναι αποκλειστικά για προγραμματισμό σε **Android**. Η έκδοση που χρησιμοποιήθηκε για την υλοποίηση αυτού του Project είναι η **3.2.1.**



### 5.4 Εφαρμογή για Μετρήσεις

Το λογισμικό που χρησιμοποιήθηκε για τις μετρήσεις του λογισμικού είναι μίια εφαρμογή του Android Studio η οποία ονομάζεται **Android Profiler.** Η εφαρμογή αυτή καταγράφει τις επιδόσεις της CPU(επεξεργαστής),της MEMORY(μνήμη) και του NETWORK(δικτύου).Το Android Profiler είναι συμβατό με Android 5.0(API Level 21) και μεγαλύτερο.

### 5.5 Plugins

Τα plugins που χρησιμοποιήθηκαν στην συγκεκριμένη εφαρμογή είναι τα παρακάτω:

* SimpleUML για την δημιουργια των class diagrams.
* LucidChart για την δημιουργία του Use Case Diagram.
* Dagger2 Framework το οποίο χρησιμοποιήθηκε στο Dependency Injection pattern ώστε να παρέχουμε τα dependencies.
* Firebase-core version 10.0.1 για να εισάγουμε της βασικές λειτουργίες του Firebase Database
* Firebase-client-android version 2.5.2
* Firebase-ui-database version 0.4.0
* Statistic είναι ένα plugin που μας εμφάνισε κάποια στατιστικά στοιχεία όπως οι γραμμές κώδικα όλου του προγράμματος.

# 6.Συμπεράσματα και Μελλοντικές επεκτάσεις

Στην συγκεκριμένη εργασία αναπτύχθηκε μία εφαρμογή καταγραφής εξόδων σε έξι διαφορετικά pattern ώστε να κάνουμε καταγραφή της αποδοτικότητας του κάθε pattern.Η συγκεκριμένη εφαρμογή δίνει την δυνατότητα στους χρήστες της να καταγράφουν τα έξοδα τους με σημειώσεις ώστε να γνωρίζουν και να βλέπουν την δραστηριότητα τον εξόδων τους.Βασικό στοιχείο αυτής της εργασίας ήταν η μεγάλη εξοικείωση με τους τρόπους ανάπτυξης και δομής μίας Android εφαρμογής.

Μέσα από την ανάπτυξη της παραπάνω εφαρμογής αποκτήθηκαν αρκετές γνώσεις. Συγκεκριμένα, έγινε εξερεύνηση στην πλατφόρμα του Android με παράλληλη εξερεύνηση των επιπλέων εφαρμογών και λειτουργιών που διαθέτει.Επίσης έγινε μεγάλη κατανόηση της αντικειμενοστραφούς γλώσσας προγραμματισμού Java αφού ήταν η γλώσσα υλοποίησης της εφαρμογής μας.Επιπλέον, υπήρξε μεγάλη εντριφή με αποδοτικούς τρόπους υλοποίησης προγραμμάτων χρησιμοποιώντας κληρονομικότητα,interfaces,abstract classes αλλα και στο επίπεδο του **ui** με τα fragments και τα activities.Παρόλες τις δυσκολίες που εμφανίστηκαν η εργασία υλοποιήθηκε με μεγάλη επιτυχία καθώς έδωσε καθοριστικά αποτελέσματα διότι, πήραμε μετρήσεις σε μία εφαρμογή η οποία δομήθηκε με 6 διαφορετικά pattern.

Όσο αναφορά τις μελλοντικές επεκτάσεις της εφαρμογής είναι γεγονός ότι υπάρχουν περιθώρια βελτίωσης με κάποιες επιπρόσθετες λειτουργίες.Κάποια από αυτά είναι η προσθήκη κοινών λογαριασμών με άλλους χρήστες όπως γίνεται στην εφαρμογή **Splitwise.**Tέλος, θα ήταν πολύ χρήσιμο η εφαρμογή να υποστηρίζει ξένες γλώσσες ώστε να στοχεύει σε χρήστες εκτός Ελλάδας.

# 7. Βιβλιογραφία

**Βιβλία:**

1. Eric Freeman,Elisabeth Robson,Bert Bates,Kathy Sierra, *“Head First Design Patterns”,1st Edition,* O’ Reilly, Οκτώβριος 2004
2. James W. Cooper,Allison Wesley, *“Java Design Patterns: A tutorial”,*Ιανουάριος 2008

**Links:**

1. “Java-Design-Patterns”, Github,<https://github.com/iluwatar/java-design-patterns>

[4]”Using Dagger2 for dependency Injection in Android-Tutorial”,vogella, <http://www.vogella.com/tutorials/Dagger/article.html>

[5]”Android”,<https://www.android.com/>

[6]”Firebase”,<https://firebase.google.com/>

**‘Αρθρα :**

[7]Rakshit Soral,Architectural Guidelines to follow for MVP pattern in

Android,Μάιος 2018, [https://android.jlelse.eu/architectural-guidelines-to-follow-formvp-pattern-in-android-2374848a0157](https://android.jlelse.eu/architectural-guidelines-to-follow-for-mvp-pattern-in-android-2374848a0157)

[8]Nitin Agarwal,Android MVP for Beginners,Απρίλιος 2017, <https://android.jlelse.eu/android-mvp-for-beginners-25889c500443>

[9]Thanos Karpouzis,Android Architecture,Άυγουστος 2015, <https://android.jlelse.eu/android-architecture-2f12e1c7d4db>

1. Bipin Pandey,Model View Presenter(MVP) in Android with a simple Demo Project,Δεκέμβριος 2017, [https://medium.com/cr8resume/make-you-hand-dirtywith-mvp-model-view-presenter-eab5b5c16e42](https://medium.com/cr8resume/make-you-hand-dirty-with-mvp-model-view-presenter-eab5b5c16e42)

1. Florina Muntunescu, Android Architecture Patterns Part 1:

Model-View-Controller,Νοέμβριος 2016, [https://medium.com/upday-devs/androidarchitecture-patterns-part-1-model-view-controller-3baecef5f2b6](https://medium.com/upday-devs/android-architecture-patterns-part-1-model-view-controller-3baecef5f2b6)

[12]Paulina Szklarska, Design Patterns in Android- Builder,Άυγουστος 2017, <https://www.thedroidsonroids.com/blog/design-patterns-in-android-builder>

[13]Ankit Sinhal, Create complex object at run-time with Builder pattern,Μάιος 2017, https://android.jlelse.eu/create-complex-object-at-run-time-with-builderpattern-d425e6f4408e

[14]Ankit Sinhal,Fundamentals of Dependency Injection and popular libraries on Android,Άυγουστος 2017, [https://android.jlelse.eu/fundamentals-of-dependencyinjection-and-popular-libraries-in-android-c17cf48b5253](https://android.jlelse.eu/fundamentals-of-dependency-injection-and-popular-libraries-in-android-c17cf48b5253)

[15]James Surgue,Proxy Pattern Tutorial with Java Examples,Μάρτιος 2010, <https://dzone.com/articles/design-patterns-proxy>

[16]James Surgue,Template Method Pattern Tutorial with Java Examples,Απρίλιος 2010,<https://dzone.com/articles/design-patterns-template-method>

# 8.Παράρτημα

### 8.1 Γλωσσάριο

Ο παρακάτω πίνακας δίνει τις μεταφράσεις όρων που ήταν στα αγγλικά μεταφρασμένους στα ελληνικά.

|  |  |
| --- | --- |
| **Αγγλικά** | **Ελληνικά** |
| Design Pattern | Σχεδιαστικό μοτίβο |
| Login | Είσοδος |
| Update | Εκσυγχρόνιση |
| Insert | Εισάγω |
| Delete | Διαγράφω |
| Email | Ηλεκτρονικό Ταχυδρομείο |
| Password | Κωδικός |
| Reset | Επαναφορά |
| Register | Εγγραφή |
| Home | Αρχική Σελίδα |
| Εxpense | Έξοδο |
| Income | Έσοδο |
| Amount | Ποσό |
| Type | Τύπος |
| Note | Σημείωση |
| Framework | Πλαίσιο Εργασίας |
| Object | Αντικείμενο |
| injectors | Διοχετευτής |
| Annotation Processor | Επεξεργαστής Συμβόλων |
| Model | Μοντέλο |
| View | Προβολή |
| Presenter | Παρουσιάζων |
| Testing | Δοκιμάζω |
| Interface | Διεπαφή |
| Activity | Δραστηριότητα |
| Layers | Επίπεδα |
| Behavioral | Συμπεριφορικό |
| Abstract | Αφηρημένο |
| Override | Καταπάτηση |
| Fragment | Τεμάχιο |
| Textbox | Πλαίσιο Κειμένου |
| Constructors | Κατασκευαστής |
| CPU | Επεξεργαστής |
| Software Metrics | Μετρήσεις Λογισμικού |
| UI | Διεπαφή χρήστη |
| Sent | Απεσταλμένα |
| Received | Έλαβα |
| Build Time | Χρόνος Κατασκευής |