ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

Πολυτεχνική Σχολή

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ Η/Υ

Εφαρμογή διαχείρισης οικονομικών για συσκευές Android

Finance managing application for Android devices

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΚΑΡΑΓΙΑΝΝΗΣ ΔΗΜΗΤΡΙΟΣ

Επιβλέποντες Καθηγητές: Αλκιβιάδης Ακρίτας, Καθηγητής
Γεώργιος Σταμούλης, Καθηγητής

Βόλος, 2015

Περίληψη

Είναι ευρέως διαδεδομένο πως η τεχνολογία έχει εισέλθει σε πολύ μεγάλο βαθμό στην καθημερινή ζωή των ανθρώπων δίνοντάς τους την δυνατότητα να ικανοποιούν ανάγκες χωρίς κόπο και σε μικρό χρονικό διάστημα. Σε αυτό έχει συμβάλλει εξαιρετικά η ραγδαία ανάπτυξη των κινητών τηλεφώνων τα οποία επιτρέπουν στους χρήστες τους να έχουν έναν ηλεκτρονικό υπολογιστή οπουδήποτε και αν βρίσκονται.

Στην παρούσα διπλωματική εργασία θα γίνει αναλυτική περιγραφή και οπτική παρουσίαση μιας εφαρμογής διαχείρισης οικονομικών που θα απευθύνεται σε χρήστες στην κατοχή των οποίων υπάρχουν συσκευές Android.

Abstract

It is well known that technology has entered heavily in the daily lives of people by giving them the ability to satisfy needs without effort and in a short time. The rapid development of mobile phones has contributed greatly to this spreading allowing the users to carry a pocket-size computer everywhere they go.

This thesis will be a detailed description and visual presentation of a financial management application targeted at users in possession of which are Android devices.

Ευχαριστίες

Με την περάτωση της παρούσας διπλωματικής εργασίας θα ήθελα να ευχαριστήσω τον επιβλέποντα Καθηγητή **κ. Ακρίτα Αλκιβιάδη** για την καθοδήγηση, τις παρατηρήσεις του καθώς και για την εμπιστοσύνη που έδειξε στο πρόσωπό μου καθ' όλη την διάρκεια εκπόνησής της.

Ένα μεγάλο ευχαριστώ από καρδιάς, επίσης, στην οικογένειά μου και στους φίλους μου που με στήριξαν και με ενθάρρυναν σε κάθε μου εγχείρημα όλα αυτά τα χρόνια καθώς χωρίς αυτούς δεν θα ήταν δυνατό να επιτευχθεί οτιδήποτε.

1 ΠΕΡΙΕΧΟΜΕΝΑ

Κεφάλαιο 1

1 MoneyGer

1.1 Περιγραφη της Εφαρμογης

Η εφαρμογή που θα περιγράψουμε και η οποία φέρει το όνομα **MoneyGer** (ως σύμπτυξη των λέξεων Money και Manager) αποτελεί μία **Android εφαρμογή διαχείρισης οικονομικών**. Παρέχει μηχανισμούς καταγραφής εσόδων και εξόδων, ενημέρωσης υπολοίπου, προβολή στατιστικών σχετικά με τις πραγματοποιημένες συναλλαγές, διαχείριση ιστορικού αποθηκευμένων συναλλαγών (διαγραφή ανεπιθύμητων ή λανθασμένων έπειτα από επιβεβαιωμένη είσοδο με κωδικό). Δημιουργήθηκε εξ΄ολοκλήρου στο περιβάλλον του Android Studio και απευθύνεται σε συσκευές με έκδοση Android API μεγαλύτερη ή ίση του 11 οι οποίες και καλύπτουν πλέον το 96% των συσκευών παγκοσμίως.

Καταβλήθηκε ιδιαίτερη προσπάθεια ώστε να γίνει όσο το δυνατόν πιο εύχρηστη και για τον λόγο αυτό όλες σχεδόν οι ενέργειες που εκτελούνται από τον χρήστη γίνονται με το «πάτημα ενός κουμπιού». Σχεδιάστηκαν έξυπνα εικονίδια τα οποία αντικαθιστούν οποιοδήποτε κείμενο κάνοντας την εφαρμογή πιο αποδοτική τόσο αισθητικά όσο και λειτουργικά. Η μετάφραση, τέλος, της εφαρμογής στα ελληνικά επιβεβαιώνει την προσπάθεια αυτή.

1.2 Εργαλεια Που Χρησιμοποιηθηκαν

Όπως προαναφέρθηκε, η εφαρμογή αναπτύχθηκε αποκλειστικά στο περιβάλλον που προσφέρει το Android Studio, το επίσημο IDE που παρέχεται για την ανάπτυξη εφαρμογών Android και το οποίο είναι βασισμένο στο IntelliJ IDEA. Η επιλογή του (πολλοί προγραμματιστές εφαρμογών Android εξακολουθούν να χρησιμοποιούν το περιβάλλον του Eclipse) έγινε βάσει της πληθώρας των δυνατοτήτων που προσφέρει, ορισμένες από τις οποίες είναι:

- Built-in σύστημα Building και Compiling του κώδικα
- Ενσωματωμένες λειτουργίες ελέγχου χρήσης μνήμης και CPU
- Εύκολη ενσωμάτωση emulators για την δοκιμή της εφαρμογής σε ασφαλές περιβάλλον
- Λειτουργία drag and drop αρχείων για άνοιγμα κατευθείαν από τον Explorer
- Δημιουργία του τελικού apk αρχείου της εφαρμογής το οποίο θα είναι ψηφιακά υπογεγραμμένο από τον προγραμματιστή μέσω του app signing που παρέχεται
- Εξαιρετικά φιλικό προς τον προγραμματιστή περιβάλλον

Οι δοκιμές του κώδικα της εφαρμογής έλαβαν χώρα σε δύο διαφορετικά σημεία. Χρησιμοποιήθηκε τόσο το Genymotion όσο και φυσικές συσκευές. Το **Genymotion** είναι ένας εξομοιωτής για το λειτουργικό σύστημα του Android που προσφέρει ένα εικονικό περιβάλλον για αναπτυξιακούς σκοπούς. Η επιλογή του δεν έγινε τυχαία, καθώς αποτελεί έναν από τους πιο γρήγορους και φιλικούς προς τον χρήστη/προγραμματιστή emulators ο οποίος περιλαμβάνει πληθώρα διαφορετικών εκδόσεων του λειτουργικού ώστε να δοκιμαστεί η εφαρμογή σε όσο το δυνατόν περισσότερες εικονικές συσκευές. Η εφαρμογή, εντούτοις, δοκιμάστηκε και σε φυσικές συσκευές και πιο συγκεκριμένα σε:

- ZTE BLADE Mini Q με έκδοση Android 4.2.2
- Huawei Ascend G620s με έκδοση Android 4.4.4
- Samsung Galaxy S2 GT-I9100 με έκδοση Android 4.1.2
- Lenovo A6000 με έκδοση Android 5.0

1.3 Δομη Εργασιας

Η κύρια δομή που θα ακολουθηθεί στην περιγραφή της εφαρμογής θα είναι η εξής: αρχικά θα γίνει μία ανάλυση των κλάσεων που την αποτελούν καθώς και των λειτουργιών που η κάθε μία υλοποιεί ενώ στην συνέχεια θα παρατεθεί και ο συνολικός κώδικας. Για την καλύτερη κατανόηση του τρόπου υλοποίησης-πέρα από τον συνολικό κώδικα στο τέλος- έχουν εισαχθεί αξιοσημείωτα μέρη του κώδικα και κατά την εξέταση των κλάσεων πάνω στα οποία λαμβάνει χώρα διεξοδική ανάλυση. Έχουν χρησιμοποιηθεί, τέλος, διάφορα στιγμιότυπα από την δοκιμή της εφαρμογής ώστε ο αναγνώστης να έχει την δυνατότητα οπτικής εξέτασης των όσων περιγράφονται.

Κεφάλαιο 2

2 NavigationDrawerFragment Και Main Activity

Το παρόν κεφάλαιο, που είναι και το πρώτο στο οποίο θα μιλήσουμε για τον τρόπο υλοποίησης της εφαρμογής, θα αναφερθούμε στις κλάσεις **Navigation Drawer** και **Main Activity**. Καταρχάς, θα πρέπει να σημειωθεί ότι αυτές είναι οι δύο κλάσεις που παράγονται ***αυτόματα*** από το Android Studio εφόσον έχει επιλεγεί δημιουργία project το οποίο θα βασίζεται στο navigation drawer. Όπως θα φανεί και από τον κώδικα που θα παρατεθεί στην συνέχεια, πάνω στο βασικό template που μας παρέχεται έχουν αλλάξει πολύ λίγα πράγματα ώστε να μπορέσουμε να τα προσαρμόσουμε στις ανάγκες της δικής μας εφαρμογής.

```
public View onCreateView(LayoutInflater inflater, final ViewGroup container,
                         Bundle savedInstanceState) {
    mDrawerListView = (ListView) inflater.inflate(
            R.layout.fragment_navigation_drawer, container, false);
    mDrawerListView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
            selectItem(position);
        }
    });
    arrayTemp = getResources().getStringArray(R.array.drawer_elements);



//################################################################################
###############
    final TypedArray typedArray =
getResources().obtainTypedArray(R.array.sections_icons);
    mDrawerListView.setAdapter(new ArrayAdapter<String>(
            getActionBar().getThemedContext(),
            android.R.layout.simple_list_item_activated_1,
            android.R.id.text1,
            arrayTemp
    ){
        @TargetApi(Build.VERSION_CODES.JELLY_BEAN_MR1)
        @Override
        public View getView(int position, View convertView, ViewGroup parent)
{
            View v = super.getView(position,convertView,parent);
            int resourceId = typedArray.getResourceId(position,0);
            Drawable icon = getResources().getDrawable(resourceId);
            ((TextView)
v).setCompoundDrawablesRelativeWithIntrinsicBounds(icon,null,null,null);

            return v;
```

```
        }
    });

//#########################################################################
###############

    mDrawerListView.setItemChecked(mCurrentSelectedPosition, true);
    return mDrawerListView;
}
```

Παρατηρώντας το παραπάνω τμήμα του κώδικα που αντιστοιχεί στο **NavigationDrawerFragment.java** αρχείο βλέπουμε ότι η ***μοναδική*** αλλαγή που χρειάζεται να πραγματοποιήσουμε είναι η εισαγωγή των ονομάτων του μενού(θα αναλυθεί στην συνέχεια καθώς σχετίζεται άμεσα με την main activity κλάση) που θα υπάρχουν στο «συρτάρι» καθώς και των εικονιδίων που θα έχουν και τα οποία είναι ορισμένα στον πίνακα **drawer_elements** και **sections_icons** αντίστοιχα στο αρχείο strings.xml όπως φαίνεται παρακάτω:

```
<!-- Navigation Drawer Data -->
    <string-array name="drawer_elements">
        <item>Home</item>
        <item>Statistics</item>
        <item>Settings</item>
        <item>About</item>
    </string-array>
```

Λίγες είναι και οι αλλαγές που συμβαίνουν στο **MainActivity.java** αρχείο οι οποίες έχουν στόχο να αναγνωρίζεται ποιο μενού έχει επιλεγεί βάσει της θέσης του στο «συρτάρι». Αυτό επιτυγχάνεται μέσω του κώδικα που παρατίθεται παρακάτω καθώς και με μικρές τροποποιήσεις που συμβαίνουν στις κλάσεις στις οποίες αντιστοιχεί το κάθε μενού(και συγκεκριμένα στην μέθοδο **newInstance** της κάθε κλάσης) και που θα επεξηγηθούν αμέσως μετά.

```
public void onNavigationDrawerItemSelected(int position) {
    // update the main content by replacing fragments
    Fragment objFragment = null;
    switch (position){
        case 0:
            objFragment = HomeFragment.newInstance(position + 1);
            break;

        case 1:
            objFragment = StatisticsFragment.newInstance(position + 1);
            break;

        case 2:
            objFragment = SettingsFragment.newInstance(position + 1);
            break;
```

```
        case 3:
            objFragment = AboutFragment.newInstance(position + 1);
            break;
    }
    onSectionAttached(position);
    FragmentManager fragmentManager = getSupportFragmentManager();
    fragmentManager.beginTransaction()
            .replace(R.id.container, objFragment)
            .commit();
}
```

Μελετώντας τον κώδικα παραπάνω, βλέπουμε ότι υπάρχει ένα switch-case το οποίο ανάλογα με την **θέση** του κάθε αντικειμένου στο navigation drawer οδηγεί και στην κατάλληλη κλάση. Στο παρόν κεφάλαιο θα εξετάσουμε, χάριν παραδείγματος, το **HomeFragment** δεδομένου ότι οι τροποποποιήσεις που λαμβάνουν χώρα σε αυτό είναι **ακριβώς** οι ίδιες και στις υπόλοιπες κλάσεις με τις οποίες συνδέεται το switch-case.

Η αλλαγή που πρέπει να κάνουμε αφορά, όπως έχει ήδη αναφερθεί, την μέθοδο **newInstance** που αυτόματα γίνεται override σε κάθε νέα κλάση που δημιουργούμε. Η μέθοδος αυτή θα είναι μια public static μέθοδος και με τύπο ίδιο με την κλάση στην οποία ορίζεται. Άρα, στη HomeFragment κλάση που εξετάζουμε, θα είναι τύπου HomeFragment. Όταν ορίζεται αυτόματα, η μέθοδος λαμβάνει δύο παραμέτρους τύπου string όπως φανερώνει και ο παρακάτω κώδικας:

```
public static HomeFragment newInstance(String param1, String param2) {
    BlankFragment fragment = new BlankFragment();
    Bundle args = new Bundle();
    args.putString(ARG_PARAM1, param1);
    args.putString(ARG_PARAM2, param2);
    fragment.setArguments(args);
    return fragment;
}
```

Στην παρούσα εφαρμογή, όμως, χρειαζόμαστε να δημιουργείται το νέο fragment (αυτός είναι και ο σκοπός της συνάρτησης αυτής) με βάση το μενού που έχει επιλεγεί από τον χρήστη. Άρα θα πρέπει να τροποποιήσουμε τον ορισμό της **newInstance** ώστε, σε όλες τις κλάσεις που σχετίζονται με το navigation drawer, να δέχεται ως παράμετρο έναν ακέραιο ο οποίος θα αντιστοιχεί στη θέση του μενού που επιλέχθηκε. Η υλοποίησή της, επομένως, θα είναι ως εξής:

```
public static HomeFragment newInstance(int position) {
    HomeFragment fragment = new HomeFragment();

    Bundle args = new Bundle();
    args.putString(ARG_SEC_NUM, String.valueOf(position));
    //args.putString(ARG_PARAM2, param2);
```

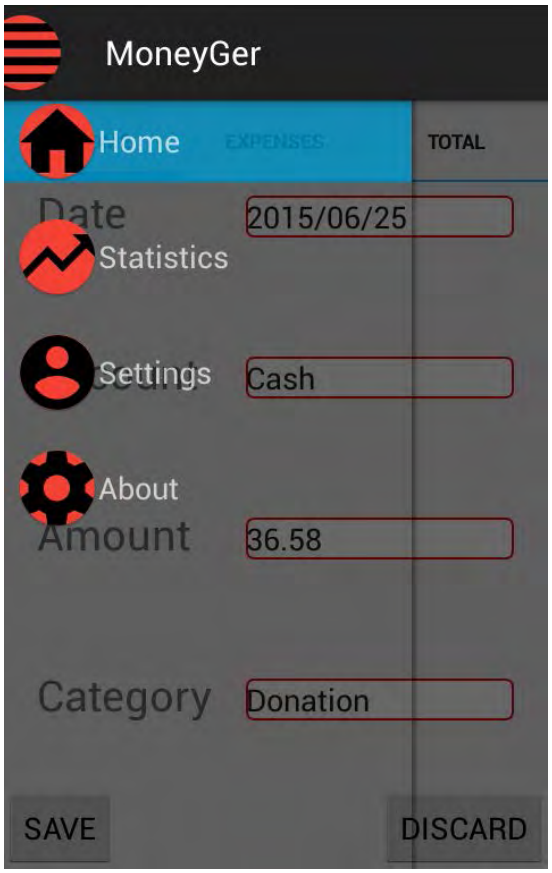
```
    fragment.setArguments(args);
    return fragment;
}
```

Τελευταία προσθήκη στον κώδικα της main activity, είναι η επιλογή των ονομάτων για το κάθε μενού, που όπως δείξαμε και στον κώδικα που υλοποιεί το ίδιο το «συρτάρι», βρίσκονται στον πίνακα drawer_elements. Η αντιστοίχιση των ονομάτων με τις θέσεις γίνεται ως εξής και γραφικά απεικονίζεται στην Εικόνα [2.1]:

```
public void onSectionAttached(int number) {
    arrayTemp = getResources().getStringArray(R.array.drawer_elements);
    switch (number) {
        case 0:
            mTitle = arrayTemp[0];
            break;

        case 1:
            mTitle = arrayTemp[1];
            break;

        case 2:
            mTitle = arrayTemp[2];
            break;

        case 3:
            mTitle = arrayTemp[3];
            break;
    }
}
```

Παρατίθεται και ο xml κώδικα που δημιουργεί το navigation drawer:

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

    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:choiceMode="singleChoice"
    android:divider="@android:color/transparent"
    android:dividerHeight="20dp"
    android:gravity="center_vertical"
    />
```



Εικόνα 2.1: Το «συρτάρι» που βλέπει ο χρήστης αν πατήσει το εικονίδιο ή σύρει το χέρι του από το αριστερό άκρο της συσκευής του

Με τα παραπάνω κλείνει το κεφάλαιο που περιγράφει τις αλλαγές που έλαβαν χώρα ώστε να προσαρμόσουν τα templates που παρέχονται από το android studio στις ανάγκες της εφαρμογής μας. Από το επόμενο κεφάλαιο θα αναπτύσσονται σταδιακά όλες οι επιμέρους κλάσεις που συνθέτουν την εφαρμογή και θα αναλύονται οι λειτουργίες που κάθε μία τους επιτελεί.

Κεφάλαιο 3

3 HOME FRAGMENT

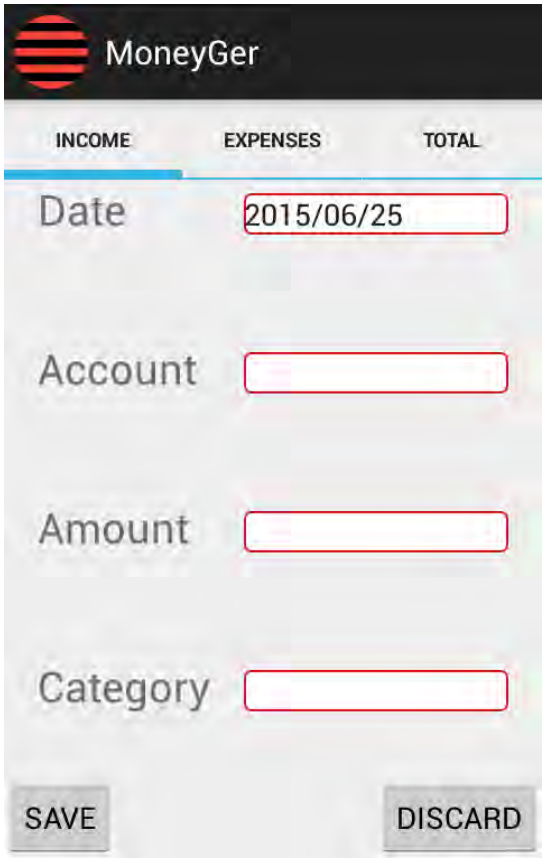
Το **HomeFragment.java** είναι η κύρια οθόνη που εμφανίζεται στον χρήστη όταν ανοίξει για πρώτη φορά την εφαρμογή. Η οθόνη αυτή περιλαμβάνει ένα tab host με τρία βασικά μενού, Έσοδα, Έξοδα και Σύνολο τα οποία και θα αναλυθούν πλήρως στην συνέχεια.

Παρουσιάζεται παρακάτω ο κώδικας που υπάρχει στο αρχείο fragment_home.xml και είναι αυτός ο οποίος σχεδιάζει την κύρια οθόνη που περιγράφηκε προηγουμένως:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.dimitris.moneyger.HomeFragment">

    <!-- TODO: Update blank fragment layout -->

    <TabHost
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:id="@+id/tabHost"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true">

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

            <TabWidget
                android:id="@android:id/tabs"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"/>

            <FrameLayout
                android:id="@android:id/tabcontent"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content">

                <LinearLayout
                    android:id="@+id/income"
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:orientation="vertical"/>

                <LinearLayout
                    android:id="@+id/expenses"
                    android:layout_width="fill_parent"
```

```
                android:layout_height="fill_parent"
                android:orientation="vertical"/>

            <LinearLayout
                android:id="@+id/total"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:orientation="vertical"/>

        </FrameLayout>
    </LinearLayout>
</TabHost>
</RelativeLayout>
```

Στον παραπάνω κώδικα φαίνεται το TabHost έχει δημιουργηθεί σαν μέρος του Relative Layout που χρησιμοποιείται και ο οποίος περιέχει τα τρία tabs όπως παρουσιάζονται και στην Εικόνα 3.1 (από προεπιλογή αυτό που φαίνεται στον χρήστη θα είναι το tab των εσόδων):



Εικόνα 3.1: Αρχική Οθόνη

Αναλύοντας, τώρα, την λειτουργία του **HomeFragment.java** θα πρέπει να αναφέρουμε ότι μόλις δημιουργηθεί το view (δηλαδή αυτό που τελικά ο χρήστης της εφαρμογής βλέπει) και

χρησιμοποιώντας την μέθοδο **onCreateView** που παρέχεται, εισάγουμε προγραμματιστικά τα tabs που σχεδιάζονται παραπάνω. Το μέρος του κώδικα που επιτυγχάνει το παραπάνω παρουσιάζεται στην συνέχεια:

```
@Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        mTabHost = new FragmentTabHost(getActivity());
        mTabHost.setup(getActivity(), getChildFragmentManager(),
      R.id.tabHost);

        mTabHost.addTab(mTabHost.newTabSpec("income").setIndicator("Income"),
                IncomeFragment.class, null);

      mTabHost.addTab(mTabHost.newTabSpec("expenses").setIndicator("Expenses",
                ExpensesFragment.class, null);
        mTabHost.addTab(mTabHost.newTabSpec("total").setIndicator("Total"),
                TotalFragment.class, null);


        return mTabHost;
    }
```

Αξίζει να σημειωθεί η χρησιμοποίηση της μεθόδου **setIndicator** η οποία προσδίδει την λειτουργικότητα του κάθε επιμέρους tab που υπάρχει στον TabHost (στον οποίο έχει δοθεί το όνομα mTabHost), λειτουργικότητα που αποτελείται από δύο μέρη, το όνομα που θα έχει το εν λόγω tab και στην κλάση που θα «δείχνει» όταν ο χρήστης επιλέξει να το χρησιμοποιήσει.

Έτσι:

- το tab **Income** θα αντιστοιχεί στην κλάση IncomeFragment.class
- το tab **Expenses** στην κλάση ExpensesFragment.class
- το tab **Total** στην κλάση TotalFragment.class

που θα παρουσιαστούν στη συνέχεια.

3.1 Income Fragment

Σε αυτό το fragment επιτελείται μία από τις κυριότερες λειτουργίες της εφαρμογής, η εισαγωγή και μορφοποιημένη αποθήκευση των εσόδων του χρήστη. Όπως φαίνεται και στην Εικόνα 3.1, το view -η απεικόνιση δηλαδή- για το εν λόγω fragment αποτελείται από τέσσερα TextViews, τέσσερα EditTexts και δύο Buttons.

Τα TextViews είναι widgets που εισάγονται έτσι ώστε να είμαστε σε θέση να εμφανίζουμε κείμενο στην οθόνη μιας android συσκευής. Με την χρησιμοποίηση, όμως, της μεθόδου **setOnClickListener** μπορούμε να προσθέσουμε επιπλέον ιδιότητες στο κείμενο αυτό αφού πλέον μπορεί να αλληλοεπιδράσει με τον χρήστη ο οποίος «πατώντας» το μπορεί να εκτελέσει κάποια συγκεκριμένη λειτουργία που αυτό επιτελεί. Η παρούσα εφαρμογή το εκμεταλλεύεται αυτό στο έπακρο και επομένως η όποια εισαγωγή στοιχείων από το εξωτερικό περιβάλλον(τόσο στο παρόν fragment όσο και στα αντίστοιχα expenses και total) γίνεται επιλέγοντας το αντίστοιχο κείμενο. Για την διευκόλυνση του χρήστη υπάρχει και αντίστοιχο μήνυμα το οποίο και εμφανίζεται κατά την «φόρτωση» του view.

Τα EditTexts είναι και αυτά widgets που παρέχονται ώστε ο χρήστης να είναι σε θέση να εισάγει κείμενο αν τα επιλέξει. Το κείμενο αυτό εμφανίζεται σε έναν μορφοποιημένο χώρο που παρέχει το ίδιο το widget. Αν χρησιμοποιηθεί, όμως, η μέθοδος **setKeyListener(null)** για ένα widget τύπου EditText τότε η λειτουργικότητά του χάνεται, από την άποψη ότι πλέον πατώντας το δεν μπορεί να εισαχθεί κείμενο. Το μορφοποιημένο περιβάλλον που παρέχει, παρόλα αυτά, παραμένει ενεργό και μπορεί να εμφανίσει κείμενο (το οποίο με κάποιον τρόπο έχει εισαχθεί). Άρα το κείμενο που θα εισάγει ο χρήστης μέσω του TextView widget που αναλύθηκε παραπάνω θα εμφανίζεται στο αντίστοιχο EditText.

Με βάση τα παραπάνω μπορούν να προκύψουν δύο ερωτήματα, πρώτον γιατί δεν χρησιμοποιούνται EditTexts για την αλληλεπίδραση με τον εξωτερικό κόσμο και δεύτερον γιατί να χρειάζεται να απενεργοποιηθεί η κύρια λειτουργία του EditText και να μην χρησιμοποιηθεί αντ' αυτού ένα απλό TextView που θεωρητικά κάνει το ίδιο (απλά εμφανίζει κάποιο επιλεγμένο κείμενο).

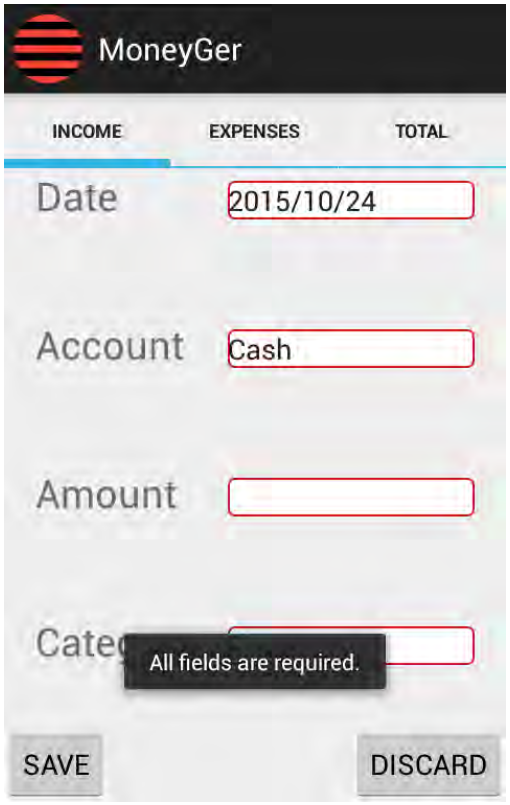
Απαντώντας στο πρώτο ερώτημα, πρέπει να αναφέρουμε ότι η εισαγωγή κειμένου με απευθείας χρήση ενός EditText δεν επιτρέπει την μορφοποιημένη του είσοδο. Στο παρόν fragment view (όπως και στο expenses view που θα περιγράψουμε παρακάτω) χρειαζόμαστε εισαγωγή ημερομηνίας, ενός ποσού, μιας κατηγορίας από την οποία προέρχεται το εισαχθέν ποσό και μιας κατηγορίας βάσει της οποίας περιγράφεται η φύση του ποσού(κατάθεση σε λογαριασμό, μετρητά κτλπ). Έτσι αν, για παράδειγμα, χρησιμοποιούσαμε EditText για την εισαγωγή ημερομηνίας ο χρήστης θα μπορούσε να την εισάγει χωρίς κάποια προδιαγεγραμμένη μορφοποίηση (ο δε κακόβουλος θα μπορούσε να εισάγει γράμματα) οδηγώντας την εφαρμογή σε αναγκαστικό τερματισμό(crash) αφού δεν θα αναγνώριζε κάποιο σωστό format για να προχωρήσει στην διαδικασία της αποθήκευσης. Το ίδιο θα μπορούσε να συμβεί και με οποιοδήποτε άλλο μέρος της εφαρμογής απαιτεί εισαγωγή στοιχείων από τον χρήστη. Για τον λόγο αυτό κάθε εισαγωγή γίνεται αρχικά με χρήση TextView widgets καθένα από τα οποία ενεργοποιεί ένα dialog box(για τα οποία και θα μιλήσουμε διεξοδικά παρακάτω) για την αυστηρά μορφοποιημένη εισαγωγή δεδομένων.

Όσον αφορά το δεύτερο ερώτημα, η επιλογή της απενεργοποίησης της ιδιότητας εισαγωγής κειμένου σε ένα EditText φαίνεται να αχρηστεύει το ίδιο το widget αλλά κάτι τέτοιο δεν ισχύει. Δεν ισχύει γιατί αν απλά χρησιμοποιούσαμε ένα TextView για την εμφάνιση των δεδομένων δεν θα μπορούσαμε να το εμπλουτίσουμε με κάποια επιπλέον μορφοποίηση όπως το πλαίσιο που

φαίνεται και στην Εικόνα [3.1]. Στο σημείο αυτό αξίζει να αναφερθεί πώς το πλαίσιο που περιβάλλει το κείμενο δεν υπάρχει έτοιμο ως template στο android studio αλλά είναι προγραμματιστικά δημιουργημένο όπως παρουσιάζεται στον παρακάτω κώδικα xml που περιλαμβάνεται στο αρχείο back.xml:

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
android:shape="rectangle" >
    <solid android:color="#ffffff" />
    <stroke android:width="1dip" android:color="#ffd51217"/>
    <corners android:radius="4dp"/>
</shape>
```

Τέλος, τα δύο κουμπιά(Buttons) που εμφανίζονται στο view που εξετάζεται, αποτελούν και αυτά με την σειρά τους widgets τα οποία εκμεταλλεύονται την μέθοδο **setOnClickListener** ώστε να αποθηκεύσουν τα μορφοποιημένα δεδομένα στην βάση ή να καθαρίσουν όλα τα πεδία αντίστοιχα ανάλογα με το ποιο από τα δυο θα επιλεγεί. Στο σημείο αυτό, είναι χρήσιμο να αναφερθεί μια συγκεκριμένη ιδιότητα του κουμπιού Save που έχει την μορφή ελέγχου. Εάν ο χρήστης επιλέξει να πατήσει το Save Button αφού προηγουμένως δεν έχει συμπληρώσει όλα τα πεδία θα εμφανιστεί ένα μήνυμα σφάλματος που θα τον ενημερώνει ότι όλα τα πεδία είναι υποχρεωτικά όπως φαίνεται και από την Εικόνα [3.2].



Εικόνα 3.2: Μήνυμα Ειδοποίησης Υποχρεωτικών Πεδίων

Εφόσον αναλύθηκε, σε γενικές γραμμές, πώς γίνεται η εισαγωγή, αποθήκευση και ακύρωση των δεδομένων του χρήστη (σε επόμενο υποκεφάλαιο θα γίνει πιο αναλυτική αναφορά σε αυτά), είναι χρήσιμο να συμπεριλάβουμε και τον κώδικα που είναι υπεύθυνος για το σχεδιαστικό μέρος που υποστηρίζει τις παραπάνω λειτουργίες:

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.dimitris.moneyger.IncomeFragment">

    <TableLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="left|top">

        <TableRow
            android:layout_gravity="left">
            <TextView
                android:layout_width="wrap_content"
                android:text="@string/dateField"
                android:id="@+id/DateTxt"
                android:textSize="25sp"
                android:layout_marginLeft="20dp"/>
            <EditText
                android:id="@+id/Date"
                android:layout_height="wrap_content"
                android:layout_marginLeft="20dip"
                android:layout_marginRight="20dip"
                android:layout_weight="1"
                android:background="@drawable/back"
                android:inputType="date">
            </EditText>

        </TableRow>


        <TableRow
            android:layout_gravity="left">
            <TextView
                android:layout_width="wrap_content"
                android:text="@string/accountField"
                android:id="@+id/AccountTxt"
                android:textSize="25sp"
                android:paddingTop="60dp"
                android:layout_marginLeft="20dp"/>

            <EditText
                android:id="@+id/Account"
                android:layout_height="wrap_content"
                android:layout_marginLeft="20dip"
                android:layout_marginRight="20dip"
```

```
            android:layout_weight="1"
            android:background="@drawable/back"
            android:inputType="text">
        </EditText>

    </TableRow>

    <TableRow
        android:layout_gravity="left">

        <TextView
            android:layout_width="wrap_content"
            android:text="@string/amountField"
            android:id="@+id/AmountTxt"
            android:textSize="25sp"
            android:paddingTop="60dp"
            android:layout_marginLeft="20dp"/>


        <EditText
            android:id="@+id/Amount"
            android:layout_height="wrap_content"
            android:layout_marginLeft="20dip"
            android:layout_marginRight="20dip"
            android:layout_weight="1"
            android:background="@drawable/back"
            android:inputType="numberDecimal"
            android:digits="1234567890.,"
            android:imeOptions="actionNext">

        </EditText>

    </TableRow>


    <TableRow
        android:layout_gravity="left">
        <TextView
            android:layout_width="wrap_content"
            android:text="@string/categoryField"
            android:id="@+id/CategoryTxt"
            android:paddingTop="60dp"
            android:textSize="25sp"
            android:layout_marginLeft="20dp"/>

        <EditText
            android:id="@+id/Category"
            android:layout_height="wrap_content"
            android:layout_marginLeft="20dip"
            android:layout_marginRight="20dip"
            android:layout_weight="1"
            android:background="@drawable/back"
            android:inputType="text">

        </EditText>

    </TableRow>
```

```
    </TableLayout>


    <Button
        android:text="@string/saveButton"
        android:clickable="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/save"
        android:layout_gravity="left|bottom"/>


    <Button
        android:text="@string/discardButton"
        android:clickable="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/discard"
        android:layout_gravity="right|bottom"/>

</FrameLayout>
```

Θα περιγράψουμε, ακολούθως, όλα τα dialog boxes που εμφανίζονται κατά το «click» σε ένα TextView widget και που χρησιμοποιούνται για εισαγωγή δεδομένων στην εφαρμογή.

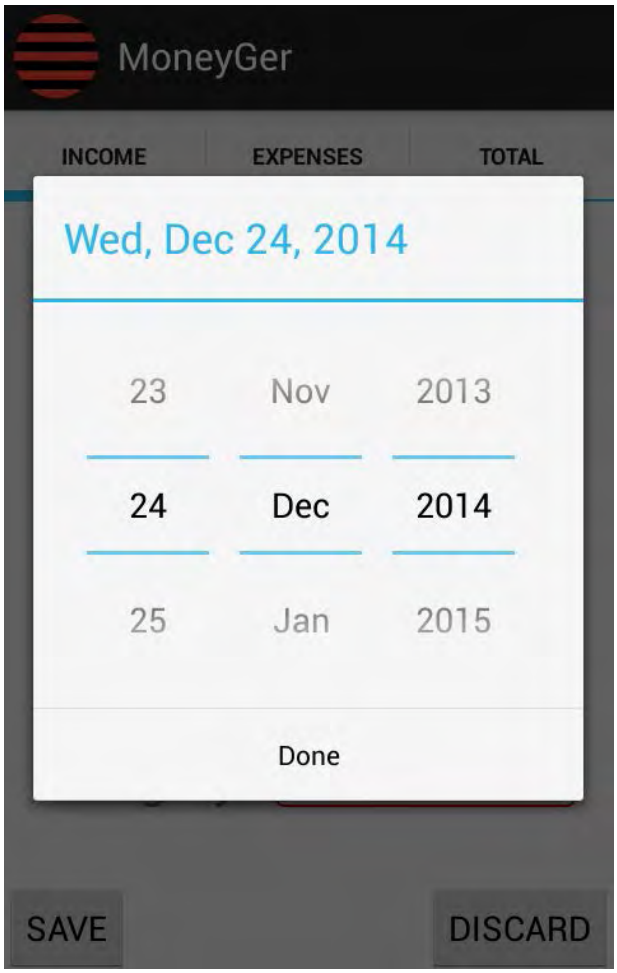
3.1.1 ΛΕΙΤΟΥΡΓΙΑ ΕΙΣΑΓΩΓΗΣ ΗΜΕΡΟΜΗΝΙΑΣ

Όπως αναφέρθηκε παραπάνω, η εισαγωγή της ημερομηνίας γίνεται αν πατήσει ο χρήστης το αντίστοιχο TextView «Date» οπότε και θα εμφανιστεί το dialog box που θα περιέχει το ημερολόγιο και που παρουσιάζεται στην Εικόνα [3.3]. Το dialog box αυτό καλείται DatePicker Dialog box και ο κώδικας που το δημιουργεί ακολουθεί στην συνέχεια.

```
//Setting the click listener for the Date textView.
dateDefault();
dateTXT.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        new DatePickerDialog(getActivity(), date, myCalendar
                .get(Calendar.YEAR), myCalendar.get(Calendar.MONTH),
                myCalendar.get(Calendar.DAY_OF_MONTH)).show();
    }
});
```

```
//Creating the datePicker popup window.
final DatePickerDialog.OnDateSetListener date = new
DatePickerDialog.OnDateSetListener() {
    @Override
    public void onDateSet(DatePicker view, int year, int monthOfYear,
                          int dayOfMonth) {
        // TODO Auto-generated method stub
        myCalendar.set(Calendar.YEAR, year);
        myCalendar.set(Calendar.MONTH, monthOfYear);
        myCalendar.set(Calendar.DAY_OF_MONTH, dayOfMonth);
        updateLabel();
    }
};


//Changing the text of the appropriate TextView.
private void updateLabel() {
    String myFormat = "yyyy/MM/dd";
    SimpleDateFormat sdf = new SimpleDateFormat(myFormat, Locale.US);
    dateText.setText(sdf.format(myCalendar.getTime()));
}
```

Ο κώδικας για το DatePickerDialog box παρατηρούμε ότι αποτελείται από τρία βασικά μέρη. Στο πρώτο μέρος, προσθέσουμε την λειτουργικότητα στο TextView ώστε να δημιουργήσει ένα αντικείμενο τύπου DialogPickerDate για το view στο οποίο βρισκόμαστε (αυτό περνάει ως παράμετρος με χρήση της μεθόδου **getApplication()**) και με τις παραμέτρους που αυτό απαιτεί, δηλαδή την ημέρα, τον μήνα και τον χρόνο τα οποία και λαμβάνουμε με αντίστοιχες κλήσεις μεθόδων της abstract κλάσης Calendar. Στο δεύτερο μέρος έχουμε την δημιουργία του εν λόγω αντικειμένου το οποίο δημιουργείται βάσει των παραμέτρων μας και στη συνέχεια καλεί την μέθοδο **updateLabel()**. Στο τρίτο και τελευταίο μέρος, έχουμε την υλοποίηση της συνάρτησης **updateLabel()** η οποία θέτει συγκεκριμένο format στην ημερομηνία και κάνει το πρώτο EditText (στο οποίο και αναφερόμαστε μέσω του reference dateText) να εμφανίσει την επιλεγμένη ημερομηνία.



Εικόνα 3.3: Dialog Box για επιλογή ημερομηνίας

Αξίζει να εστιάσουμε σε δύο σημεία, πρώτον στο format που θα έχει η ημερομηνία και δεύτερον στην μέθοδο **dateDefault()** που χρησιμοποιείται στην αρχή του πρώτου μέρους του κώδικα. Καταρχάς το format της ημερομηνίας είναι το συγκεκριμένο-δηλαδή πρώτα το έτος με τέσσερα ψηφία, ακολούθως ο μήνας και τέλος η μέρα με δύο ψηφία- καθώς έτσι θα αποθηκευτεί και στην βάση δεδομένων. Η SQLite -η οποία χρησιμοποιείται για την δημιουργία της βάσης καθώς και για οποιοδήποτε διάβασμα ή οποιαδήποτε εγγραφή- υποστηρίζει αυστηρά μόνο το συγκεκριμένο format. Σε κάθε άλλη περίπτωση η εγγραφή στην βάση θα γινότανε σωστά αλλά θα είχαμε προβλήματα κατά την ανάγνωση και ανάκτηση δεδομένων από αυτή σε queries που θα βασίζονταν στην αναζήτηση με χρήση ημερομηνιών. Όσον αφορά την μέθοδο **dateDefault()** την χρησιμοποιούμε έτσι ώστε κάθε μέρα να εμφανίζεται από προεπιλογή η τρέχουσα ημερομηνία με αποτέλεσμα ο χρήστης να μην χρειάζεται σε κάθε εισαγωγή συναλλαγής να την επιλέγει εκ νέου. Ο κώδικας που υλοποιεί την συνάρτηση αυτή παρουσιάζεται στην συνέχεια.

```
//Displaying the current date in date text field.
private void dateDefault(){
    Calendar cal = Calendar.getInstance();
    SimpleDateFormat df = new SimpleDateFormat("yyyy/MM/dd");
    String formattedDate = df.format(cal.getTime());
    dateText.setText(formattedDate);
}
```

3.1.2 ΛΕΙΤΟΥΡΓΙΑ ΕΠΙΛΟΓΗΣ ΚΑΤΗΓΟΡΙΑΣ ΕΝΤΑΞΗΣ ΠΟΣΟΥ

Αφού επιλεγεί η ημερομηνία (εισάγοντας νέα ή αφήνοντας την τρέχουσα ημερομηνία ως είσοδο στην εφαρμογή) ο χρήστης θα πρέπει να επιλέξει μία από τις διαθέσιμες κατηγορίες μέσω των οποίων γνωστοποιείται η φύση του ποσού που παρέλαβε ο χρήστης. Πρόκειται, δηλαδή, για χρήματα που πιθανόν να εντάσσονται στην κατηγορία μετρητά, στην τράπεζα ως κατάθεση ή στην κάρτα καθώς και σε κάποια άλλη κατηγορία υπό την επιλογή other. Και πάλι χρησιμοποιείται ένα dialog box όπως αυτό που απεικονίζεται στην Εικόνα [3.4] ενώ ο κώδικας κατασκευής του ακολουθεί.

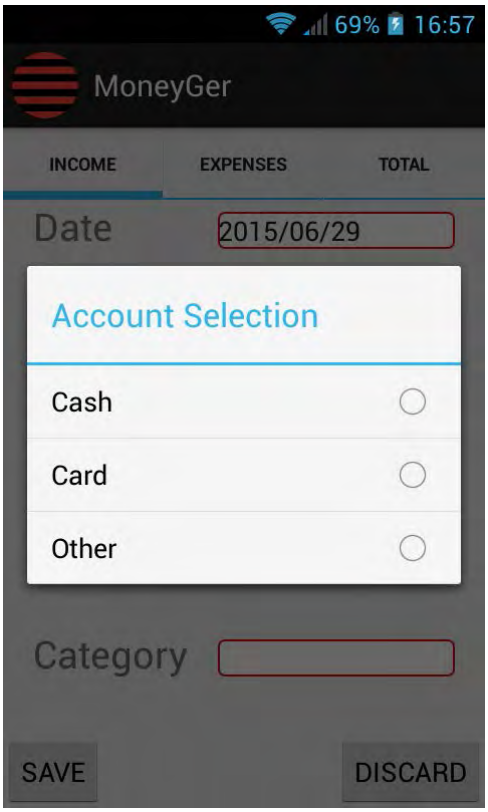
```
//The Dialog box used for the account menu.
private void accountDialog(){
    final TypedArray accountIncDial =
getResources().obtainTypedArray(R.array.accountIncDialog);

    dialogBuilder = new AlertDialog.Builder(getActivity());


    dialogBuilder.setTitle(R.string.account_select);
    dialogBuilder.setSingleChoiceItems(R.array.accountIncDialog, -1,
            new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    Toast.makeText(getActivity(), R.string.account_select_success,
                            Toast.LENGTH_SHORT).show();
                    accountText.setText(accountIncDial.getText(which));
                    dialog.dismiss();
                }
            });
    AlertDialog accountSelectDialog = dialogBuilder.create();
    accountSelectDialog.show();

}
```

Από τον παραπάνω κώδικα αυτό που είναι άξιο προσοχής είναι ο τρόπος με τον οποίο πραγματοποιείται η «φόρτωση» των στοιχείων που περιέχει η λίστα απλής επιλογής που

εφαρμόζεται και η γνωστοποίηση της κατηγορίας που επιλέχθηκε. Καταρχάς, όπως έχει αναφερθεί ήδη, όλα τα strings που χρησιμοποιούνται από την εφαρμογή βρίσκονται αποθηκευμένα στο αρχείο strings.xml και η προσπέλασή τους (φόρτωση) γίνεται με χρήση references. Όταν ορίζουμε έναν πίνακα από strings στο strings.xml αρχείο (όπως παρουσιάζεται στον κώδικα παρακάτω) η προσπέλαση του κάθε στοιχείου του γίνεται κάπως διαφορετικά.

```
<!-- Account PopUp Menu for INCOME-->
    <string-array name="accountIncDialog">
        <item>Cash</item>
        <item>Card</item>
        <item>Other</item>
    </string-array>
```

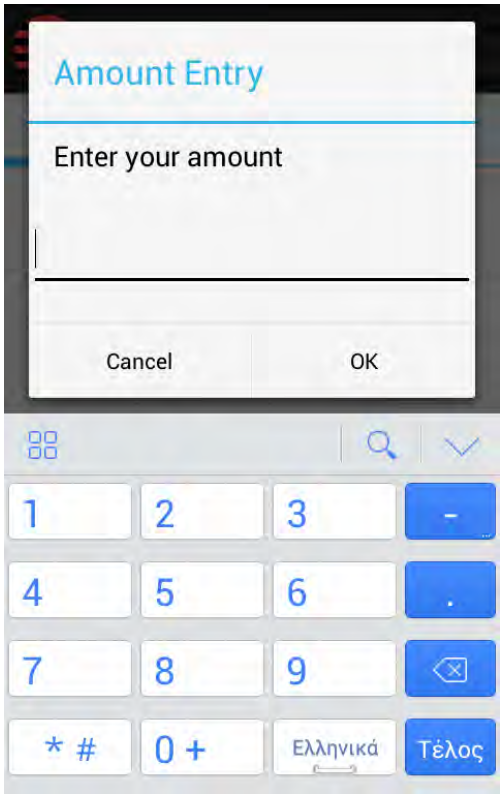
Στην μέθοδο που κατασκευάζει την λίστα των κατηγοριών, **singleChoiseItems**, περνάμε ως παράμετρο *ολόκληρο* τον πίνακα χρησιμοποιώντας το **R.array.accountIncDialog**. Αφού εμφανιστεί η λίστα πρέπει να ξέρουμε και ποια κατηγορία επιλέχθηκε. Κατά το override της μεθόδου **onClick** και χρησιμοποιώντας την παράμετρό της **which**, μπορούμε να πάρουμε την «θέση» του αντικειμένου που επέλεξε ο χρήστης ώστε να το εμφανίσουμε στο κατάλληλο EditText widget. Στην εικόνα που ακολουθεί παρουσιάζεται η λίστα επιλογής κατηγορίας.



Εικόνα 3.4: Επιλογή κατηγορίας ένταξης εσόδων

3.1.3 ΛΕΙΤΟΥΡΓΙΑ ΕΙΣΑΓΩΓΗΣ ΠΟΣΟΥ

Στο σημείο αυτό ο χρήστης θα πρέπει να εισάγει ταο ποσό που θέλει. Για να γίνει αυτό χρησιμοποιείται ένα dialog box όπως αυτό της Εικόνας [3.5] το οποίο περιλαμβάνει ένα EditText μορφοποιημένο ώστε να δέχεται αποκλειστικά και μόνο αριθμούς όπως το πληκτρολόγιο του κινητού κατά την πραγματοποίηση μιας κλήσης.



Εικόνα 3.5: Εισαγωγή Ποσού

Ο κώδικας για το παραπάνω dialog box παρουσιάζεται στην συνέχεια:

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

        //Creating custom dailog boxes to get the amount of money the user
        //wants to enter.
        dialogBuilder = new AlertDialog.Builder(getActivity());
        dialogBuilder.setTitle(R.string.amountTitle);
        dialogBuilder.setMessage(R.string.amountPrompt);
        dialogBuilder.setCancelable(false);

        //For the user to enter money an EditText field is used with input
        //type phone
```

```
        //so as not to display letters or other special characters.
        final EditText money_entered = new EditText(getView().getContext());
        money_entered.setInputType(InputType.TYPE_CLASS_PHONE);

        dialogBuilder.setView(money_entered);


        dialogBuilder.setPositiveButton(R.string.ok, new
        DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                String salarySTR =
money_entered.getEditableText().toString();

                if (salarySTR.contains(",")){

                       salarySTR = salarySTR.replace(',','.');
                }


                amountEntered = Double.parseDouble(salarySTR);
                amountEntered = twoDecimals.roundTwoDecimals(amountEntered);
                amountText.setText(salarySTR);
            }
        });

        dialogBuilder.setNegativeButton(R.string.cancel, new DialogInterface.
                OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                dialog.cancel();
            }
        });

        AlertDialog alert = dialogBuilder.create();
        alert.show();

    }
});
```

Για να περιοριστεί το EditText που περιέχεται στο view (στην «οθόνη») του dialog box ώστε να εμφανίζει μόνο αριθμητικά ψηφία(όπως προαναφέρθηκε) πρέπει να αλλάξει ο τύπος των δεδομένων που δέχεται το widget σε **TYPE_CLASS_PHONE** έτσι ώστε να εμφανίζεται ένα πληκτρολόγιο ανάλογο με αυτό της εικόνας. Όταν εισαχθεί το ποσό και ο χρήστης πατήσει OK στο dialog box, ενεργοποιείται η μέθοδος **setPositiveButton** η οποία μετατρέπει το κείμενο από τύπο String σε τύπο Double, αφήνει στο ποσό μόνο δύο δεκαδικά ψηφία καλώντας την μέθοδο **roundTwoDecimals** και εν τέλει εμφανίζει το ποσό στο κατάλληλο EditText πλαίσιο.

Ένα πολύ σημαντικό σημείο του παραπάνω κώδικα είναι οι δυο γραμμές που ακολουθούν και αποτρέπουν ένα σοβαρό **bug**:

```
if (salarySTR.contains(",")){
    salarySTR = salarySTR.replace(',','.');
}
```

Το bug στο οποίο αναφερόμαστε πρόκυπτε από την χρήση **ελληνικού πληκτρολογίου** κατά την εισαγωγή του ποσού στο dialog box. Στο ελληνικό πληκτρολόγιο όταν προσπαθήσει ο χρήστης να εισάγει ένα δεκαδικό ποσό η υποδιαστολή εκλαμβάνεται ως κόμμα με αποτέλεσμα η εφαρμογή να οδηγούνταν σε crash αφού το μόνο που αναγνωρίζει σε έναν δεκαδικό αριθμό είναι η τελεία. Έτσι ο παραπάνω κώδικας αυτό που κάνει είναι να ελέγχει αν το ποσό που εισήγαγε ο χρήστης περιέχει κόμμα και αν βρεθεί να το μετατρέψει σε τελεία ώστε να αναγνωρίζεται και να γίνει στη συνέχεια η μετατροπή του string σε Double(όπως αναφέρθηκε και προηγουμένως).

Η μέθοδος **roundTwoDecimals** υλοποιείται μέσα στην public κλάση **RoundDecimals** και ο κώδικας που υλοποιεί είναι ο εξής:

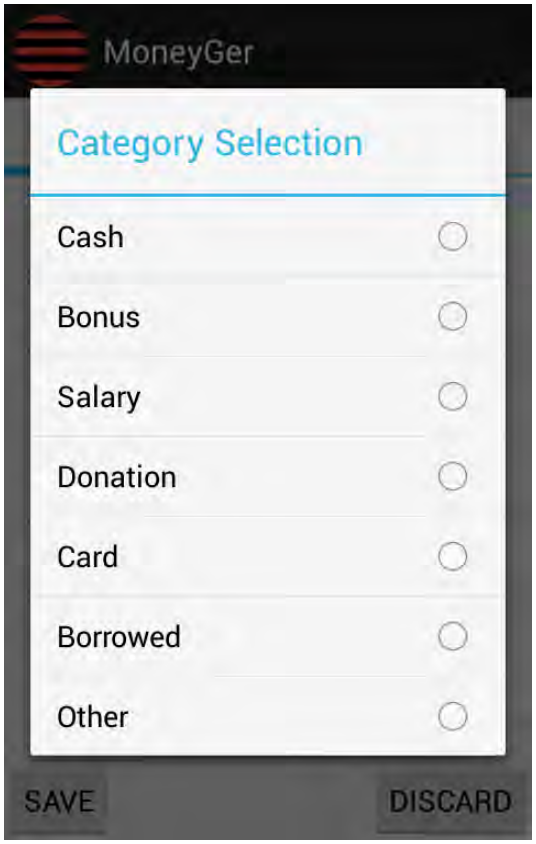
```
//Allowing only two decimals and rounding to the closest double.
public double roundTwoDecimals(double number){

    return Math.floor(number * 100.0) / 100.0;
}
```

3.1.4 ΛΕΙΤΟΥΡΓΙΑ ΕΠΙΛΟΓΗΣ ΚΑΤΗΓΟΡΙΑΣ ΠΡΟΕΛΕΥΣΗΣ ΕΣΟΔΩΝ

Ακριβώς όπως περιγράψαμε και προηγουμένως στην επιλογή κατηγορίας για την φυσική ένταξη των εσόδων έτσι και εδώ υπάρχει ένα αντίστοιχο dialog box που περιέχει μία λίστα απλής επιλογής ώστε ο χρήστης να είναι σε θέση να κατηγοριοποιήσει την προέλευση των εσόδων που εισάγει. Το dialog box απεικονίζεται στην Εικόνα [3.6] και ο κώδικας που το υλοποιεί ακολουθεί:

```
//The Dialog box used for the category menu.
private void categoryDialog(){
    final TypedArray categoryIncDial =
```

```
getResources().obtainTypedArray(R.array.
            categoryIncDialog);

    dialogBuilder = new AlertDialog.Builder(getActivity());

    dialogBuilder.setTitle(R.string.category_select);
    dialogBuilder.setSingleChoiceItems(R.array.categoryIncDialog, -1,
            new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    Toast.makeText(getActivity(),
R.string.category_select_success,
                            Toast.LENGTH_SHORT).show();
                    categoryText.setText(categoryIncDial.getText(which));
                    dialog.dismiss();
                }
            });
    AlertDialog accountSelectDialog = dialogBuilder.create();
    accountSelectDialog.show();
}
```



Εικόνα 3.6: Επιλογή κατηγορίας προέλευσης εσόδων

3.1.5 ΛΕΙΤΟΥΡΓΙΑ ΑΠΟΘΗΚΕΥΣΗΣ ΔΕΔΟΜΕΝΩΝ

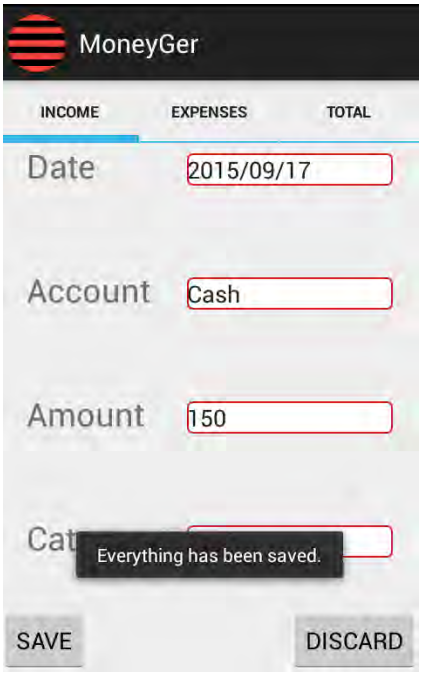
Μία από τις σημαντικότερες λειτουργίες που επιτελούνται είναι αυτή της αποθήκευσης των δεδομένων στην βάση δεδομένων μέσω του Save κουμπιού. Εκτός από την αποθήκευση, το κουμπί αυτό εξασφαλίζει ότι όλα τα πεδία είναι συμπληρωμένα και με την μορφοποίηση που πρέπει. Ο κώδικας που εξασφαλίζει τις παραπάνω λειτουργίες είναι ο εξής:

```
/**
 * Setting the click listeners used by the buttons of the view.
 */
saveButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        dateStr         = dateText.getText().toString();
        accountStr      = accountText.getText().toString();
        amountStr       = amountText.getText().toString();
        categoryStr     = categoryText.getText().toString();


        //Popup message in case where some of the fields are not filled.
        if (dateStr.equals("") || amountStr.equals("") ||
        accountStr.equals("") ||categoryStr.equals("")){

        Toast.makeText(getActivity(),R.string.fields_required,Toast.LENGTH_SHO
        RT).show();
        }

        //If all textFields are filled proceed(BUG FIXED).
        else {

            resultValidation = amountValidation(amountStr);


            /**
             * 2 decimal digits for all doubles.
             */

            if (resultValidation == 1) {
                amountEntered = Double.parseDouble(amountStr);
                amountEntered = twoDecimals.roundTwoDecimals(amountEntered);

                Transactions transaction = new Transactions(dateStr,
                       categoryStr,
                        accountStr, amountEntered, incomeFrag,0.0);

                myDBHandler.addRow(transaction);

                Toast.makeText(getActivity(),R.string.saved_all,
```

```
                        Toast.LENGTH_SHORT).show();
                }
                else{
                    amountText.setText("");
                }
            }
        }
});
```

Το σημείο που αξίζει να παρατηρήσουμε εδώ είναι η εισαγωγή των δεδομένων στην βάση. Καταρχάς, για να μπορέσουμε να γράψουμε(αντίστοιχα και να διαβάσουμε από την βάση) στην βάση θα πρέπει να ορίσουμε ένα αντικείμενο τύπου **dbHandler**, που είναι η κλάση η οποία είναι υπεύθυνη για την δημιουργία και διαχείριση της βάσης. Το αντικείμενο αυτό θα καλέσει τον constructor της **dbHandler.class** με τις κατάλληλες παραμέτρους(για την κλάση αυτή θα μιλήσουμε αναλυτικά σε ερχόμενο κεφάλαιο) και θα αρχικοποιηθεί ώστε να μπορεί να χρησιμοποιήσει τις public μεθόδους που η κλάση προσφέρει. Μία εξ' αυτών είναι και η μέθοδος **addRow** η οποία και παίρνει μια παράμετρο τύπου **Transactions**.

Η κλάση Transactions είναι αυτή η οποία ορίζει και υλοποιεί όλες τις **setter** και **getter** μεθόδους για την εγγραφή στην βάση καθώς και για την ανάγνωση από την βάση. **Ο constructor της δέχεται σαν παράμετρο ό,τι βάζει ο χρήστης ως δεδομένα στα dialog boxes του IncomeFragment καθώς και μια static παράμετρο που διαχωρίζει τα έσοδα από τα έξοδα.** Με την κλήση της **addRow** και τις παραμέτρους που προέρχονται από τα δεδομένα του χρήστη(οι οποίες έχουν συμπεριληφθεί σε μία μεταβλητή τύπου Transactions) η εγγραφή στην βάση είναι επιτυχής, όπως φαίνεται και στην Εικόνα [3.7].



Εικόνα 3.7: Επιτυχής αποθήκευση δεδομένων στη βάση

3.2 EXPENSES FRAGMENT

Το δεύτερο Tab στο κύριο μενού(TabHost) του HomeFragment χρησιμοποιείται ως μέσο εισαγωγής δεδομένων που αναφέρονται σε έξοδα. Ο κώδικας, τα dialog boxes, τα μηνύματα, οι έλεγχοι και η λειτουργικότητα είναι **ακριβώς** ίδιοι με το **IncomeFragment**. Η μόνη διαφορά, όπως αναφέρεται και παραπάνω, είναι στην static παράμετρο που χρησιμοποιείται για την εγγραφή στην βάση. Η παράμετρος αυτή θα έχει **πάντα** την τιμή **0** για τα **έσοδα** και την τιμή **1** για τα **έξοδα**.

3.3 TOTAL FRAGMENT

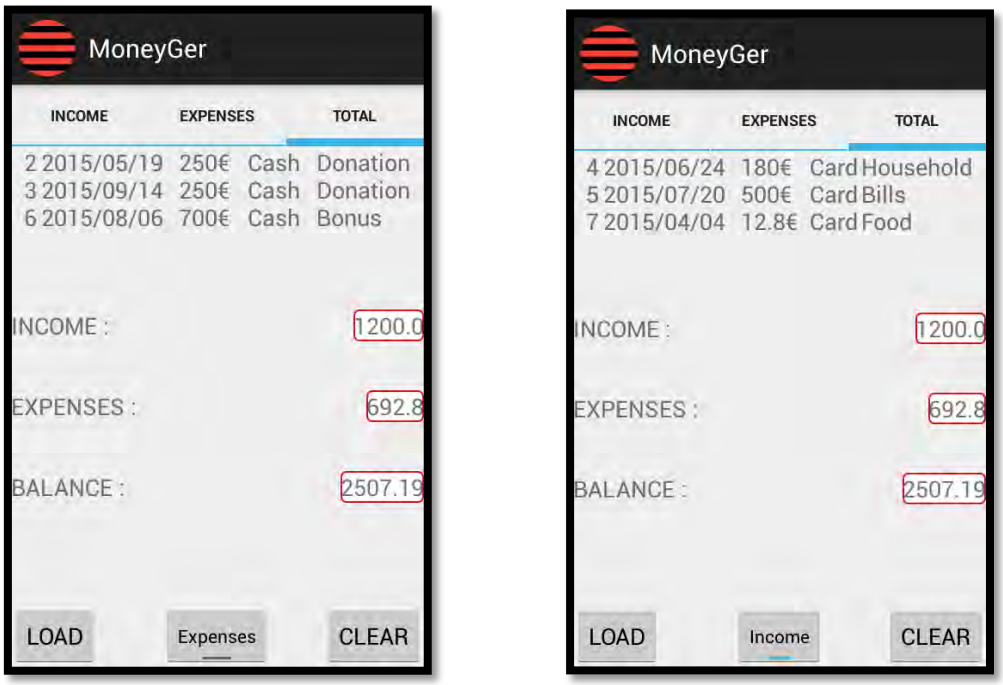
Το τρίτο και τελευταίο tab είναι αυτό που μας δείχνει μια συνολική εικόνα των συναλλαγών καθώς επίσης και τα συνολικά έσοδα, έξοδα και το υπόλοιπο που απομένει. Περιλαμβάνει επίσης ένα TextView που είναι scrollable και εμφανίζει όλες τις συναλλαγές του χρήστη ταξινομημένες κατά ημερομηνία, ένα toggle button που είναι ένα widget button το οποίο και χρησιμοποιείται για την εναλλαγή της εμφάνισης των συναλλαγών εσόδων και των συναλλαγών εξόδων στο TextView, ένα κουμπί φόρτωσης των συναλλαγών Load και ένα κουμπί εκκαθάρισης του TextView Clear. Τα παραπάνω παρουσιάζονται στην Εικόνα [3.8] ενώ ο xml κώδικας που τα υλοποιεί παρέχεται στην συνέχεια.

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.dimitris.moneyger.TotalFragment">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:maxLines="5"
        android:scrollbars="vertical"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:id="@+id/loadText"
        android:layout_gravity="center_horizontal|top" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/loadButton"
        android:id="@+id/loadButton"
        android:layout_gravity="left|bottom" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/clearButton"
        android:id="@+id/clearButton"
```

```
    android:layout_gravity="right|bottom" />

<ToggleButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/toggleButton"
    android:id="@+id/toggleButton"
    android:textOff="@string/textOff"
    android:textOn="@string/textOn"
    android:layout_gravity="center_horizontal|bottom" />

<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="150dp"
    android:layout_gravity="center">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="@string/totalIncome"
        android:id="@+id/textViewInc"
        android:layout_gravity="left|top" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="@string/totalExpenses"
        android:id="@+id/textViewExp"
        android:layout_gravity="left|center_vertical" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="@string/totalBalance"
        android:id="@+id/textViewBal"
        android:layout_gravity="left|bottom" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="@string/totalTextInc"
        android:background="@drawable/back"
        android:id="@+id/textViewIncomeBox"
        android:layout_gravity="right|top" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="@string/totalTextExp"
        android:background="@drawable/back"
        android:id="@+id/textViewExpensesBox"
        android:layout_gravity="right|center_vertical" />
```

```
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="@string/totalTextBal"
            android:background="@drawable/back"
            android:id="@+id/textViewBalanceBox"
            android:layout_gravity="right|bottom" />

    </FrameLayout>

</FrameLayout>
```

Εικόνα 3.8: Απεικόνιση Εσόδων και Εξόδων



Το κυριότερο σημείο της υλοποίησης του **TotalFragment** είναι η εμφάνιση των Εσόδων. Τα έσοδα προέρχονται τόσο από τα δεδομένα που εισάγει ο χρήστης στο αντίστοιχο tab όσο και από τα μηνιαία έσοδα που προστίθενται στον λογαριασμό του κάθε πρώτη του μήνα. Στην συνέχεια παρουσιάζεται ο κώδικας υπολογισμού εσόδων και θα αναλυθεί ένα **bug** που προέκυψε και χρειάστηκε να αντιμετωπιστεί.

```
/**Getting only the day from the current date to use it as a check in order to
 * determine if the month has changed (and it is the first day of the new month) and
add the
 * salary to the total income.
 */
Calendar calendarView = Calendar.getInstance();
String formattedDate = String.valueOf(calendarView.get(Calendar.DAY_OF_MONTH));
String currentMonth   = String.valueOf(calendarView.get(Calendar.MONTH) + 1);



//Income textView box.
totalInc = handler.totalIncome();
//System.out.println("TOTAL INCOME -----> " +totalInc);


//Having the income amount displayed allowing two decimals.
totalInc = twoDecimals.roundTwoDecimals(totalInc);

//Setting the view of the TextView to display the current income.
incomeView.setText(String.valueOf(totalInc));


//Getting the month of the current Calendar instance.
SimpleDateFormat df = new SimpleDateFormat("M");
String dateSTR = df.format(calendarView.getTime());

//Getting the month's integer value.
date = Integer.parseInt(dateSTR);

//The counter is saved into a table in the database so it's value IS NOT LOST. In the
//beginning counter's value is set to 0 showing it's the first time we access this
PART
//of the fragment.
handler.saveCounter(counter,date);

//If this is the FIRST day of the month counter is increased and it's NEW value is
saved in
//the database using the same method as before.
if (formattedDate.equals("1")){
    counter++;
    String str = handler.counterToString();
    handler.saveCounter(counter,date);

    //This is a variable used to show if there is a new month.To wit it is used in
order to
    //see if the month has changed. Using the checkRepeat method,which takes the
INTEGER
    //value of the month(current) as a parameter, we either get -1 if the months in
the db
    //are all <= with the current or a value different to -1 if there is even one
month in
    //the database whose value(integer) is > than the current month.
    check = handler.checkRepeat(Integer.parseInt(currentMonth));


    /**
     * MAJOR BUG FIX: Without the implementation below, every time the user changed
view in
     * the fragment, say from Total Fragment to Income Fragment or simply anywhere
```

```
else, then
     * the income would increase by the salary he had stored in the Settings Fragment.
Now
     * the salary is ONLY changed when the month is changed and is it's FIRST DAY.
     */

    if (check != -1 && (handler.getCounter(date) <= 1 && handler.getCounter(date) >
0)){
        totalSal = handler.getSalary();
        handler.totalSaveRepeat(totalSal);

    }


}
totalSal = totalInc + handler.gettotalSaveRepeat();
```

Το bug προερχότανε από το γεγονός ότι κάθε φορά που ο χρήστης άλλαζε view (για παράδειγμα από το view του TotalFragment μετέβαινε στο IncomeFragment) και τύχαινε να είναι πρώτη του μηνός το ποσό των συνολικών εσόδων αυξάνονταν κατά του μηνιαίο ποσό χωρίς να περιορίζονταν στην μία φορά όπως είναι το ορθό. Η λύση βασίζεται σε τρία βήματα.

- Πρώτον, χρησιμοποιείται ένας μετρητής ο οποίος αποθηκεύει σε έναν ειδικό πίνακα της βάσης την τιμή του και την ημερομηνία η οποία δεν είναι πλήρης και κρατάει μόνο την ακέραια τιμή του τρέχοντος μήνα**(ουσιαστικά μόνο ο μήνας μας χρειάζεται καθώς και η αποθήκευση του μηνιαίου ποσού από τον χρήστη γράφει μόνο τον μήνα στην βάση όπως θα αναλυθεί και σε ερχόμενο κεφάλαιο).** Η αποθήκευση στην βάση των παραπάνω γίνεται μέσω της μεθόδου **saveCounter**.

- Δεύτερον, εάν είναι η πρώτη μέρα του μήνα η τιμή του counter αυξάνεται **και η νέα τιμή αποθηκεύεται στην βάση.** Επίσης χρησιμοποιείται η μέθοδος **checkRepeat** η οποία και δέχεται ως παράμετρο τον τρέχον μήνα. Η μέθοδος αυτή, ελέγχει τον πίνακα στον οποίο ο χρήστης αποθηκεύει το μηνιαίο του ποσό μαζί με την ημερομηνία εισαγωγής (μόνο την τιμή του μήνα όπως αναφέραμε προηγουμένως) **και** όσα δεδομένα εισάγει στο IncomeFragment. Αν βρεθεί μήνας ο οποίος έχει αριθμητική τιμή **μικρότερη** από την αντίστοιχη του μήνα που περνάμε ως παράμετρο τότε αυτό σημαίνει ότι ο μήνας έχει αλλάξει και επιστρέφεται μια τιμή διάφορη του -1.

- Τρίτον, ελέγχεται μέσω της συνάρτησης **getCounter** ο πίνακας στον οποίο αποθηκεύεται η τιμή του μετρητή και η τιμή του μήνα από το πρώτο βήμα. Επιλέγονται όλες οι τιμές(entries του πίνακα) οι οποίες αντιστοιχούν σε έσοδα και έχουν ημερομηνία (μήνα) ίδιο με τον τρέχοντα. Αν το πλήθος τους είναι ίσο με την μονάδα **και επιπλέον** από το δεύτερο βήμα επιστρέφεται τιμή διάφορη του -1 τότε το **συνολικό ποσό** (εισαχθέντα δεδομένα και μηνιαίο ποσό) αποθηκεύεται στον πίνακα του βήματος 2 με χρήση της μεθόδου **totalSalRepeat**. Τέλος το συνολικό ποσό που έχει υπολογιστεί εμφανίζεται ως συνολικά έσοδα στο αντίστοιχο πλαίσιο.

Με τον τρόπο αυτό όσες φορές και αν αλλάξει ο χρήστης το view κάθε πρώτη του μήνα, το συνολικό ποσό θα υπολογίζεται αυστηρά μία φορά μόνο. Αξίζει να αναφερθούμε στο γεγονός ότι το υπόλοιπο υπολογίζεται και με βάσει το μηνιαίο ποσό που πιθανόν να εισάγει ο χρήστης. Επίσης, το ποσό αυτό θα είναι διαθέσιμο και θα εμφανίζεται στα συνολικά έσοδα και στο υπόλοιπο τον **επόμενο** μήνα από τον οποίο το εισάγει ο χρήστης εκτός εάν το ποσό εισαχθεί πρώτη του μηνός. Για παράδειγμα, αν ο τρέχον μήνας είναι ο Ιούνιος και **δεν** έχουμε 1/6 τότε αν ο χρήστης επιλέξει να εισάγει ένα μηνιαίο ποσό αυτό θα είναι διαθέσιμο την 1/7, δηλαδή την πρώτη του επόμενου μήνα. Αν, όμως, έχουμε 1/6 και ο χρήστης εισάγει ένα μηνιαίο ποσό τότε αυτό θα είναι **άμεσα διαθέσιμο** για χρήση.

Με τα παραπάνω κλείνει το κεφάλαιο που αναφέρεται στο HomeFragment και το οποίο αποτελεί τον βασικό κορμό της εφαρμογής.

Κεφάλαιο 4

4 SETTINGS FRAGMENT

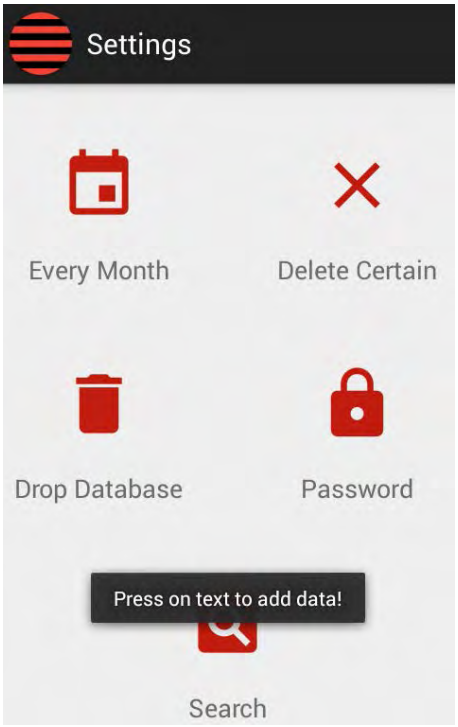
Ένα επόμενο βασικό μέρος της παρούσας εφαρμογής αποτελούν οι ρυθμίσεις του συστήματος, ο κώδικας υλοποίησης των οποίων συνοψίζεται στο αρχείο SettingsFragment.java και στα fragment_settings.xml, prompts.xml, password.xml, confirm_drop_db.xml τα οποία και θα αναλυθούν εν συνεχεία.

Στο μενού αυτό, στο οποίο ο χρήστης μπορεί να οδηγηθεί(όπως έχει προαναφερθεί) μέσω του «συρταριού» navigation drawer, υποστηρίζονται λειτουργίες όπως:

- η εισαγωγή του μηνιαίου ποσού,
- η διαγραφή μεμονωμένων συναλλαγών με χρήση του αναγνωριστικού τους,
- διαγραφή όλων των αποθηκευμένων ρυθμίσεων και συναλλαγών απευθείας,
- δημιουργία και ανανέωση κωδικού
- εύρεση όλων των συναλλαγών ανάμεσα σε μία αρχική και μία τελική ημερομηνία.

Κάθε υποενότητα που θα ακολουθήσει θα περιγράψει αναλυτικά τον τρόπο υλοποίησης, τόσο τον αλγοριθμικό όσο και τον σχεδιαστικό, για κάθε μία από τις παραπάνω λειτουργίες αρχής γενομένης της εισαγωγής του μηνιαίου ποσού. Στην Εικόνα [4.1] παρέχεται μία οπτική αναπαράσταση του μενού που θα δει ο χρήστης.

Αξίζει να αναφερθεί ότι τα εικονίδια του παραπάνω μενού εμφανίζονται με χρήση **animation** που παρέχεται από την γλώσσα προγραμματισμού δημιουργώντας ένα απλό αλλά αισθητικό οπτικό εφέ ο κώδικας του οποίου θα είναι διαθέσιμος μαζί με τον κώδικα ολόκληρου του αρχείου.



Εικόνα 4.1: Settings Fragment

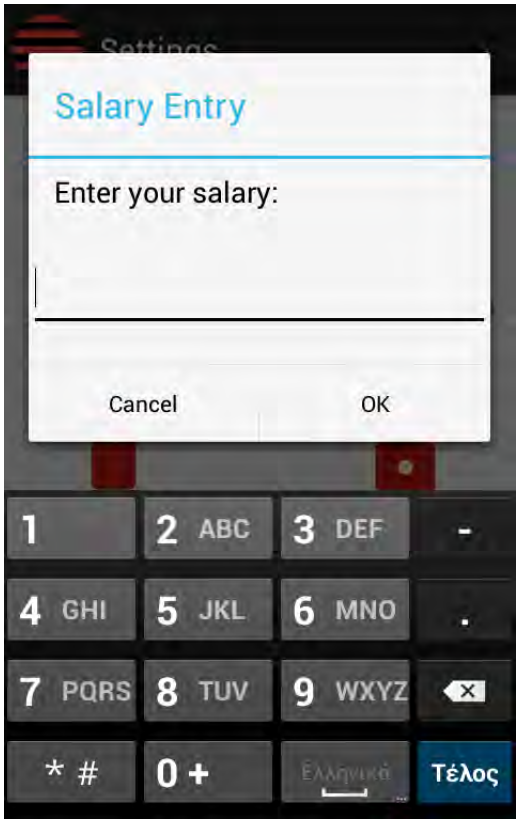
4.1 Εισαγωγη Μηνιαιου Ποσου

Εάν ο χρήστης επιλέξει να εισάγει ένα μηνιαίο ποσό, τότε χρησιμοποιεί το πρώτο εικονίδιο το οποίο και ενεργοποιεί ένα dialog box εισαγωγής του ποσού ανάλογο με αυτό της Εικόνας [4.2]. Όπως αναφέρθηκε και στο Κεφάλαιο 3 μαζί με την εισαγωγή του ποσού αποθηκεύεται στον αντίστοιχο πίνακα στην βάση και η ακέραια τιμή του τρέχοντος μήνα. Το πληκτρολόγιο που χρησιμοποιείται στο εν λόγω dialog box είναι το ίδιο με όλα τα υπόλοιπα που χρησιμοποιούνται για την εισαγωγή κάποιου ποσού στην εφαρμογή.

Το ποσό που θα εισάγει ο χρήστης ελέγχεται για το προαναφερθέν **bug** που προέρχεται από την χρήση ελληνικού πληκτρολογίου, μετατρέπεται σε Double με ακρίβεια δύο δεκαδικά ψηφία και εφόσον όλα είναι εντάξει καλείται η μέθοδος **insertMonthly**.

```
//Function to automatically insert the given amount in the db.
public void insertMonthly(double salary){
    Calendar = Calendar.getInstance();
    SimpleDateFormat df = new SimpleDateFormat("M");
```

```
    String formattedDate = df.format(calendar.getTime());
    myDBHandler.saveSalary(salary, formattedDate);
}
```



Εικόνα 4.2: Εισαγωγή Μηνιαίου Ποσού

Όπως φαίνεται και στον παραπάνω κώδικα, η μέθοδος insertMonthly δέχεται ως παράμετρο **μόνο** το ποσό που εισάγει ο χρήστης. Επιτελεί μία επιπλέον λειτουργία όμως. Μόλις κληθεί, υπολογίζει και κρατάει την ακέραια αριθμητική τιμή του τρέχοντος μήνα, και καλεί την συνάρτηση **saveSalary** της κλάσης **dbHandler** για να πραγματοποιήσει την αποθήκευση στον πίνακα της βάσης σύμφωνα με τα όσα έχουμε ήδη αναφέρει. Ο κώδικας που υλοποιεί η μέθοδος **saveSalary** παρουσιάζεται στην συνέχεια:

```
public void saveSalary(double salary,String date){

    SQLiteDatabase db = getWritableDatabase();

    String query = "INSERT INTO " + TABLE_TOTALSAL +
            " (" + COLUMN_SALARY + " , " + COLUMN_DATE +
            " ) VALUES('"+salary+"','"+date+"');";

    Cursor c  = db.rawQuery(query,null);
    c.moveToFirst();
```

```
    c.close();
    db.close();
}
```

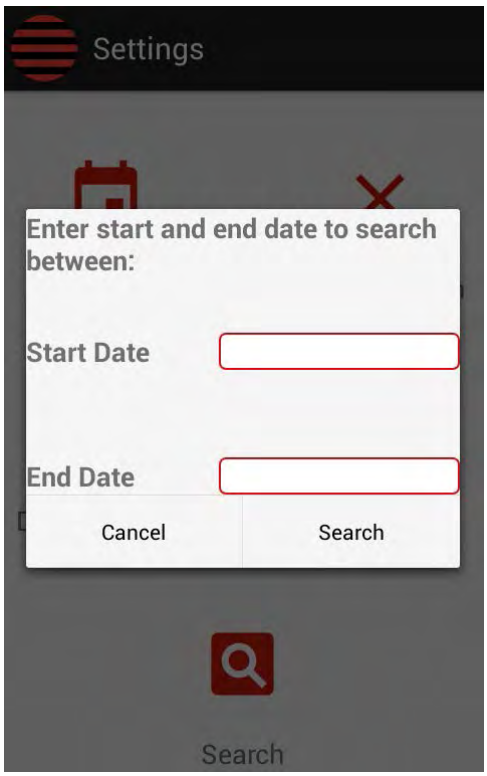
Παρατίθεται στην συνέχεια και ολόκληρος ο κώδικας για την μέθοδο **insertMonthly:**

```
public void monthDialog(){

    dialogBuilder = new AlertDialog.Builder(getActivity());
    dialogBuilder.setTitle(R.string.salaryEntryTitle);
    dialogBuilder.setMessage(R.string.salary_enter);
    dialogBuilder.setCancelable(false);

    final EditText input  =  new EditText(getView().getContext());

    input.setInputType(InputType.TYPE_CLASS_PHONE);
    dialogBuilder.setView(input);

    dialogBuilder.setPositiveButton(R.string.ok, new
DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
            String salarySTR = input.getEditableText().toString();

            if  (salarySTR.contains(",")){
                salarySTR = salarySTR.replace(",",".");
            }

            salary_entry = Double.parseDouble(salarySTR);
            salary_entry = twoDecimals.roundTwoDecimals(salary_entry);

            insertMonthly(salary_entry);

        }
    });
    dialogBuilder.setNegativeButton(R.string.cancel, new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.cancel();
        }
    });

    AlertDialog alert = dialogBuilder.create();
    alert.show();
}


//Function to automatically insert the given amount in the db.
public void insertMonthly(double salary){
    Calendar calendar = Calendar.getInstance();
    SimpleDateFormat df = new SimpleDateFormat("M");
    String formattedDate = df.format(calendar.getTime());
```

```
    myDBHandler.saveSalary(salary, formattedDate);
}
```

4.2 ΕΥΡΕΣΗ ΣΥΝΑΛΛΑΓΩΝ ΜΕ ΒΑΣΗ ΑΡΧΙΚΗ ΚΑΙ ΤΕΛΙΚΗ ΗΜΕΡΟΜΗΝΙΑ

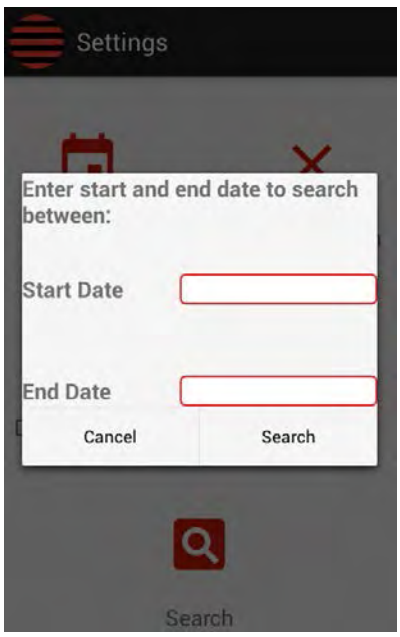
Η επόμενη λειτουργία που υποστηρίζεται είναι αυτή της εύρεσης όλων των συναλλαγών που υπάρχουν ανάμεσα σε μία αρχική και μια τελική ημερομηνία που επιλέγει ο χρήστης. Καταρχάς το dialog box που δημιουργείται μόλις ο χρήστης επιλέξει το εικονίδιο εύρεσης παρουσιάζεται στην Εικόνα [4.3].

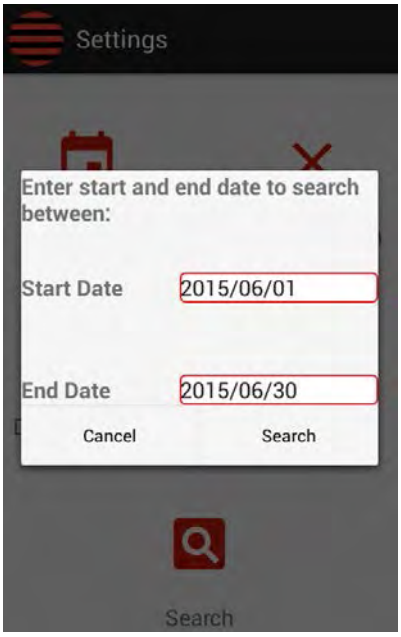


Εικόνα 4.3: Εύρεση Συναλλαγών

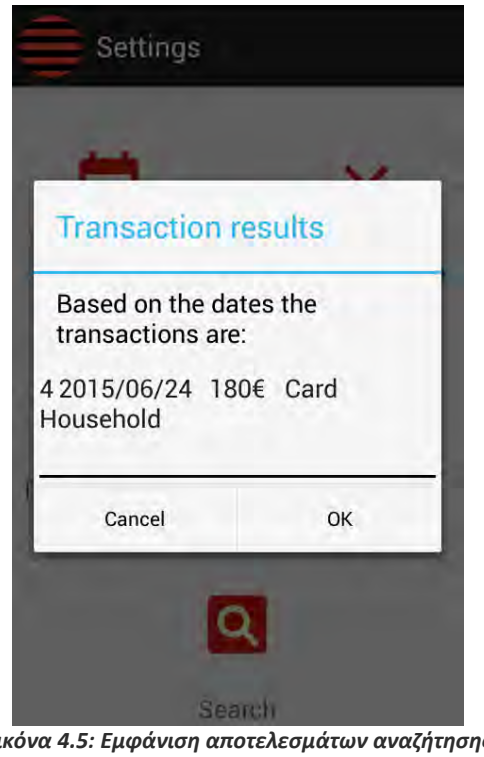
Παρατηρούμε ότι και πάλι πρόκειται για το καθιερωμένο dialog box με ένα positive (search) και ένα negative (cancel) κουμπί καθώς επίσης και δύο TextView widgets και δύο EditText widgets. Η εισαγωγή δεδομένων (δηλαδή της αρχικής και της τελικής ημερομηνίας) ακολουθεί το μοτίβο κάθε άλλης εισαγωγής δεδομένων στην εφαρμογή, χρησιμοποιώντας, δηλαδή, τα TextViews. Μόλις ο χρήστης πατήσει στο **Start Date** εμφανίζεται ένα νέο dialog box από το οποίο επιλέγει μια αρχική η ημερομηνία. Όμοια και για το **End Date**. Τα dialog boxes είναι ακριβώς ίδια με αυτά που χρησιμοποιούνται και στην εισαγωγή ημερομηνίας στα **IncomeFragment** και **ExpensesFragment**. Στην Εικόνα [4.4] παρουσιάζεται οπτικά το αποτέλεσμα εισαγωγής ημερομηνιών από τον χρήστη.

Εικόνα 4.4 Πριν και μετά την εισαγωγή ημερομηνιών





Αφού εισαχθούν οι ημερομηνίες ο χρήστης είναι σε θέση να δει τις πιθανές (καθώς μπορεί να μην υπάρχουν συναλλαγές οπότε και εμφανίζεται κατάλληλο μήνυμα) συναλλαγές που πραγματοποιήθηκαν μεταξύ των δύο αυτών ημερομηνιών. Το dialog box που θα δημιουργηθεί με το πάτημα του Positive Button Search παρουσιάζεται στην Εικόνα [4.5].



Εικόνα 4.5: Εμφάνιση αποτελεσμάτων αναζήτησης

Στη συνέχεια παρέχεται ο κώδικας που υλοποιεί την αναζήτηση στην βάση για την εύρεση των πιθανών συναλλαγών και την εμφάνιση-δημιουργία των dialog boxes που χρησιμοποιούνται για την είσοδο των δεδομένων και την παρουσίαση των αποτελεσμάτων.

```
public void findTransDialog() {

    dialogBuilder = new AlertDialog.Builder((getActivity()));
    dialogBuilder.setView(promptsView);
    dialogBuilder.setCancelable(false);

    //Setting the onclick actions for the text views.
    dateStart.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            new DatePickerDialog(getActivity(), dateST, myCalendar
                    .get(Calendar.YEAR), myCalendar.get(Calendar.MONTH),
                    myCalendar.get(Calendar.DAY_OF_MONTH)).show();
        }
    });

    dateEnd.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            new DatePickerDialog(getActivity(), dateEN, myCalendar
                    .get(Calendar.YEAR), myCalendar.get(Calendar.MONTH),
                    myCalendar.get(Calendar.DAY_OF_MONTH)).show();
        }
    });


    /**
     * MAJOR BUG FIXED: When the dialog box is destroyed either by cancel or
by search
     * it's parent still has this child as it is loaded in the onCreated
method,so if
     * --remaining in the settings fragment(parent)-- one pressed the Search
icon for
     * the second time that would lead the app to stop working. Thus when the
dialog is
     * closed the promptsView that it uses is removed FROM THE PARENT.
     */

    dialogBuilder.setPositiveButton(R.string.search, new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {

            if (dateSTedit.getText().toString().equals("") ||
dateENedit.getText().
                    toString().equals("")){
                Toast.makeText(getActivity(),R.string.fields_required,
Toast.LENGTH_SHORT)
                        .show();
                dateSTedit.setText("");
```

```
                dateENedit.setText("");
            }

            /**
             * Need the check if the start date picked is greater than the
ending one. If not it
             * may cause serious trouble when searching in the database
causing the application
             * to stop.Splitting the string from the editText threeways to
get a value for the
             * day,the month and the year.
             */

            else{
                start_date_str = dateSTedit.getText().toString();
                end_date_str   = dateENedit.getText().toString();

                //Year.
                start_year = Integer.parseInt(start_date_str.substring(0,4));
                end_year   = Integer.parseInt(end_date_str.substring(0,4));


                //Month.
                start_month =
Integer.parseInt(start_date_str.substring(5,7));
                end_month   = Integer.parseInt(end_date_str.substring(5,7));


                //Day.
                start_day = Integer.parseInt(start_date_str.substring(8,10));
                end_day   = Integer.parseInt(end_date_str.substring(8,10));


                if (start_year > end_year){
                    Toast.makeText(getActivity(),R.string.year_error
                            ,Toast.LENGTH_SHORT).show();
                }
                else{
                    if (start_month == end_month){
                        //same_month = 1;
                    }
                    else{
                        if (start_month > end_month){
                            Toast.makeText(getActivity(),R.string.month_error
                                    ,Toast.LENGTH_SHORT).show();
                        }
                    }
                }



//*************************************************************************
***
                //      Method to check the database for transactions between
the two dates.

                /**MAJOR BUG FIXED:
```

```
                *If the user used e.g 1/1/2015 as a start date and
30/1/2015 as
                *an end date then SQL call would display all results from
the database,inlcu-
                *ding results that relate to OTHER MONTHS. This is because
of the SQLite itse-
                *lf which supports every date format BUT works properly
only with the
                *yyyy/mm/dd one.When i changed the format WRITTEN in the
database(in Income-
                * Fragment and in ExpensesFragment) everything worked as
supposed.
                */

                transaction_exists =
myDBHandler.transactionsExistBetweenDates(dateSTedit.

getText().toString(),dateENedit.getText().toString());

                if (transaction_exists > 0 ){
                    check =
myDBHandler.searchTransaction(dateSTedit.getText().toString(),
                            dateENedit.getText().toString());
                    transBetweenDatesDialog(check);
                }
                else{
                    Toast.makeText(getActivity(),R.string.trans_existence,
                            Toast.LENGTH_SHORT).show();
                }


//*************************************************************************
***


                /**
                 * After the dialog is closed either by cancel or by search
click,all the fields
                 * it contains must become empty again in order to begin a
new search if a user
                 * wants to.
                 */
                dateSTedit.setText("");
                dateENedit.setText("");
            }

            ((ViewGroup) promptsView.getParent()).removeView(promptsView);
        }
    });
    dialogBuilder.setNegativeButton(R.string.cancel, new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dateSTedit.setText("");
            dateENedit.setText("");
            dialog.cancel();
            ((ViewGroup)promptsView.getParent()).removeView(promptsView);
```

```
        }
    });

    AlertDialog alert = dialogBuilder.create();
    alert.show();
}
```

Με βάση τον παραπάνω κώδικα θα πρέπει να σταθούμε σε τρία κυρίως σημεία του. Καταρχάς, πρέπει να αναφέρουμε ότι όλα τα dialog boxes που χρησιμοποιούνται είναι **παιδιά** τα οποία έχουν ως γονέα το βασικό View που ανήκουν (στην προκειμένη περίπτωση το view του **SettingsFragment**). Όταν το παιδί «καταστρέφεται», είτε επειδή ο χρήστης πάτησε το negative κουμπί της ακύρωσης είτε το positive κουμπί της ολοκλήρωσης, δεν καταστρέφεται τελείως αλλά ο γονιός του εξακολουθεί να το θεωρεί φορτωμένο. Ως αποτέλεσμα, εάν μετά το κλείσιμο του dialog box ο χρήστης προσπαθούσε να το επαναχρησιμοποιήσει θα είχαμε εμφάνιση crash στην εφαρμογή. Για να αντιμετωπιστεί αυτό το **bug** θα πρέπει το παιδί να αφαιρείται τελείως από τον γονιό του κατά το κλείσιμο κάτι που επιτυγχάνεται με τον εξής κώδικα:

```
((ViewGroup) promptsView.getParent()).removeView(promptsView);
((ViewGroup)promptsView.getParent()).removeView(promptsView);
```

Ο παραπάνω κώδικας αφαιρεί το view του παιδιού- που είναι το prompts view- από τον γονιό του –που υπολογίζεται με χρήση της μεθόδου **getParent** της κλάσης **ViewGroup**- πάνω στο παιδί.

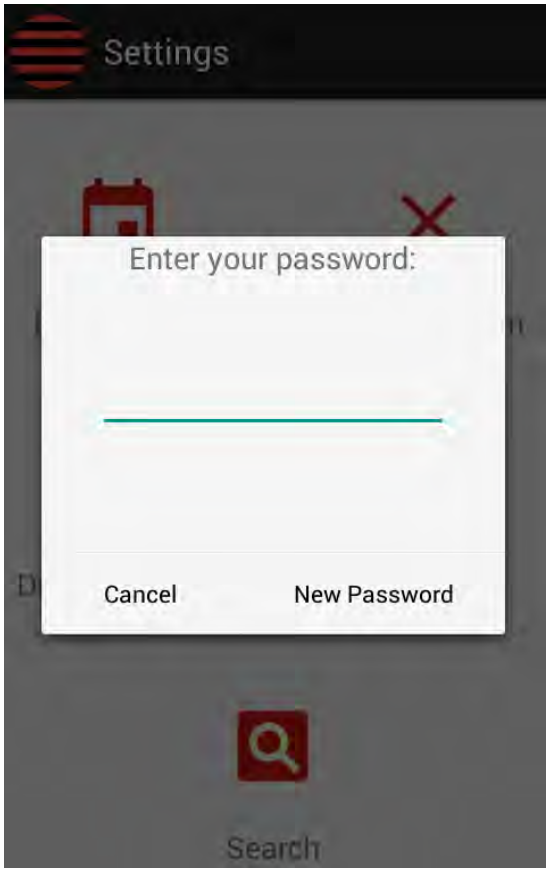
Ένα δεύτερο σημείο άξιο παρατήρησης είναι η εμφάνιση αποτελεσμάτων για συναλλαγές στις οποίες η αρχική και τελική ημερομηνία αντιστοιχούν στον ίδιο μήνα (όπως φαίνεται και στην Εικόνα [4.4]). Στο Κεφάλαιο 3 αναφέρεται ότι το format της ημερομηνίας που αποθηκεύεται στην βάση πρέπει να είναι το συγκεκριμένο αλλιώς μπορούσαν να προκύψουν προβλήματα κατά την **ανάγνωση** από αυτή. Εδώ είναι που παρατηρείται το δεύτερο **bug** το οποίο αντιμετωπίστηκε. Εάν η ημερομηνία δεν είχε το σωστό format τότε η αναζήτηση στην βάση, ας πούμε για παράδειγμα από 1/6/2015 έως 30/6/2015, θα εμφάνιζε και αποτελέσματα τα οποία δεν ανήκουν στο πλαίσιο αυτό. Συμμορφώνοντας, όμως, τον κώδικα στις απαιτήσεις της SQLite το **bug** επιλύθηκε και η αναζήτηση εμφανίζει πάντα τα σωστά αποτελέσματα.

Τέλος, αξίζει να αναφερθούμε και στους ελέγχους που λαμβάνουν χώρα ώστε να διασφαλίζεται σε κάθε περίπτωση η λογική συνέπεια στις δύο ημερομηνίες που εισάγονται. Για παράδειγμα, θα μπορούσε κάποιος χρήστης να βάλει την ημερομηνία 2015/6/30 ως την αρχική και την ημερομηνία 2015/6/1 ως την τελική παραβιάζοντας την λογική συνέπεια που προϋποθέτει ότι η ημερομηνία έναρξης θα πρέπει να είναι μικρότερη ή ίση από την ημερομηνία κατάληξης. Επομένως, ξεκινώντας από τον έλεγχο του έτους, προχωράμε μέχρι και τον μήνα εμφανίζοντας μήνυμα σφάλματος αν υπάρξει κάποια παραβίαση.

4.3 Δημιουργια και Διαχειριση Κωδικου

Οι δύο προηγούμενες υποενότητες που περιεγράφηκαν είναι και οι μόνες που δεν απαιτούν χρήση κωδικού στο SettingsFragment και για τον λόγο αυτό αναφέρθηκαν πρώτες. Για όλες τις υπόλοιπες λειτουργίες η εισαγωγή κωδικού είναι απαραίτητη καθώς περιλαμβάνουν την διαγραφή δεδομένων από την εφαρμογή(και επομένως μη εξουσιοδοτημένη είσοδος μπορεί να είναι επικίνδυνη).

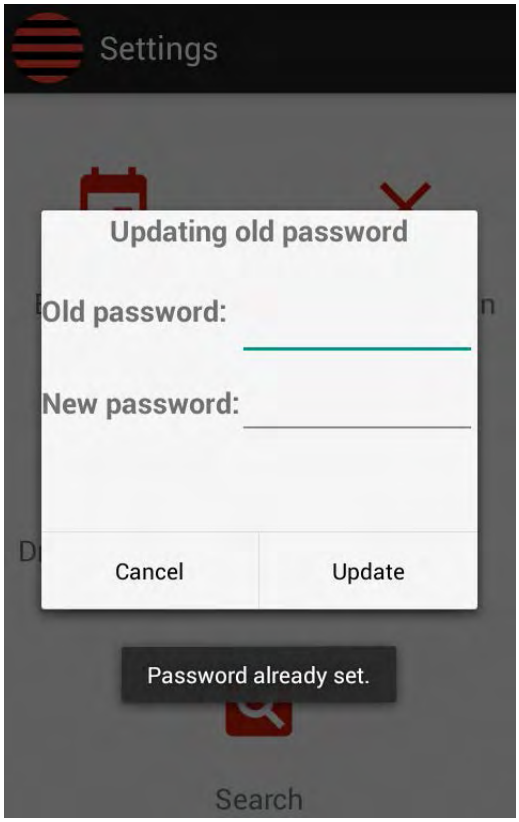
Καταρχάς, για να εισάγει έναν νέο κωδικό ο χρήστης πρέπει να επιλέξει το αντίστοιχο εικονίδιο το οποίο με την σειρά του καλεί την συνάρτηση **passwordDialog** που είναι υπεύθυνη για την δημιουργία των dialog boxes που σχετίζονται με την δημιουργία του νέου και την διαχείριση του παλιού κωδικού. Για την εισαγωγή του κωδικού χρησιμοποιείται το dialog box που παρουσιάζεται στην Εικόνα [4.6].



Εικόνα 4.6: Δημιουργία Κωδικού

Σε περίπτωση που ο κωδικός είναι ο πρώτος που εισάγει ο χρήστης θα εμφανιστεί ένα μήνυμα επιτυχούς αποθήκευσης. Αν από την άλλη πλευρά, είτε ο ίδιος ο χρήστης είτε κάποιος άλλος προσπαθήσει να εισάγει κωδικό την στιγμή που ήδη υπάρχει ένας σε ισχύ τότε θα εμφανιστεί

ένα μήνυμα που θα αναφέρει ότι ο κωδικός έχει ήδη κατοχυρωθεί και ένα dialog box που θα δίνει την δυνατότητα για αλλαγή του παλιού. Το νέο dialog box παρουσιάζεται στην Εικόνα [4.7].



Εικόνα 4.7: Διαχείριση Παλιού Κωδικού

Εφόσον ο παλιός κωδικός ταιριάζει με αυτόν στην βάση τότε ο χρήστης μπορεί να τον ανανεώσει με κάποιον καινούργιο. Ο έλεγχος αυτός επιτυγχάνεται μέσω του παρακάτω κώδικα:

```
public void checkOldPasswordDialog() {

    dialogBuilder = new AlertDialog.Builder(getActivity());
    dialogBuilder.setView(confirmView);
    dialogBuilder.setCancelable(false);

    dialogBuilder.setPositiveButton(R.string.update, new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {

            old_password_entered = edit_old.getText().toString();
            new_password_entered = edit_new.getText().toString();
            password_equality =
myDBHandler.passwordChecking(old_password_entered);
```

```
            if (password_equality > 0) {
                myDBHandler.updatePasswordToDb(new_password_entered);
                Toast.makeText(getActivity(), R.string.update_success,
Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(getActivity(), R.string.password_inequality,
Toast.LENGTH_SHORT)
                        .show();
            }

            edit_old.setText("");
            edit_new.setText("");

            dialog.cancel();
            ((ViewGroup) confirmView.getParent()).removeView(confirmView);

        }

    });

    dialogBuilder.setNegativeButton(R.string.cancel, new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            edit_old.setText("");
            edit_new.setText("");
            dialog.cancel();
            ((ViewGroup) confirmView.getParent()).removeView(confirmView);
        }
    });


    AlertDialog alert = dialogBuilder.create();
    alert.show();
}
```

Τα δεδομένα που εισάγονται από τον χρήστη στο πεδίο **Παλιός Κωδικός** του dialog box περνάνε ως όρισμα στην μέθοδο **passwordChecking** της κλάσης **dbHandler**. Μέσω της συνάρτησης αυτής ελέγχεται αν ο παλιός κωδικός ταιριάζει με αυτόν που βρίσκεται αποθηκευμένος στην βάση και επιστρέφει μια τιμή μεγαλύτερη του μηδενός αν δεν υπάρχει αναντιστοιχία. Στη συνέχεια καλείται η μέθοδος **updatePasswordToDb** της κλάσης **dbHandler** η οποία κάνει update την παλιά τιμή του κωδικού στην καινούργια. Ο παρακάτω κώδικας υλοποιεί την λειτουργικότητα της μεθόδου αυτής:

```
public void updatePasswordToDb(String password) {
    SQLiteDatabase db = getWritableDatabase();
    String query = "UPDATE " + TABLE_PASSWORD +
            " SET " + COLUMN_PASSWORD  + "=\"" + password + "\";";

    Cursor c  = db.rawQuery(query,null);
    c.moveToFirst();
```

```
    c.close();
    db.close();
}
```

Είναι απαραίτητο, για την κατανόηση του τρόπου λειτουργίας της παραπάνω μεθόδου, να αναλύσουμε το πώς πραγματοποιείται η εισαγωγή/αποθήκευση ενός **νέου** κωδικού στην βάση δεδομένων. Για διευκόλυνση παρατίθεται, πρώτα, ο κώδικας που επιτυγχάνει το παραπάνω και στην συνέχεια θα λάβει χώρα η ανάλυσή του:

```
public void passwordDialog(){
    dialogBuilder = new AlertDialog.Builder(getActivity());
    dialogBuilder.setView(passwordView);
    dialogBuilder.setCancelable(false);

    //Checking if a password is already stored in the database. This button
has two uses,first
    //if the password is not already stored in the db it creates a new one
and stores it in the
    //db.Secondly,if the password exists then it is used to update the
password to a new one.


    dialogBuilder.setPositiveButton(R.string.new_pass, new
DialogInterface.OnClickListener(){
        @Override
        public void onClick(DialogInterface dialog, int which) {

            existence = myDBHandler.passwordExistToDb();
            if (existence == 0) {

                if (password_set.getText().toString().equals("")){

Toast.makeText(getActivity(),R.string.fields_required,Toast.LENGTH_SHORT)
                            .show();
                }
                else {

myDBHandler.addPasswordToDb(password_set.getText().toString());
                    Toast.makeText(getActivity(), R.string.password_add,
Toast.LENGTH_SHORT)
                            .show();
                }
            }
            else{

Toast.makeText(getActivity(),R.string.password_exists,Toast.LENGTH_SHORT)
                        .show();
                checkOldPasswordDialog();
            }
            password_set.setText("");
```

```
            dialog.cancel();
            ((ViewGroup) passwordView.getParent()).removeView(passwordView);
        }
    });

    dialogBuilder.setNegativeButton(R.string.cancel, new
DialogInterface.OnClickListener(){
        @Override
        public void onClick(DialogInterface dialog, int which) {
            password_set.setText("");
            dialog.cancel();
            ((ViewGroup) passwordView.getParent()).removeView(passwordView);
        }
    });


    AlertDialog alert = dialogBuilder.create();
    alert.show();

}
```

Μελετώντας τον παραπάνω κώδικα διαπιστώνει κανείς δύο βασικά μέρη, τον έλεγχο για τυχόν ύπαρξη κωδικού και την αποθήκευση του κωδικού στην βάση. Για τον έλεγχο χρησιμοποιείται η κλήση της μεθόδου **passwordExistTodb** της κλάσης **dbHandler** η οποία επιστρέφει μια τιμή μεγαλύτερη του 0 εάν υπάρχει μία εισαγωγή στον πίνακα των κωδικών της βάσης. Ο κώδικας που εκτελεί είναι ο εξής:

```
public int passwordExistToDb(){
    int existence;

    SQLiteDatabase db = getWritableDatabase();
    String query = "SELECT * FROM " + TABLE_PASSWORD +
            " WHERE 1";

    Cursor c = db.rawQuery(query,null);
    c.moveToFirst();

    existence = c.getCount();

    c.close();
    db.close();

    return existence;
}
```

Αν διαπιστωθεί ότι το αποτέλεσμα που επιστρέφεται δεν είναι μεγαλύτερο από 0, τότε δημιουργείται το dialog box και το μήνυμα που παρουσιάστηκαν στην Εικόνα [4.7]. Αν από την

άλλη πλευρά, δεν υπάρχει κανένας κωδικός στην βάση καλείται η μέθοδος **addPasswordToDb** της κλάσης **dbHandler** η οποία αποθηκεύει τον κωδικό όπως φαίνεται παρακάτω:

```
public void addPasswordToDb(String password){
    SQLiteDatabase db = getWritableDatabase();
    String query = "INSERT INTO " + TABLE_PASSWORD +
            " (" + COLUMN_PASSWORD + " ) VALUES('"+password+"');";
    Cursor c  = db.rawQuery(query,null);
    c.moveToFirst();


    c.close();
    db.close();
}
```

Άρα την πρώτη φορά, ο χρήστης, θα εισάγει τον κωδικό του,

1. θα κληθεί η **passwordExistTodb**
2. θα επιστραφεί τιμή μεγαλύτερη από 0
3. θα κληθεί η **addPasswordToDb** που θα αποθηκεύσει τον κωδικό στην βάση.

Αν στη συνέχεια ο χρήστης προσπαθήσει να ξανά εισάγει κωδικό **passwordExistTodb** θα επιστρέψει τιμή μικρότερη ή ίση του μηδενός μην επιτρέποντας την εγγραφή. Παρατηρούμε, λοιπόν, ότι στον πίνακα του κωδικού, με τον παραπάνω τρόπο, περιορίζουμε τον αριθμό των εγγραφών σε **μία** διευκολύνοντας πάρα πολύ την διαχείρισή του μέσω της χρήσης της **updatePasswordToDb** μεθόδου(όπως προαναφέραμε).

4.4 Διαγραφη Όλων Των Δεδομενων Της Εφαρμογης

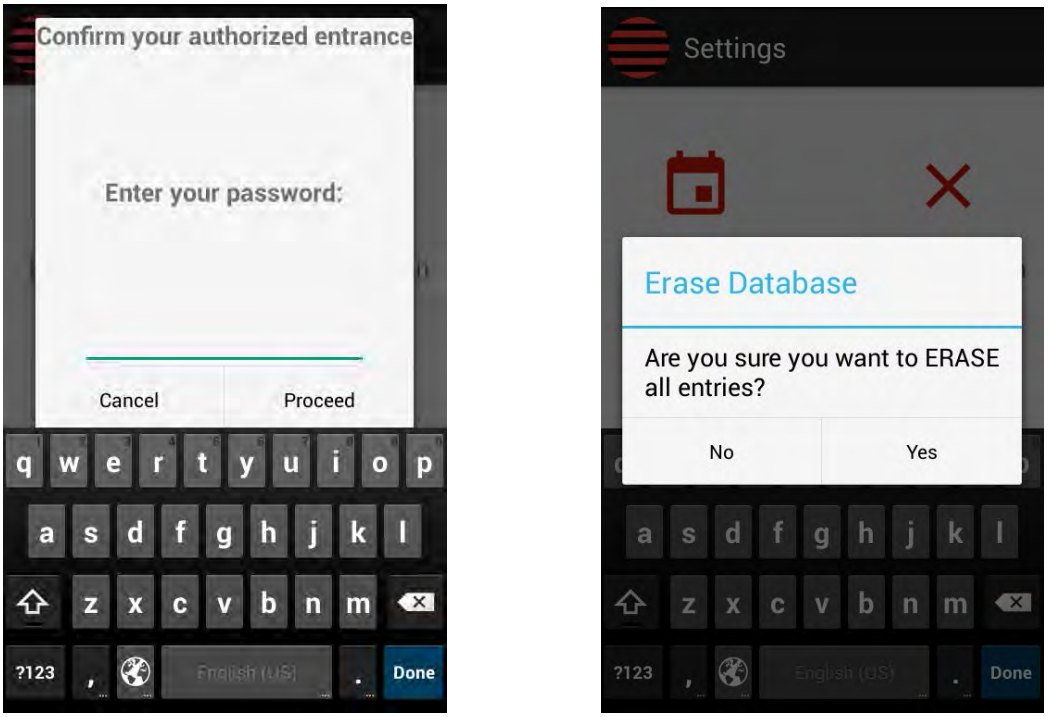
Αφού, ο χρήστης έχει επιλέξει έναν κωδικό, είναι πλέον σε θέση να διαγράψει όλα τα δεδομένα τα οποία έχει εισάγει. Αρχικά και πάλι εμφανίζεται ένα dialog box το οποίο ζητάει τον κωδικό εξουσιοδότησης που έχει οριστεί, τον ελέγχει και αν είναι ο σωστός προχωράει στην διαδικασία της διαγραφής. Επειδή πρόκειται για την μαζική διαγραφή όλων των δεδομένων, μετά από την επιτυχή αντιστοίχιση του εισαχθέντος κωδικού με τον ορισμένο στην βάση, εμφανίζεται και ένα επιπλέον dialog box το οποίο έχει ως στόχο να επιβεβαιώσει την επιλογή του χρήστη. Παρουσιάζεται, εν συνεχεία, ο κώδικας που υλοποιεί τα παραπάνω dialog boxes, τους ελέγχους και τελικά την διαγραφή των δεδομένων της βάσης.

```
private void clearDBDialog(){
    dialogBuilder = new AlertDialog.Builder(getActivity());
    dialogBuilder.setView(confirmDropView);
    dialogBuilder.setCancelable(false);

    dialogBuilder.setPositiveButton(R.string.proceed, new
```

```
DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
            drop_password = confirm_pass.getText().toString();

            password_equality = myDBHandler.passwordChecking(drop_password);
            if (password_equality > 0) {
                deleteConfirmedDialog();
            } else {
                Toast.makeText(getActivity(), R.string.password_inequality,
Toast.LENGTH_SHORT)
                        .show();
            }
            confirm_pass.setText("");
            dialog.cancel();

            ((ViewGroup)
confirmDropView.getParent()).removeView(confirmDropView);
        }
    });
    dialogBuilder.setNegativeButton(R.string.cancel, new
DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
            confirm_pass.setText("");
            dialog.cancel();

            ((ViewGroup)
confirmDropView.getParent()).removeView(confirmDropView);
        }
    });

    AlertDialog alert = dialogBuilder.create();
    alert.show();
}

//Drop all tables after confirmation.
private void deleteConfirmedDialog(){
    dialogBuilder = new AlertDialog.Builder(getActivity());
    dialogBuilder.setTitle(R.string.eraseTitle);
    dialogBuilder.setMessage(R.string.eraseConfirm);
    dialogBuilder.setCancelable(false);
    dialogBuilder.setPositiveButton(R.string.yes, new
DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
            myDBHandler.clearDB();
        }
    });
    dialogBuilder.setNegativeButton(R.string.no, new
DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
            dialog.cancel();
        }
    });

    AlertDialog alert = dialogBuilder.create();
    alert.show();
}
```

Τα στιγμιότυπα που βλέπει ο χρήστης κατά την αλληλεπίδρασή του με την παραπάνω λειτουργία συνοψίζονται στην Εικόνα [4.8].

Εικόνα 4.8: Έλεγχος εισόδου και μήνυμα επιβεβαίωσης



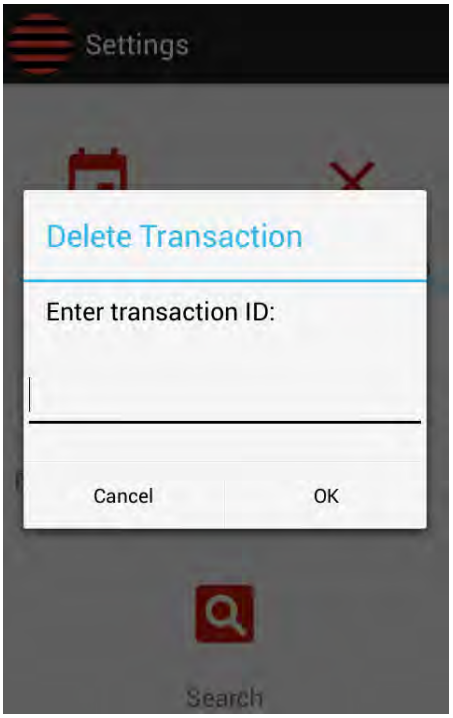
Ο κώδικας ο οποίος είναι υπεύθυνος για την διαγραφή των δεδομένων από την βάση υλοποιείται στην μέθοδο **clearDB** της κλάσης **dbHandler** και παρουσιάζεται στην συνέχεια.

```
public void clearDB(){
    SQLiteDatabase db = getWritableDatabase();
    String query  = "DELETE FROM " + TABLE_TRANSACTIONS + " WHERE 1";
    String query2 = "DELETE FROM " + TABLE_SAVESALARY + " WHERE 1";
    String query3 = "DELETE FROM " + TABLE_TOTALSAL + " WHERE 1";

    db.execSQL(query);
    db.execSQL(query2);
    db.execSQL(query3);
}
```

Πάλι αν κάποιος παρατηρήσει τον κώδικα που υλοποιεί την συγκεκριμένη λειτουργικότητα θα δει όλους τους ελέγχους και τα **bug fixes** για τα οποία μιλήσαμε διεξοδικά σε προηγούμενες υποενότητες.

4.5 ΔΙΑΓΡΑΦΗ ΣΥΓΚΕΚΡΙΜΕΝΗΣ ΣΥΝΑΛΛΑΓΗΣ

Τελευταία λειτουργία που παρέχεται από το **SettingsFragment** είναι αυτή της στοχευμένης διαγραφής συναλλαγών με χρήση του μοναδικού αριθμού που η κάθε μία διαθέτει. Αρχικά ζητείται, μέσα από ένα dialog box ίδιο με αυτό στην Εικόνα [4.8], από τον χρήστη να εισάγει τον κωδικό του. Εφόσον επιβεβαιωθεί ότι ο κωδικός είναι έγκυρος εμφανίζεται ένα νέο dialog box, όπως αυτό που παρουσιάζεται στην Εικόνα [4.9], το οποίο ζητάει από τον χρήστη να εισάγει το αναγνωριστικό της συναλλαγής που επιθυμεί να διαγράψει.



Εικόνα 4.9: Εισαγωγή αναγνωριστικού συναλλαγής προς διαγραφή

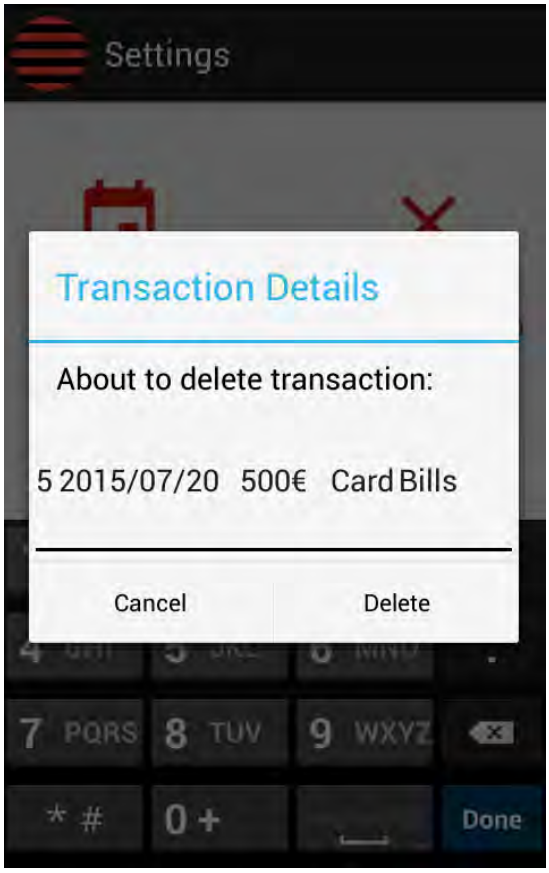
Αφού το εισάγει γίνεται έλεγχος για το κατά πόσο αυτό υφίσταται. Αν δεν υπάρχει εμφανίζεται μήνυμα αδυναμίας εύρεσης της συναλλαγής. Αν υπάρχει τότε εμφανίζεται ένα νέο dialog box το οποίο περιέχει τα στοιχεία της συναλλαγής που πρόκειται να διαγραφεί. Το dialog box αυτό αναπαρίσταται στην Εικόνα [4.10]. Εφόσον είναι η συναλλαγή που επιθυμεί ο χρήστης να διαγράψει από την βάση, προχωράει στην διαγραφή της. Ο κώδικας που υλοποιεί την παραπάνω λειτουργικότητα παρατίθεται στη συνέχεια.

```
public void delCertainDialog(){

    dialogBuilder = new AlertDialog.Builder(getActivity());
    dialogBuilder.setView(confirmDropView);
    dialogBuilder.setCancelable(false);

    dialogBuilder.setPositiveButton(R.string.proceed, new
DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
```

```
            drop_password = confirm_pass.getText().toString();

            password_equality = myDBHandler.passwordChecking(drop_password);
            if (password_equality > 0) {
                deleteCertainConfDialog();
            } else {
                Toast.makeText(getActivity(), R.string.password_inequality,
Toast.LENGTH_SHORT)
                        .show();
            }
            confirm_pass.setText("");
            dialog.cancel();

            ((ViewGroup)
confirmDropView.getParent()).removeView(confirmDropView);
        }
    });
    dialogBuilder.setNegativeButton(R.string.cancel, new
DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
            confirm_pass.setText("");
            dialog.cancel();

            ((ViewGroup)
confirmDropView.getParent()).removeView(confirmDropView);
        }
    });

    AlertDialog alert = dialogBuilder.create();
    alert.show();
}

//Delete certain transaction-if exists- after confirmation.
public void deleteCertainConfDialog(){
    dialogBuilder = new AlertDialog.Builder(getActivity());
    dialogBuilder.setTitle(R.string.delete_transaction);
    dialogBuilder.setMessage(R.string.transaction_id);
    dialogBuilder.setCancelable(false);


    final EditText inputid = new EditText(getView().getContext());

    inputid.setInputType(InputType.TYPE_CLASS_PHONE);
    dialogBuilder.setView(inputid);


    dialogBuilder.setPositiveButton(R.string.ok , new
DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {

            /*
             * BUG FIXED when pressing OK without any values entered.
             */
            if (inputid.getText().toString().equals("")){
                Toast.makeText(getActivity(),R.string.fields_required,
Toast.LENGTH_SHORT)
                        .show();
```

```
            }
            else {
                id_transaction =
Integer.parseInt(inputid.getEditableText().toString());
            }


            if (myDBHandler.findTransaction(id_transaction) == 1) {
                transactionDialog(id_transaction);
            } else {
                Toast.makeText(getActivity(),R.string.transaction_error,
Toast.LENGTH_SHORT)
                        .show();
            }

        }
    });

    dialogBuilder.setNegativeButton(R.string.cancel, new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.cancel();
        }
    });

    AlertDialog alert = dialogBuilder.create();
    alert.show();
}


//Dialog window for transaction status display.
public void transactionDialog(final int id) {
    dialogBuilder = new AlertDialog.Builder(getActivity());
    dialogBuilder.setTitle(R.string.transaction_details);
    dialogBuilder.setMessage(R.string.warning_delete_transaction);
    dialogBuilder.setCancelable(false);


    String details;
    details = myDBHandler.transactionToString(id);

    final EditText transaction_details = new
EditText(getView().getContext());

    transaction_details.setText(details);
    transaction_details.setKeyListener(null);
    dialogBuilder.setView(transaction_details);

    dialogBuilder.setPositiveButton(R.string.delete, new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            myDBHandler.removeRow(id);
        }
    });
```

```
    dialogBuilder.setNegativeButton(R.string.cancel, new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.cancel();
        }

    });

    AlertDialog alert = dialogBuilder.create();
    alert.show();

}
```



Εικόνα 4.10: Εμφάνιση στοιχείων συναλλαγής προς διαγραφή

Από τον παραπάνω κώδικα υπάρχουν ορισμένα σημεία που χρίζουν προσοχής. Καταρχάς, αναφερόμαστε στο **bug** που αντιμετωπίστηκε και το οποίο συσχετιζότανε με την κενή εισαγωγή στοιχείων **(το bug αυτό έχει επιλυθεί σε όλα τα dialog boxes που χρησιμοποιούνται ως μέσα εισαγωγής δεδομένων και στα οποία έχουμε ήδη αναφερθεί).** Όταν ο χρήστης περνούσε το στάδιο της επιβεβαίωσης κωδικού με επιτυχία, και στο dialog box που ζητούσε το

αναγνωριστικό της συναλλαγής δεν εισήγαγε τίποτα αλλά πατούσε ΟΚ (με κενά δεδομένα) τότε η εφαρμογή οδηγούνταν σε crash. Για να επιλυθεί αυτό χρησιμοποιήθηκε ο παρακάτω έλεγχος:

```
if (inputid.getText().toString().equals("")){
    Toast.makeText(getActivity(),R.string.fields_required,
Toast.LENGTH_SHORT)
            .show();
}
```

Εφόσον τα δεδομένα που εισήχθησαν δεν ήταν κενά, με το πάτημα του Positive Button του dialog box, ΟΚ, ενεργοποιείται η κλήση της μεθόδου **findTransaction** της κλάσης **dbHandler**. Σκοπός της είναι να ελέγξει τον πίνακα των συναλλαγών της βάσης δεδομένων ώστε να διαπιστώσει εάν υπάρχει συναλλαγή με το συγκεκριμένο αναγνωριστικό. Αξίζει να σημειωθεί ότι το αναγνωριστικό της κάθε συναλλαγής είναι στην ουσία το **πρωτεύον κλειδί** που χρησιμοποιείται κατά την εγγραφή της καθεμίας στην βάση οπότε θα είναι μοναδικό. Ο κώδικας που υλοποιεί η παραπάνω μέθοδος είναι ο εξής:

```
public int findTransaction(int transid){
    int score;

    SQLiteDatabase db = getWritableDatabase();
    String query = "SELECT * FROM " + TABLE_TRANSACTIONS +
            " WHERE " + COLUMN_ID + "=\"" + transid + "\";";

    Cursor c = db.rawQuery(query,null);
    c.moveToFirst();
    score = c.getCount();


    c.close();
    db.close();

    return score;
}
```

Από τον κώδικα βλέπουμε ότι επιστρέφεται το πλήθος των entries (γραμμών) του πίνακα που ικανοποιούν το query-*το πλήθος θα είναι το πολύ ίσο με 1 όπως εξηγήσαμε λίγο παραπάνω*. Αφού, λοιπόν, προκύψει match έχουμε την κλήση της μεθόδου **transactionDialog** η οποία βρίσκεται στην κλάση **SettingsFragment** και είναι υπεύθυνη για την δημιουργία του δεύτερου dialog box το οποίο και κάνει χρήση της μεθόδου **transactionToString** της κλάσης **dbHandler** για να εμφανίσει τα στοιχεία της συναλλαγής που επιλέχθηκε προς διαγραφή όπως παρουσιάστηκε και στην Εικόνα [4.10].

Μόλις ο χρήστης επιλέξει το OK κουμπί, καλείται η μέθοδος **deleteRow** της κλάσης **dbHandler** και η επιλεγμένη γραμμή διαγράφεται. Ο κώδικας για την μέθοδο αυτή παρατίθεται στην συνέχεια.

```
public void removeRow(int transid){
    SQLiteDatabase db = getWritableDatabase();
    db.execSQL("DELETE FROM " + TABLE_TRANSACTIONS + " WHERE " +
            COLUMN_ID + "=\"" + transid + "\";");

    db.close();
}
```

Με την λειτουργία αυτή, κλείνει το κεφάλαιο το οποίο περιγράφει το μενού Settings και ανοίγει ένα νέο που αναφέρεται στο μενού Statistics.

Κεφάλαιο 5

5 STATISTICS FRAGMENT

Ένα επιπλέον μενού που συναντάει ο χρήστης ανοίγοντας το navigation drawer είναι αυτό των Statistics. Η κύρια λειτουργία που προσφέρεται είναι μία συνολική εικόνα (**Overview**) του αριθμού των συναλλαγών, του ποσού των εσόδων και εξόδων, καθώς και του αριθμού των συναλλαγών που αντιστοιχούν σε έξοδα και σε έσοδα. Τα συγκεντρωτικά αυτά αποτελέσματα είναι δυναμικά συνδεδεμένα με όλες τις υπόλοιπες λειτουργίες τις εφαρμογής και ανανεώνονται αυτόματα όταν προκύψει κάποια αλλαγή. Το χαρακτηριστικό που διαφοροποιεί το μενού αυτό, είναι –όπως θα δούμε και αναλυτικότερα στην συνέχεια- ο υπολογισμός των στατιστικών από την **αρχή του έτους μέχρι και την τρέχουσα ημερομηνία**.

Πιο συγκεκριμένα, ο χρήστης επιλέγοντας το παραπάνω μενού, ο λειτουργικός κώδικας του οποίου υλοποιείται στην κλάση **StatisticsFragment.java** και ο σχεδιαστικός στο **fragment_statistics.xml**, μπορεί να έχει μία πλήρη εικόνα όλων των ενεργειών που έχει πραγματοποιήσει μέχρι και την τρέχουσα ημερομηνία. Η αρχή του έτους ως εναρκτήρια ημερομηνία τέθηκε με το σκεπτικό ότι είναι κοινή για κάθε χρήστη άσχετα αν κάποιος(που είναι και εξαιρετικά πιθανό) ξεκινήσει να αλληλεπιδρά με την εφαρμογή κάποια στιγμή αργότερα. Μία διαφορετική υλοποίηση θα μπορούσε να είναι η ημερομηνία που ο χρήστης ξεκινά να χρησιμοποιεί την εφαρμογή για πρώτη φορά. Αυτό, όμως, μπορούσε να δημιουργήσει προβλήματα ειδικά μετά από μια **Εκκαθάριση Δεδομένων** της εφαρμογής (Clear Data). Όπως θα φανεί και στον κώδικα που παρατίθεται στην συνέχεια, συγκεκριμένα μέτρα έχουν ληφθεί για την ομαλή μετάβαση από το ένα έτος στο επόμενο.

```
public void showStatistics(){
    Calendar c1 = Calendar.getInstance();
    Calendar c2 = Calendar.getInstance();

    //Setting the START date of CURRENT year.
    year = c1.get(Calendar.YEAR);
    c1.set(Calendar.YEAR,year);
    c1.set(Calendar.MONTH,1);
    c1.set(Calendar.DAY_OF_YEAR,1);

    SimpleDateFormat df = new SimpleDateFormat("yyyy/MM/dd");

    String date1 = df.format(c1.getTime());
    String date2 = df.format(c2.getTime());

    //Checking if there are any transactions up to NOW.
    existence = myDBHandler.yearlyTransactions(date1,date2);


    //If there are transactions in the database they are loaded.
```

```
    if (existence > 0){

Toast.makeText(getActivity(),R.string.transactions_found_success,Toast.LENGTH
_SHORT)
                .show();
        entry = myDBHandler.showTransactionsYearly(date1,date2);
        load_area.setText(entry);
    }
    else{

Toast.makeText(getActivity(),R.string.transactions_not_exist,Toast.LENGTH_SHO
RT)
                .show();
    }

    //This button simply clears the screen from the statistics.
    clear_button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            load_area.setText("");
        }
    });
}
```

Για να είμαστε σε θέση να ανανεώνουμε αυτόματα το έτος χρησιμοποιούμε τον εξής κώδικα:

```
Calendar c1 = Calendar.getInstance();
Calendar c2 = Calendar.getInstance();

//Setting the START date of CURRENT year.
year = c1.get(Calendar.YEAR);
c1.set(Calendar.YEAR,year);
c1.set(Calendar.MONTH,1);
c1.set(Calendar.DAY_OF_YEAR,1);
```

Καταρχάς, με την χρήση της μεθόδου **getInstance** της abstract κλάσης **Calendar** είμαστε σε θέση να δημιουργήσουμε αντικείμενα της κλάσης αυτής τα οποία έχουν αρχικοποιηθεί στην τρέχουσα ώρα και ημερομηνία. Επομένως, τα αντικείμενα αυτά μπορούν τώρα να αξιοποιήσουν τις μεθόδους που προσφέρει η κλάση **Calendar**. Η γενική ιδέα είναι να δημιουργήσουμε μια μεταβλητή η οποία θα είναι της μορφής 20XX/01/01, όπου XX θα είναι το τρέχον έτος και θα είναι το μόνο που μεταβάλλεται. Εφόσον ο μήνας και η ημέρα θα έχουν σταθερές τιμές, αρκεί να θέσουμε (**set**) την τιμή της μεταβλητής για τον MONTH και για το DAY_OF_YEAR (**οι οποίες είναι public static final μεταβλητές της κλάσης Calendar**) ίση με 1. Για το έτος, όμως, χρησιμοποιείται η μεταβλητή *year* η οποία παίρνει την τιμή του αυτόματα:

```
year = c1.get(Calendar.YEAR);
```

Αφού επιλύσαμε το παραπάνω, θα πρέπει τώρα να διαπιστώσουμε αν όντως ο χρήστης έχει πραγματοποιήσει κάποιες συναλλαγές από την αρχή του έτους. Για να γίνει αυτό χρησιμοποιείται η μέθοδος **yearlyTransactions** της κλάσης **dbHandler** ο κώδικας της οποίας παρουσιάζεται στην συνέχεια:

```
public int yearlyTransactions(String date1, String date2){
    int existence;

    SQLiteDatabase db = getWritableDatabase();
    String query = "SELECT * FROM " + TABLE_TRANSACTIONS +
            " WHERE " + COLUMN_DATE + " BETWEEN '"+date1+"' AND '"+date2+"'";

    Cursor c = db.rawQuery(query,null);
    c.moveToFirst();

    existence = c.getCount();

    c.close();
    db.close();

    return existence;
}
```

Παρατηρώντας τον παραπάνω κώδικα, διαπιστώνουμε ότι η μέθοδος αυτή δέχεται δύο παραμέτρους, την αρχική ημερομηνία (που είναι και η πρώτη μέρα του τρέχοντος έτους) και την καταληκτική ημερομηνία που είναι η τρέχουσα ημερομηνία. Χρησιμοποιώντας στον πίνακα των συναλλαγών ένα query(ερώτημα προς τον πίνακα της βάσης) το οποίο κάνει χρήση των εντολών **"BETWEEN AND"** της SQLite, είμαστε σε θέση –εφαρμόζοντας και την μέθοδο **getCount** της κλάσης **Cursor**- να υπολογίσουμε το **πλήθος** των συναλλαγών ανάμεσα στις δύο ημερομηνίες *date1* και *date2* και να το επιστρέψουμε.

Από το αποτέλεσμα της κλήσης της παραπάνω μεθόδου κρίνεται αυτό που θα εμφανιστεί στην οθόνη του χρήστη. Αν δεν υπάρχουν συναλλαγές απλά εμφανίζεται ένα μήνυμα μη εύρεσης συναλλαγών. Αν, από την άλλη υπάρχουν, καλείται η μέθοδος **showTransactionsYearly** της κλάσης **dbHandler** ο κώδικας της οποίας φαίνεται στην συνέχεια.

```
public String showTransactionsYearly(String date1,String date2){

    String dbString = "";

    String start_date = "From: ";
    String end_date = " To: ";

    String stats = " the statistics are: ";
    String total_income = "Total Income: ";
    String total_expenses = "Total Expenses: ";
```

```
    String total_transactions = "Total number of transactions is: ";
    String income_trans = "Total number of income transactions: ";
    String expenses_trans = "Total number of expenses transactions: ";

    double total_inc,total_exp;
    int transactions_number,inc_transactions,exp_transactions;



    SQLiteDatabase db = getWritableDatabase();



    String query = "SELECT * FROM " + TABLE_TRANSACTIONS +
            " WHERE " + COLUMN_DATE + " BETWEEN '"+date1+"' AND '"+date2+"'";


    String query2 = "SELECT SUM(" + COLUMN_AMOUNT + " ) " + " AS IncomeTotal
" +
            " FROM " + TABLE_TRANSACTIONS +
            " WHERE " + COLUMN_TRANSACTION + "=\"" + 0 + "\";";


    String query3 = "SELECT SUM(" + COLUMN_AMOUNT + " ) " + " AS
ExpensesTotal " +
            " FROM " + TABLE_TRANSACTIONS +
            " WHERE " + COLUMN_TRANSACTION + "=\"" + 1 + "\";";


    String query4 = "SELECT * FROM " + TABLE_TRANSACTIONS +
            " WHERE " + COLUMN_DATE + " BETWEEN '"+date1+"' AND '"+date2+"'";


    String query5 = "SELECT * FROM " + TABLE_TRANSACTIONS +
            " WHERE " + COLUMN_TRANSACTION + "=\"" + 0 + "\";";


    String query6 = "SELECT * FROM " + TABLE_TRANSACTIONS +
            " WHERE " + COLUMN_TRANSACTION + "=\"" + 1 + "\";";



    //Cursor points to a location in your results
    Cursor c  = db.rawQuery(query, null);
    Cursor c1 = db.rawQuery(query2, null);
    Cursor c2 = db.rawQuery(query3, null);
    Cursor c3 = db.rawQuery(query4, null);
    Cursor c4 = db.rawQuery(query5, null);
    Cursor c5 = db.rawQuery(query6, null);


    //Move to the first row in your results
    c.moveToFirst();
    c1.moveToFirst();
```

```
c2.moveToFirst();
c3.moveToFirst();
c4.moveToFirst();
c5.moveToFirst();


total_inc = c1.getDouble(c1.getColumnIndex("IncomeTotal"));
total_exp = c2.getDouble(c2.getColumnIndex("ExpensesTotal"));
transactions_number = c3.getCount();
inc_transactions = c4.getCount();
exp_transactions = c5.getCount();

start_date += date1;
start_date += end_date;
start_date += date2;
start_date += stats;
start_date += "\n";
start_date += "############################";
start_date += "\n";
start_date += total_income;
start_date += total_inc;
start_date += "\n";
start_date += "-------------------------------------------------";
start_date += "\n";
start_date += total_expenses;
start_date += total_exp;
start_date += "\n";
start_date += "-------------------------------------------------";
start_date += "\n";
start_date += total_transactions;
start_date += transactions_number;
start_date += "\n";
start_date += "-------------------------------------------------";
start_date += "\n";
start_date += income_trans;
start_date += inc_transactions;
start_date += "\n";
start_date += "-------------------------------------------------";
start_date += "\n";
start_date += expenses_trans;
start_date += exp_transactions;
start_date += "\n";
start_date += "-------------------------------------------------";
start_date += "\n";




c.close();
c1.close();
c2.close();
c3.close();
c4.close();
c5.close();

db.close();
```

```
    return start_date;
}
```

Μελετώντας τον κώδικα, μπορεί κάποιος να παρατηρήσει τα queries που έχουν δημιουργηθεί και που το καθένα αντιστοιχεί σε κάθε ένα από τα εμφανιζόμενα στατιστικά που προαναφέραμε στην αρχή του κεφαλαίου.

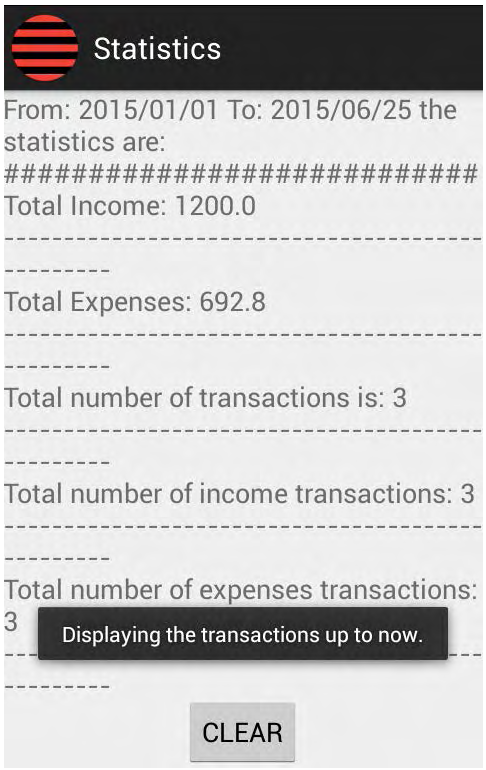
Με την κλήση της μεθόδου αυτής, λοιπόν, και δεδομένου ότι υπάρχουν κάποιες πραγματοποιημένες συναλλαγές, το TextView widget που υποστηρίζει το view του **StatisticsFragment** εμφανίζει τα επιθυμητά αποτελέσματα. Προγραμματιστικά αυτό φαίνεται στον παρακάτω κώδικα:

```
if (existence > 0){

Toast.makeText(getActivity(),R.string.transactions_found_success,Toast.LENGTH
_SHORT)
            .show();
    entry = myDBHandler.showTransactionsYearly(date1,date2);
    load_area.setText(entry);
}
```

Εφόσον ολοκληρώθηκε η περιγραφή της λειτουργίας που επιτελεί το εν λόγω μενού, παραθέτουμε τον σχεδιαστικό κώδικα του statistics_fragment.xml καθώς και την βασική απεικόνιση των αποτελεσμάτων όπως παρουσιάζονται στον χρήστη, στην Εικόνα [4.11].

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.dimitris.moneyger.StatisticsFragment">



    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:maxLines="20"
        android:scrollbars="vertical"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:id="@+id/loadTextStatistics"
        android:layout_gravity="center_horizontal|top" />
```

```
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/clearButton"
        android:id="@+id/buttonMonthOk"
        android:layout_gravity="center_horizontal|bottom" />


</FrameLayout>
```



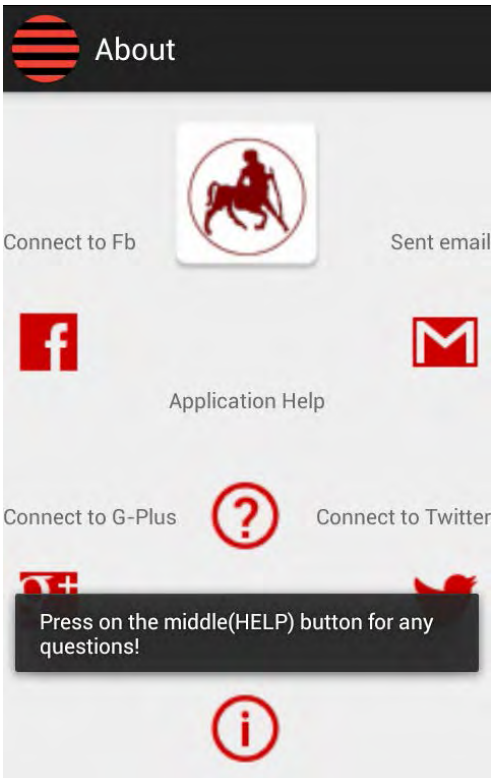
Εικόνα 4.11: Συνολική εικόνα ενεργειών χρήστη

Κεφάλαιο 6

6 ABOUT FRAGMENT

Το About Fragment είναι το τελευταίο μενού το οποίο ο χρήστης συναντά αν ανοίξει το navigation drawer. Πρόκειται, ίσως, για το πιο απλό μενού της εφαρμογής που έχει ως στόχο, πρώτον, την αλληλεπίδραση του χρήστη με τον προγραμματιστή και δεύτερον την παροχή πληροφοριών σχετικά με τον τρόπο λειτουργίας της εφαρμογής.

Χρησιμοποιούνται τέσσερα βασικά εικονίδια τα οποία είναι όλα pressable, τα τρία εκ των οποίων λειτουργούν ως Links σε προσωπικές ιστοσελίδες του προγραμματιστή ενώ το τέταρτο χρησιμοποιείται για αποστολή mail με τυχόν ερωτήσεις. Υπάρχουν, επίσης, δύο κεντρικά εικονίδια τα οποία παρέχουν, το πρώτο, βοήθεια σχετικά με τον τρόπο λειτουργίας της εφαρμογής και το δεύτερο πληροφορίες για τον δημιουργό της. Τέλος, στο άνω μέρος του view υπάρχει το **Logo** του **Πανεπιστημίου Θεσσαλίας** μέσω του οποίο ο χρήστης μπορεί να μεταβεί στην ιστοσελίδα της σχολής www.inf.uth.gr. Όλα όσα περιγράφτηκαν παρουσιάζονται και γραφικά στην Εικόνα [6.1].



Εικόνα 6.1: Γενική απεικόνιση του About μενού

Θα αναλυθεί στην συνέχεια ο τρόπος λειτουργίας των παραπάνω εικονιδίων τόσο με την χρήση του κώδικα στο αρχείο **AboutFragment.java** όσο και στο **fragment_about.xml**. Ξεκινώντας, παρατίθεται ο κώδικας που αντιστοιχεί στο σχεδιαστικό κομμάτι του παρόντος μενού και ο οποίος υλοποιείται στο xml αρχείο.

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.dimitris.moneyger.AboutFragment">

    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        />


    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="center">

        <FrameLayout
            android:layout_width="match_parent"
            android:layout_height="125dp"
            android:layout_gravity="left|top">

            <ImageView
                android:layout_width="117dp"
                android:layout_height="117dp"
                android:id="@+id/imageViewUth"
                android:layout_gravity="center_horizontal|bottom"
                android:background="@drawable/ic_about_me" />

        </FrameLayout>

        <FrameLayout
            android:layout_width="match_parent"
            android:layout_height="275dp"
            android:layout_gravity="center">

            <FrameLayout
                android:layout_width="match_parent"
                android:layout_height="106dp"
                android:layout_gravity="left|top">

                <ImageButton
                    android:layout_width="60dp"
                    android:layout_height="60dp"
                    android:id="@+id/imageButtonFacebook"
                    android:layout_gravity="left|bottom"
                    android:background="@drawable/ic_action_facebook" />
```

```
                <ImageButton
                    android:layout_width="60dp"
                    android:layout_height="60dp"
                    android:id="@+id/imageButtonGMail"
                    android:layout_gravity="right|bottom"
                    android:background="@drawable/ic_action_gmail" />

                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"

android:textAppearance="?android:attr/textAppearanceSmall"
                    android:text="@string/facebook"
                    android:id="@+id/textView12"
                    android:layout_gravity="left|top" />

                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"

android:textAppearance="?android:attr/textAppearanceSmall"
                    android:text="@string/gmail"
                    android:id="@+id/textView13"
                    android:layout_gravity="right|top" />

            </FrameLayout>

            <FrameLayout
                android:layout_width="match_parent"
                android:layout_height="171dp"
                android:layout_gravity="left|bottom">

                <ImageButton
                    android:layout_width="60dp"
                    android:layout_height="60dp"
                    android:id="@+id/imageButtonGPlus"
                    android:layout_gravity="left|bottom"
                    android:background="@drawable/ic_action_gplus" />

                <ImageButton
                    android:layout_width="60dp"
                    android:layout_height="60dp"
                    android:id="@+id/imageButtonTwitter"
                    android:layout_gravity="right|bottom"
                    android:background="@drawable/ic_action_twitter" />

                <ImageButton
                    android:layout_width="70dp"
                    android:layout_height="70dp"
                    android:id="@+id/imageButtonHelp"
                    android:layout_gravity="center"
                    android:background="@drawable/ic_action_help" />

                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
```

```
android:textAppearance="?android:attr/textAppearanceSmall"
                    android:text="@string/help"
                    android:id="@+id/textView14"
                    android:layout_gravity="center_horizontal|top" />

                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"

android:textAppearance="?android:attr/textAppearanceSmall"
                    android:text="@string/gplus"
                    android:id="@+id/textView15"
                    android:layout_gravity="left|center_vertical" />

                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"

android:textAppearance="?android:attr/textAppearanceSmall"
                    android:text="@string/twitter"
                    android:id="@+id/textView16"
                    android:layout_gravity="right|center_vertical" />
            </FrameLayout>
        </FrameLayout>

        <FrameLayout
            android:layout_width="match_parent"
            android:layout_height="111dp"
            android:layout_gravity="left|bottom">


            <ImageButton
                android:layout_width="70dp"
                android:layout_height="70dp"
                android:id="@+id/imageButtonMe"
                android:layout_gravity="center_horizontal|bottom"
                android:background="@drawable/ic_action_info" />

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textAppearance="?android:attr/textAppearanceSmall"
                android:text="@string/info"
                android:id="@+id/textView"
                android:layout_gravity="center_horizontal|top" />

        </FrameLayout>

    </FrameLayout>


</FrameLayout>
```

Ο κώδικας ο οποίος προσδίδει λειτουργικότητα στα αντικείμενα που σχεδιάζονται παραπάνω βρίσκεται στο java αρχείο. Η κύρια μέθοδος, όπως θα φανεί και στον κώδικα στην συνέχεια, που αρχικοποιεί όλες τις μεταβλητές είναι η **setupFields** η οποία και καλείται αφού έχει δημιουργηθεί το view μέσα από την μέθοδο **onViewCreated**.

```
public void setupFields(){
    imgUth = (ImageView) getView().findViewById(R.id.imageViewUth);

    facebook    =   (ImageButton)
getView().findViewById(R.id.imageButtonFacebook);
    twitter     =   (ImageButton)
getView().findViewById(R.id.imageButtonTwitter);
    gplus       =   (ImageButton)
getView().findViewById(R.id.imageButtonGPlus);
    gmail       =   (ImageButton)
getView().findViewById(R.id.imageButtonGMail);
    help        =   (ImageButton)
getView().findViewById(R.id.imageButtonHelp);
    author      =   (ImageButton)
getView().findViewById(R.id.imageButtonMe);



    //Adding animations.

    facebook.startAnimation(AnimationUtils.loadAnimation(getActivity(),
            android.R.anim.slide_in_left));
    twitter.startAnimation(AnimationUtils.loadAnimation(getActivity(),
            android.R.anim.slide_in_left));
    gplus.startAnimation(AnimationUtils.loadAnimation(getActivity(),
            android.R.anim.slide_in_left));
    gmail.startAnimation(AnimationUtils.loadAnimation(getActivity(),
            android.R.anim.slide_in_left));
    author.startAnimation(AnimationUtils.loadAnimation(getActivity(),
            android.R.anim.slide_in_left));
    help.startAnimation(AnimationUtils.loadAnimation(getActivity(),
            android.R.anim.slide_in_left));


    //Setting the actions performed by clicking the buttons and logo.

    imgUth.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            uthPage();
        }
    });


    facebook.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            facebookPage();
        }
```

```
    });


    twitter.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            twitterPage();
        }
    });


    gmail.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            gmailPage();
        }
    });


    gplus.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            gplusPage();
        }
    });



    help.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            helpPage();
        }
    });


    author.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            auhtorPage();
        }
    });

}
```

Αν παρατηρήσουμε πιο προσεκτικά τον παραπάνω κώδικα θα διαπιστώσουμε ότι υπάρχουν δύο σημεία τα οποία χρίζουν προσοχής. Το πρώτο είναι η χρήση **animations** που χρησιμοποιήθηκαν και στο **SettingsFragment** και το δεύτερο η λειτουργία των κουμπιών. Για την υποστήριξη του Animation χρησιμοποιείται ο εξής κώδικας ο οποίος και επεξηγείται στην συνέχεια:

```
facebook.startAnimation(AnimationUtils.loadAnimation(getActivity(),
        android.R.anim.slide_in_left));
twitter.startAnimation(AnimationUtils.loadAnimation(getActivity(),
```

```
        android.R.anim.slide_in_left));
gplus.startAnimation(AnimationUtils.loadAnimation(getActivity(),
        android.R.anim.slide_in_left));
gmail.startAnimation(AnimationUtils.loadAnimation(getActivity(),
        android.R.anim.slide_in_left));
author.startAnimation(AnimationUtils.loadAnimation(getActivity(),
        android.R.anim.slide_in_left));
help.startAnimation(AnimationUtils.loadAnimation(getActivity(),
        android.R.anim.slide_in_left));
```

Όπως φαίνεται χρησιμοποιούνται δύο, κυρίως, μέθοδοι η **startAnimation** και η **loadAnimation** των κλάσεων **View** και **AnimationUtils** αντίστοιχα. Αναλύοντας την δεύτερη μέθοδο, χρησιμοποιείται για να «φορτώσει» το animation που ο προγραμματιστής επιθυμεί (στην προκειμένη περίπτωση έχει επιλεγεί μία κύλιση από τα αριστερά προς τα δεξιά) να ενσωματώσει σε κάθε ένα από τα διαθέσιμα **ImageButtons**. Η πρώτη μέθοδος χρησιμοποιείται, εν συνεχεία, για να θέσει σε λειτουργία το ενσωματωμένο animation. Στο σημείο αυτό θα κάνουμε μια σύντομη παρένθεση ώστε να εξηγήσουμε τι ακριβώς είναι το ImageButton.

Όπως φανερώνει και το όνομά του καθώς και ο κώδικας xml που παρουσιάστηκε προηγουμένως, το **ImageButton**, είναι ακόμα ένα ***widget*** που υποστηρίζει η γλώσσα. Πιο συγκεκριμένα είναι *ακριβώς ίδιο με ένα απλό Button widget*, τόσο ως προς την λειτουργία όσο και ως προς την χρήση του, με την διαφορά ότι έχει την δυνατότητα να απεικονίζει αντί για κείμενο εικόνα κάνοντάς το καταλληλότερο από ένα απλό Button για τις ανάγκες της παρούσας εφαρμογής.

Κλείνοντας αυτή την μικρή παρένθεση, και παίρνοντας ως παράδειγμα το κουμπί που μας συνδέει με την προσωπική σελίδα του Google+ του προγραμματιστή (όλα τα κουμπιά-links έχουν **ακριβώς** τον ίδιο κώδικα υλοποιημένο) θα αναλύσουμε πως επιτυγχάνεται η σύνδεση αυτή. Εφόσον το πάτημα του κουμπιού θα οδηγήσει στο άνοιγμα μιας νέας εφαρμογής (μπορεί να ανοίξει ο browser ή και η ίδια η εφαρμογή για το Google+ αν βρίσκεται προεγκατεστημένη στην συσκευή του χρήστη) χρειαζόμαστε ένα αντικείμενο τύπου κλάσης **Intent**, η οποία χρησιμοποιείται κυρίως για την έναρξη κάποιου νέου **activity**.

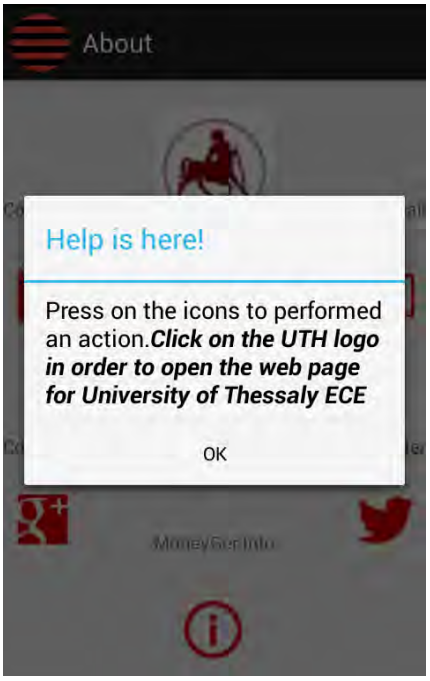
Αφού δημιουργηθεί το εν λόγω αντικείμενο, χρησιμοποιούνται τρεις βασικές μέθοδοι της κλάσης **Intend**, η **setAction**, η **addCategory** και η **setData** οι οποίες και θα περιγράφουνε αμέσως.

- Η πρώτη μέθοδος χρησιμοποιείται με την παράμετρο *ACTION_VIEW* η οποία της επιτρέπει να φέρει στο προσκήνιο την λίστα με τις διαθέσιμες εφαρμογές στην συσκευή του χρήστη που μπορούν να εκτελέσουν την μετάβαση στην ιστοσελίδα.
- Η δεύτερη μέθοδος είναι απαραίτητη όταν ο χρήστης επιλέξει μετάβαση στην ιστοσελίδα μέσω του browser. Αν δεν υπήρχε η παράμετρος *CATEGORY_BROWSABLE* ο χρήστης θα μετέβαινε στην σελίδα επιτυχώς αλλά δεν θα μπορούσε να εκτελέσει καμία περεταίρω ενέργεια-όπως για παράδειγμα σύνδεση στον λογαριασμό του.
- Η τρίτη μέθοδος, τέλος, είναι αυτή που παρέχει το Link για την πραγματοποίηση της σύνδεσης.

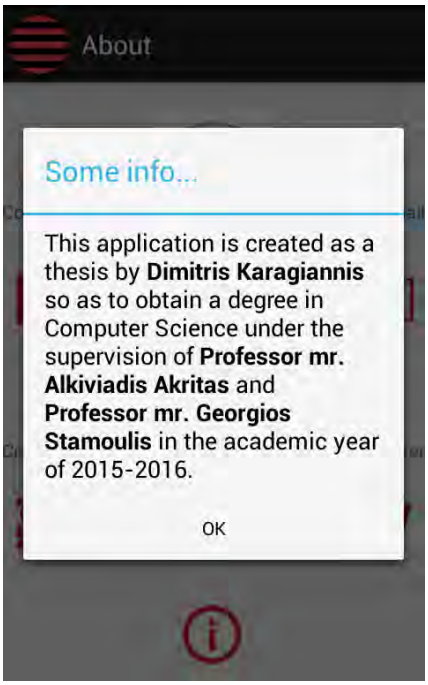
Εφόσον κάνουμε τα παραπάνω, απομένει να ενεργοποιήσουμε την επιλεγμένη εφαρμογή με χρήση της μεθόδου **startActivity** και ως παράμετρο το αντικείμενο τύπου **Intent** που περιέχει όλες τις παραπάνω «παραμέτρους». Με την ίδια λογική γίνεται και η αποστολή email προς τον προγραμματιστή του οποίου η διεύθυνση έχει περάσει ως fixed παράμετρος μέσω της μεθόδου **putExtra**(δέχεται δύο παραμέτρους, την *EMAIL_EXTRA* που μας προετοιμάζει ότι περιμένει το String το οποίο περιέχει την διεύθυνση προορισμού του μηνύματος, και το String αυτό).

Υπάρχουν, τέλος, και τα δύο κεντρικά κουμπιά των οποίων οι λειτουργίες παρουσιάζονται στην Εικόνα [6.2] και στην Εικόνα [6.3] αντίστοιχα και περιλαμβάνουν την παροχή πληροφοριών σχετικά με τον δημιουργό της εφαρμογής και τον τρόπο λειτουργίας της αντίστοιχα.

Εικόνα 6.2: Πατώντας το κουμπί της Βοήθειας Πληροφοριών



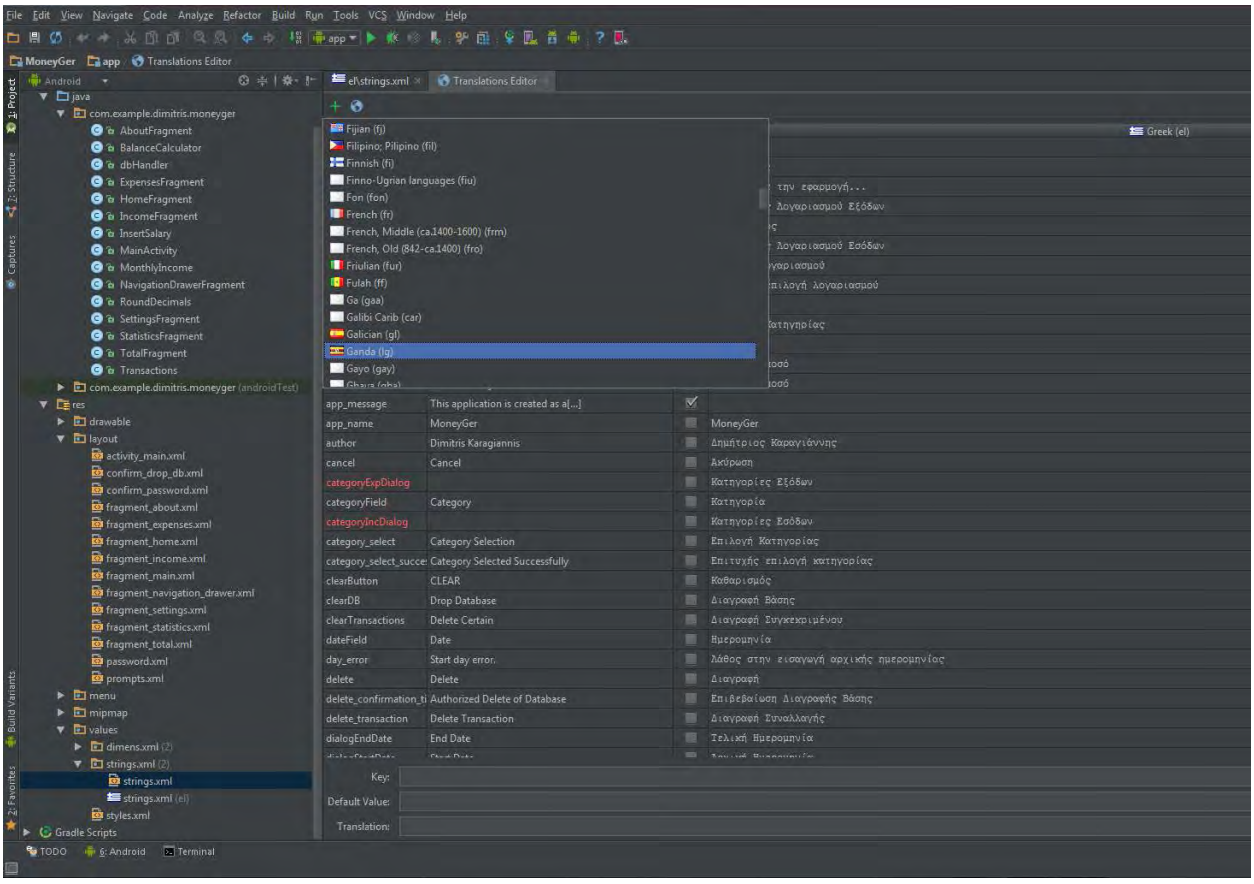
Εικόνα 6.3: Πατώντας το κουμπί των



Κεφάλαιο 7

7 ΥΠΟΣΤΗΡΙΞΗ ΕΛΛΗΝΙΚΗΣ ΓΛΩΣΣΑΣ

Το παρόν κεφάλαιο έχει ως στόχο την επίδειξη της, μεταφρασμένης στα ελληνικά, εφαρμογής μέσω της παράθεσης του xml κώδικα που το επιτυγχάνει. Καταρχάς θα πρέπει να αναφέρουμε ότι η αλλαγή της γλώσσας επιτυγχάνεται μόνο με την αλλαγή της προεπιλεγμένης γλώσσας της συσκευής του χρήστη στα ελληνικά. Η ίδια η μετάφραση πραγματοποιείται χρησιμοποιώντας τον Translations Editor που παρέχεται από το Android Studio και που παρουσιάζεται στην Εικόνα [7.1].



Στην συνέχεια παρατίθεται και το μεταφρασμένο αρχείο strings.xml που περιέχει τις μεταφράσεις για όλα τα strings που χρησιμοποιούνται στην εφαρμογή. Αξίζει να σημειωθεί, ότι η μετάφραση σε οποιαδήποτε γλώσσα δεν επηρεάζει σε καμία περίπτωση το αρχικό strings.xml αρχείο αλλά δημιουργεί αντίγραφά του ένα για κάθε γλώσσα.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">MoneyGer</string>
    <string name="abort">Ακύρωση</string>
    <string name="about">Σχετικά με</string>
    <string name="about_app">Σχετικά με την εφαρμογή...</string>
    <string name="accountField">Λογαριασμός</string>
    <string name="action_settings">Ρυθμίσεις</string>
    <string name="addCategory">Προσθήκη Κατηγηρίας</string>
    <string name="amountField">Ποσό</string>
    <string name="amountPrompt">Εισάγετε ποσό</string>
    <string name="amountTitle">Εισαχθέν ποσό</string>
    <string name="author">Δημήτριος Καραγιάννης</string>
    <string name="cancel">Ακύρωση</string>
    <string name="categoryField">Κατηγορία</string>
    <string name="clearButton">Καθαρισμός</string>
    <string name="clearDB">Διαγραφή Βάσης</string>
    <string name="clearTransactions">Διαγραφή Συγκεκριμένου</string>
    <string name="dateField">Ημερομηνία</string>
    <string name="day_error">Λάθος στην εισαγωγή αρχικής ημερομηνίας</string>
    <string name="delete">Διαγραφή</string>
    <string name="delete_confirmation_title">Επιβεβαίωση Διαγραφής
Βάσης</string>
    <string name="delete_transaction">Διαγραφή Συναλλαγής</string>
    <string name="dialogEndDate">Τελική Ημερομηνία</string>
    <string name="dialogStartDate">Αρχική Ημερομηνία</string>
    <string name="dialogTitle">Εισάγετε αρχική και τελική ημερομηνία για
έλεγχο:</string>
    <string name="discardButton">Ακύρωση</string>
    <string name="drawer_elements">Στοιχεία Συρταριού</string>
    <string name="host_elements">Στοιχεία Μενού</string>
    <string name="sections_icons">Εικονίδια</string>
    <string name="accountIncDialog">Κατηγορίες Λογαριασμού Εσόδων</string>
    <string name="accountExpDialog">Κατηγορίες Λογαριασμού Εξόδων</string>
    <string name="entrance_authorization">Επιβεβαιώστε την είσοδό
σας</string>
    <string name="categoryIncDialog">Κατηγορίες Εσόδων</string>
    <string name="categoryExpDialog">Κατηγορίες Εξόδων</string>
    <string name="eraseConfirm">Είστε βέβαιοι πως θέλετε να διαγράψετε όλες
τις ρυθμίσεις?</string>
    <string name="eraseTitle">Διαγραφή Βάσης Δεδομένων</string>
    <string name="facebook">Σύνδεση στο Fb</string>
    <string name="fields_required">Όλα τα πεδία είναι υποχρεωτικά.</string>
    <string name="findTransaction">Εύρεση</string>
    <string name="gmail">Αποστολή mail</string>
    <string name="gplus">Σύνδεση στο Google+</string>
    <string name="help">Βοήθεια εφαρμογής</string>
    <string name="help_purpose">Πατήστε στα εικονίδια για κάποια από τις
εγγεγραμμένες ενέργειες. Πατήστε στο logo του πανεπιστημίου για μετάβαση στην
σελίδα της σχολής.</string>
    <string name="help_title">Βοήθεια!</string>
    <string name="home">Αρχική</string>
    <string name="loadButton">Φόρτωση</string>
    <string name="new_pass">Νέος κωδικός</string>
```

```
    <string name="new_pass_msg">Νέος κωδικός</string>
    <string name="no">Όχι</string>
    <string name="old_pass_msq">Παλιός Κωδικός</string>
    <string name="old_title">Ανανέωση Κωδικού</string>
    <string name="password">Κωδικός</string>
    <string name="password_add">Ο κωδικός καταχωρήθηκε</string>
    <string name="password_exists">Ο κωδικός έχει ενεργοποιηθεί ήδη</string>
    <string name="password_inequality">Οι κωδικοί δεν ταιριάζουν</string>
    <string name="password_prompt">Εισάγετε κωδικό</string>
    <string name="press_on_help">Πατήστε στο μεσσαίο εικονίδιο για
βοήθεια.</string>
    <string name="press_on_text">Πατήστε στο κείμενο για εισαγωγή
δεδομένων</string>
    <string name="proceed">Προχώρα</string>
    <string name="salaryEntryTitle">Εισαγωγή Μισθού</string>
    <string name="salaryPrompt">Εισάγετε μισθό</string>
    <string name="salary_enter">Εισάγετε μισθό:</string>
    <string name="saveButton">Αποθήκευση</string>
    <string name="search">Εύρεση</string>
    <string name="settings">Ρυθμίσεις</string>
    <string name="show_transactions">Εμφάνιση Συναλλαγών</string>
    <string name="statistics">Στατιστικά</string>
    <string name="toggleButton">Αλλαγή εμφάνισης</string>
    <string name="totalBalance">Υπόλοιπο:</string>
    <string name="totalExpenses">Έξοδα:</string>
    <string name="totalIncome">Έσοδα:</string>
    <string name="trans_between_datesMSG">Βάσει των ημερομηνιών οι συναλλαγές
είναι:</string>
    <string name="trans_between_datesTITLE">Αποτελέσματα συναλλαγών</string>
    <string name="trans_existence">Δεν υπάρχουν συναλλαγές ανάμεσα στις
ημερομηνίες.</string>
    <string name="transaction_details">Πληροφορίες συναλλαγών</string>
    <string name="transaction_error">Η συναλλαγή δεν βρέθηκε</string>
    <string name="transaction_id">Εισάγετε κωδικό συναλλαγής:</string>
    <string name="transactions_found_success">Εμφάνιση συναλλαγών από αρχή
του έτους:</string>
    <string name="transactions_not_exist">Καμία συναλλαγή έως τώρα</string>
    <string name="twitter">Σύνδεση στο Twitter</string>
    <string name="update">Ανανέωση</string>
    <string name="update_success">Ανανέωση κωδικού επιτυχής</string>
    <string name="warning_delete_transaction">Πρόκειται να διαγράψετε την
συναλλαγή:</string>
    <string name="yes">Ναι</string>
    <string name="saved_all">Όλα αποθηκεύτηκαν</string>
    <string name="account_select">Επιλογή Λογαριασμού</string>
    <string name="account_select_success">Επιτυχής επιλογή
λογαριασμού</string>
    <string name="category_select">Επιλογή Κατηγορίας</string>
    <string name="category_select_success">Επιτυχής επιλογή
κατηγορίας</string>
</resources>
```

Κεφάλαιο 8

8 ΠΑΡΟΥΣΙΑΣΗ ΚΩΔΙΚΑ ΕΦΑΡΜΟΓΗΣ

Στο κεφάλαιο αυτό θα παρουσιαστεί ο κώδικας που υλοποιεί την εφαρμογή- μέρη του οποίου χρησιμοποιούνται και σε προηγούμενα κεφάλαια ώστε να αναλυθούν συγκεκριμένες λειτουργίες και να επεξηγηθεί ο τρόπος σκέψης πίσω από αυτές.

8.1 ΚΩΔΙΚΑΣ ΓΙΑ NAVIGATIONDRAWERFRAGMENT ΚΛΑΣΗ

8.1.1 Κώδικας Java

```
package com.example.dimitris.moneyger;

import android.annotation.TargetApi;
import android.content.res.TypedArray;
import android.graphics.drawable.Drawable;
import android.os.Build;
import android.support.v7.app.ActionBarActivity;
import android.app.Activity;
import android.support.v7.app.ActionBar;
import android.support.v4.app.Fragment;
import android.support.v4.app.ActionBarDrawerToggle;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.content.SharedPreferences;
import android.content.res.Configuration;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

/**
 * Fragment used for managing interactions for and presentation of a navigation
drawer.
 * See the <a href="https://developer.android.com/design/patterns/navigation-
drawer.html#Interaction">
 * design guidelines</a> for a complete explanation of the behaviors implemented here.
 */
public class NavigationDrawerFragment extends Fragment {
```

```
    /*
     * Using for string.xml reference.
     */
    private String[] arrayTemp;

    /**
     * Remember the position of the selected item.
     */
    private static final String STATE_SELECTED_POSITION =
"selected_navigation_drawer_position";

    /**
     * Per the design guidelines, you should show the drawer on launch until the user
manually
     * expands it. This shared preference tracks this.
     */
    private static final String PREF_USER_LEARNED_DRAWER =
"navigation_drawer_learned";

    /**
     * A pointer to the current callbacks instance (the Activity).
     */
    private NavigationDrawerCallbacks mCallbacks;

    /**
     * Helper component that ties the action bar to the navigation drawer.
     */
    private ActionBarDrawerToggle mDrawerToggle;

    private DrawerLayout mDrawerLayout;
    private ListView mDrawerListView;
    private View mFragmentContainerView;

    private int mCurrentSelectedPosition = 0;
    private boolean mFromSavedInstanceState;
    private boolean mUserLearnedDrawer;


    public NavigationDrawerFragment() {
    }

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

        // Read in the flag indicating whether or not the user has demonstrated
awareness of the
        // drawer. See PREF_USER_LEARNED_DRAWER for details.
        SharedPreferences sp =
PreferenceManager.getDefaultSharedPreferences(getActivity());
        mUserLearnedDrawer = sp.getBoolean(PREF_USER_LEARNED_DRAWER, false);

        if (savedInstanceState != null) {
            mCurrentSelectedPosition =
savedInstanceState.getInt(STATE_SELECTED_POSITION);
            mFromSavedInstanceState = true;
        }

        // Select either the default item (0) or the last selected item.
        selectItem(mCurrentSelectedPosition);
    }

    @Override
```

```
    public void onActivityCreated(Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
        // Indicate that this fragment would like to influence the set of actions in
the action bar.
        setHasOptionsMenu(true);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, final ViewGroup container,
                             Bundle savedInstanceState) {
        mDrawerListView = (ListView) inflater.inflate(
                R.layout.fragment_navigation_drawer, container, false);
        mDrawerListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {
                selectItem(position);
            }
        });
        arrayTemp = getResources().getStringArray(R.array.drawer_elements);



//#############################################################################################
######
        final TypedArray typedArray =
getResources().obtainTypedArray(R.array.sections_icons);
        mDrawerListView.setAdapter(new ArrayAdapter<String>(
                getActionBar().getThemedContext(),
                android.R.layout.simple_list_item_activated_1,
                android.R.id.text1,
                arrayTemp
        ){
            @TargetApi(Build.VERSION_CODES.JELLY_BEAN_MR1)
            @Override
            public View getView(int position, View convertView, ViewGroup parent) {
                View v = super.getView(position,convertView,parent);
                int resourceId = typedArray.getResourceId(position,0);
                Drawable icon = getResources().getDrawable(resourceId);
                ((TextView)
v).setCompoundDrawablesRelativeWithIntrinsicBounds(icon,null,null,null);

                return v;
            }
        });

//#############################################################################################
######


        mDrawerListView.setItemChecked(mCurrentSelectedPosition, true);
        return mDrawerListView;
    }

    public boolean isDrawerOpen() {
        return mDrawerLayout != null &&
mDrawerLayout.isDrawerOpen(mFragmentContainerView);
    }

    /**
     * Users of this fragment must call this method to set up the navigation drawer
interactions.
     *
```

```
     * @param fragmentId   The android:id of this fragment in its activity's layout.
     * @param drawerLayout The DrawerLayout containing this fragment's UI.
     */
    public void setUp(int fragmentId, DrawerLayout drawerLayout) {
        mFragmentContainerView = getActivity().findViewById(fragmentId);
        mDrawerLayout = drawerLayout;

        // set a custom shadow that overlays the main content when the drawer opens
        mDrawerLayout.setDrawerShadow(R.drawable.drawer_shadow, GravityCompat.START);
        // set up the drawer's list view with items and click listener

        ActionBar actionBar = getActionBar();
        actionBar.setDisplayHomeAsUpEnabled(true);
        actionBar.setHomeButtonEnabled(true);

        // ActionBarDrawerToggle ties together the the proper interactions
        // between the navigation drawer and the action bar app icon.
        mDrawerToggle = new ActionBarDrawerToggle(
                getActivity(),                    /* host Activity */
                mDrawerLayout,                    /* DrawerLayout object */
                R.drawable.ic_drawer,             /* nav drawer image to replace 'Up'
caret */
                R.string.navigation_drawer_open,  /* "open drawer" description for
accessibility */
                R.string.navigation_drawer_close  /* "close drawer" description for
accessibility */
        ) {
            @Override
            public void onDrawerClosed(View drawerView) {
                super.onDrawerClosed(drawerView);
                if (!isAdded()) {
                    return;
                }

                getActivity().supportInvalidateOptionsMenu(); // calls
onPrepareOptionsMenu()
            }

            @Override
            public void onDrawerOpened(View drawerView) {
                super.onDrawerOpened(drawerView);
                if (!isAdded()) {
                    return;
                }

                if (!mUserLearnedDrawer) {
                    // The user manually opened the drawer; store this flag to prevent
auto-showing
                    // the navigation drawer automatically in the future.
                    mUserLearnedDrawer = true;
                    SharedPreferences sp = PreferenceManager
                            .getDefaultSharedPreferences(getActivity());
                    sp.edit().putBoolean(PREF_USER_LEARNED_DRAWER, true).commit();
                }

                getActivity().supportInvalidateOptionsMenu(); // calls
onPrepareOptionsMenu()
            }
        };

        // If the user hasn't 'learned' about the drawer, open it to introduce them to
the drawer,
        // per the navigation drawer design guidelines.
```

```
        if (!mUserLearnedDrawer && !mFromSavedInstanceState) {
            mDrawerLayout.openDrawer(mFragmentContainerView);
        }

        // Defer code dependent on restoration of previous instance state.
        mDrawerLayout.post(new Runnable() {
            @Override
            public void run() {
                mDrawerToggle.syncState();
            }
        });

        mDrawerLayout.setDrawerListener(mDrawerToggle);
    }

    private void selectItem(int position) {
        mCurrentSelectedPosition = position;
        if (mDrawerListView != null) {
            mDrawerListView.setItemChecked(position, true);
        }
        if (mDrawerLayout != null) {
            mDrawerLayout.closeDrawer(mFragmentContainerView);
        }
        if (mCallbacks != null) {
            mCallbacks.onNavigationDrawerItemSelected(position);
        }
    }

    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
        try {
            mCallbacks = (NavigationDrawerCallbacks) activity;
        } catch (ClassCastException e) {
            throw new ClassCastException("Activity must implement
NavigationDrawerCallbacks.");
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        mCallbacks = null;
    }

    @Override
    public void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
        outState.putInt(STATE_SELECTED_POSITION, mCurrentSelectedPosition);
    }

    @Override
    public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
        // Forward the new configuration the drawer toggle component.
        mDrawerToggle.onConfigurationChanged(newConfig);
    }

    @Override
    public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
        // If the drawer is open, show the global app actions in the action bar. See
also
        // showGlobalContextActionBar, which controls the top-left area of the action
```

```
bar.
        if (mDrawerLayout != null && isDrawerOpen()) {
            inflater.inflate(R.menu.global, menu);
            showGlobalContextActionBar();
        }
        super.onCreateOptionsMenu(menu, inflater);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (mDrawerToggle.onOptionsItemSelected(item)) {
            return true;
        }
        /*
        if (item.getItemId() == R.id.action_example) {
            Toast.makeText(getActivity(), "Example action.",
Toast.LENGTH_SHORT).show();
            return true;
        }
        */
        return super.onOptionsItemSelected(item);
    }

    /**
     * Per the navigation drawer design guidelines, updates the action bar to show the
global app
     * 'context', rather than just what's in the current screen.
     */
    private void showGlobalContextActionBar() {
        ActionBar actionBar = getActionBar();
        actionBar.setDisplayShowTitleEnabled(true);
        actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_STANDARD);
        actionBar.setTitle(R.string.app_name);
    }

    private ActionBar getActionBar() {
        return ((ActionBarActivity) getActivity()).getSupportActionBar();
    }

    /**
     * Callbacks interface that all activities using this fragment must implement.
     */
    public static interface NavigationDrawerCallbacks {
        /**
         * Called when an item in the navigation drawer is selected.
         */
        void onNavigationDrawerItemSelected(int position);
    }
}
```

8.1.2 Κώδικας XML

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

    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:choiceMode="singleChoice"
    android:divider="@android:color/transparent"
    android:dividerHeight="20dp"
```

```
    android:gravity="center_vertical"
    />
```

8.2 Κωδικας Για Main Activity Κλαση

8.2.1 Κώδικας Java

```
package com.example.dimitris.moneyger;


import android.app.Activity;
import android.net.Uri;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBar;
import android.support.v7.app.ActionBarActivity;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;

public class MainActivity extends ActionBarActivity
        implements NavigationDrawerFragment.NavigationDrawerCallbacks,
        HomeFragment.OnFragmentInteractionListener,
        StatisticsFragment.OnFragmentInteractionListener,
        AboutFragment.OnFragmentInteractionListener,
        SettingsFragment.OnFragmentInteractionListener,
        IncomeFragment.OnFragmentInteractionListener,

ExpensesFragment.OnFragmentInteractionListener,TotalFragment.OnFragmentInteractionList
ener{


    /**
     * Fragment managing the behaviors, interactions and presentation of the
navigation drawer.
     */
    private NavigationDrawerFragment mNavigationDrawerFragment;
    private String[] arrayTemp;

    /**
     * Used to store the last screen title. For use in {@link #restoreActionBar()}.
     */

    private CharSequence mTitle;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mNavigationDrawerFragment = (NavigationDrawerFragment)
                getSupportFragmentManager().findFragmentById(R.id.navigation_drawer);
        mTitle = getTitle();
```

```
    // Set up the drawer.
    mNavigationDrawerFragment.setUp(
            R.id.navigation_drawer,
            (DrawerLayout) findViewById(R.id.drawer_layout));
}

@Override
public void onNavigationDrawerItemSelected(int position) {
    // update the main content by replacing fragments
    Fragment objFragment = null;
    switch (position){
        case 0:
            objFragment = HomeFragment.newInstance(position + 1);
            break;

        case 1:
            objFragment = StatisticsFragment.newInstance(position + 1);
            break;

        case 2:
            objFragment = SettingsFragment.newInstance(position + 1);
            break;

        case 3:
            objFragment = AboutFragment.newInstance(position + 1);
            break;
    }
    onSectionAttached(position);
    FragmentManager fragmentManager = getSupportFragmentManager();
    fragmentManager.beginTransaction()
            .replace(R.id.container, objFragment)
            .commit();
}

//Getting the title for the Action Bar.
public void onSectionAttached(int number) {
    arrayTemp = getResources().getStringArray(R.array.drawer_elements);
    switch (number) {
        case 0:
            mTitle = arrayTemp[0];
            break;

        case 1:
            mTitle = arrayTemp[1];
            break;

        case 2:
            mTitle = arrayTemp[2];
            break;

        case 3:
            mTitle = arrayTemp[3];
            break;
    }
}

public void restoreActionBar() {
    ActionBar actionBar = getSupportActionBar();
    //actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_STANDARD);
    actionBar.setDisplayShowTitleEnabled(true);
    actionBar.setTitle(mTitle);
}
```

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        if (!mNavigationDrawerFragment.isDrawerOpen()) {
            // Only show items in the action bar relevant to this screen
            // if the drawer is not showing. Otherwise, let the drawer
            // decide what to show in the action bar.
            getMenuInflater().inflate(R.menu.main, menu);
            restoreActionBar();
            return true;
        }
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    /**
     * A placeholder fragment containing a simple view.
     */
    public static class PlaceholderFragment extends Fragment {
        /**
         * The fragment argument representing the section number for this
         * fragment.
         */
        private static final String ARG_SECTION_NUMBER = "section_number";

        /**
         * Returns a new instance of this fragment for the given section
         * number.
         */
        public static PlaceholderFragment newInstance(int sectionNumber) {
            PlaceholderFragment fragment = new PlaceholderFragment();
            Bundle args = new Bundle();
            args.putInt(ARG_SECTION_NUMBER, sectionNumber);
            fragment.setArguments(args);
            return fragment;
        }

        public PlaceholderFragment() {
        }

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                 Bundle savedInstanceState) {
            View rootView = inflater.inflate(R.layout.fragment_main, container,
false);
            return rootView;
        }
```

```
        @Override
        public void onAttach(Activity activity) {
            super.onAttach(activity);
            ((MainActivity) activity).onSectionAttached(
                    getArguments().getInt(ARG_SECTION_NUMBER));
        }
    }

    @Override
    public void onFragmentInteraction(Uri uri) {

    }

}
```

8.2.2 Κώδικας XML

```
<!-- A DrawerLayout is intended to be used as the top-level content view using
match_parent for both width and height to consume the full space available. -->
<android.support.v4.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:id="@+id/drawer_layout"
    android:layout_width="match_parent" android:layout_height="match_parent"
    tools:context=".MainActivity">

    <!-- As the main content view, the view below consumes the entire
         space available using match_parent in both dimensions. -->
    <FrameLayout android:id="@+id/container" android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <!-- android:layout_gravity="start" tells DrawerLayout to treat
         this as a sliding drawer on the left side for left-to-right
         languages and on the right side for right-to-left languages.
         If you're not building against API 17 or higher, use
         android:layout_gravity="left" instead. -->
    <!-- The drawer is given a fixed width in dp and extends the full height of
         the container. -->
    <fragment android:id="@+id/navigation_drawer"
        android:layout_width="@dimen/navigation_drawer_width"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:name="com.example.dimitris.moneyger.NavigationDrawerFragment"
        tools:layout="@layout/fragment_navigation_drawer"
        />

</android.support.v4.widget.DrawerLayout>
```

8.3 ΚΩΔΙΚΑΣ ΓΙΑ HOMEFRAGMENT ΚΛΑΣΗ

8.3.1 Κώδικας Java

```
package com.example.dimitris.moneyger;

import android.animation.AnimatorSet;
import android.app.Activity;
import android.app.AlertDialog;
import android.database.sqlite.SQLiteConstraintException;
import android.net.Uri;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentTabHost;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TabHost;
import android.widget.Toast;

import java.text.SimpleDateFormat;
import java.util.Calendar;



/**
 * A simple {@link Fragment} subclass.
 * Activities that contain this fragment must implement the
 * {@link HomeFragment.OnFragmentInteractionListener} interface
 * to handle interaction events.
 * Use the {@link HomeFragment#newInstance} factory method to
 * create an instance of this fragment.
 */
public class HomeFragment extends Fragment{

    private double totalSal = 0.0;
    private double totalInc = 0.0;
    private double totalRes = 0.0;


    private MonthlyIncome salary;
    private InsertSalary insert_salary;



    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";
    private static final String ARG_SEC_NUM = "section_number";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;

    //Getting the action for the Home Fragment
    private OnFragmentInteractionListener mListener;

    //TabHost manager
    private FragmentTabHost mTabHost;
```

```
    //Mandatory Constructor
    public HomeFragment() {
        // Required empty public constructor
    }

    // TODO: Rename and change types and number of parameters
    public static HomeFragment newInstance(int position) {
        HomeFragment fragment = new HomeFragment();
        Bundle args = new Bundle();
        args.putString(ARG_SEC_NUM, String.valueOf(position));
        //args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

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

        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

    /**
     * When the view is created a tab host menu with 3 tabs is displayed. The first
tab will
     * appear by default when the view is created. No gesture listeners are added due
to animation
     * lagging.
     */

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        mTabHost = new FragmentTabHost(getActivity());
        mTabHost.setup(getActivity(), getChildFragmentManager(), R.id.tabHost);

        mTabHost.addTab(mTabHost.newTabSpec("income").setIndicator("Income"),
                IncomeFragment.class, null);
        mTabHost.addTab(mTabHost.newTabSpec("expenses").setIndicator("Expenses"),
                ExpensesFragment.class, null);
        mTabHost.addTab(mTabHost.newTabSpec("total").setIndicator("Total"),
                TotalFragment.class, null);


        return mTabHost;
    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();
        mTabHost = null;
    }

    // TODO: Rename method, update argument and hook method into UI event
    public void onButtonPressed(Uri uri) {
        if (mListener != null) {
            mListener.onFragmentInteraction(uri);
        }
```

```
    }

    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
        try {
            mListener = (OnFragmentInteractionListener) activity;
        } catch (ClassCastException e) {
            throw new ClassCastException(activity.toString()
                    + " must implement OnFragmentInteractionListener");
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        mListener = null;
    }


    /**
     * This interface must be implemented by activities that contain this
     * fragment to allow an interaction in this fragment to be communicated
     * to the activity and potentially other fragments contained in that
     * activity.
     * <p/>
     * See the Android Training lesson <a href=
     * "http://developer.android.com/training/basics/fragments/communicating.html"
     * >Communicating with Other Fragments</a> for more information.
     */
    public interface OnFragmentInteractionListener {
        // TODO: Update argument type and name
        public void onFragmentInteraction(Uri uri);
    }

}
```

8.3.2 Κώδικας XML

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.dimitris.moneyger.HomeFragment">

    <!-- TODO: Update blank fragment layout -->

    <TabHost
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:id="@+id/tabHost"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true">

        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:orientation="vertical">
```

```
            <TabWidget
                android:id="@android:id/tabs"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"/>

            <FrameLayout
                android:id="@android:id/tabcontent"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content">

                <LinearLayout
                    android:id="@+id/income"
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:orientation="vertical"/>

                <LinearLayout
                    android:id="@+id/expenses"
                    android:layout_width="fill_parent"
                    android:layout_height="fill_parent"
                    android:orientation="vertical"/>

                <LinearLayout
                    android:id="@+id/total"
                    android:layout_width="fill_parent"
                    android:layout_height="fill_parent"
                    android:orientation="vertical"/>

            </FrameLayout>
        </LinearLayout>
    </TabHost>
</RelativeLayout>
```

8.4 ΚΩΔΙΚΑΣ ΓΙΑ INCOMEFRAGMENT ΚΛΑΣΗ

8.4.1 Κώδικας Java

```
package com.example.dimitris.moneyger;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.DatePickerDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.res.Configuration;
import android.content.res.TypedArray;
import android.net.Uri;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.text.InputType;
import android.text.method.KeyListener;
import android.view.LayoutInflater;
import android.view.View;
```

```
import android.view.ViewGroup;
import android.view.animation.AnimationUtils;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Locale;


/**
 * A simple {@link Fragment} subclass.
 * Activities that contain this fragment must implement the
 * {@link IncomeFragment.OnFragmentInteractionListener} interface
 * to handle interaction events.
 * Use the {@link IncomeFragment#//newInstance} factory method to
 * create an instance of this fragment.
 */
public class IncomeFragment extends Fragment {
    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;

    private OnFragmentInteractionListener mListener;


    //References for use in SharedPreferences class.
    private EditText dateText,accountText,amountText,categoryText;
    private EditText text;
    private String dateStr,accountStr,amountStr,categoryStr;


    //Variable saved in the DB to show if the current transaction is from Incomes or
Expenses.
    //private static final int incomeFrag = 0;
    private static  int incomeFrag = 0;

    //Constructor for the class that handles the DB.
    private dbHandler myDBHandler;

    /**
     * DOUBLE EXAMPLE
     */
    private double amountEntered;


    //Soft keyboard when amountTXT is clicked.
    private KeyListener keyListener;

    //Getting the calendar instance to use it in the datePicker.
    Calendar myCalendar = Calendar.getInstance();

    //Dialog builders for the popup windows.
```

```
    AlertDialog.Builder dialogBuilder;

    //Getting the selected dialogs from the popup windows.
    String accountSelected = null;
    String categorySelected = null;

    //Result from amountValidation method.
    int resultValidation;


    //Round Decimal class reference.
    private RoundDecimals twoDecimals;


    //Counter for save clicks.
    private int counter = 0;



//##########################################################################################
##########

//##########################################################################################
##########

//##########################################################################################
##########

    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
     *
     * @param param1 Parameter 1.
     * @param param2 Parameter 2.
     * @return A new instance of fragment IncomeFragment.
     */
    // TODO: Rename and change types and number of parameters
    public static IncomeFragment newInstance(String param1, String param2) {
        IncomeFragment fragment = new IncomeFragment();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }


    // Required empty public constructor
    public IncomeFragment() {

    }


    //OnCreate default method.
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        myDBHandler = new dbHandler(getActivity(),null,null,1);
        twoDecimals = new RoundDecimals();

        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
```

```
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }


    //OnCreateView method called after onCreate to inflate the fragment's layout.
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_income, container, false);
    }



    //Main method to initialize and utilize all the buttons and fields of the View
after it's
    //created using the onCreateView method.
    @Override
    public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

Toast.makeText(getActivity(),R.string.press_on_text,Toast.LENGTH_SHORT).show();
        setupFields();
    }


    //Initializing all the components of the current view.
    private void setupFields(){

        //Reference the buttons of Income Fragment.
        final Button saveButton    = (Button) getView().findViewById(R.id.save);
        Button discardButton = (Button) getView().findViewById(R.id.discard);


        //Reference the textViews of the Income Fragment.
        TextView dateTXT              = (TextView)
getView().findViewById(R.id.DateTxt);
        final TextView accountTXT     = (TextView)
getView().findViewById(R.id.AccountTxt);
        final TextView amountTXT      = (TextView)
getView().findViewById(R.id.AmountTxt);
        final TextView categoryTXT    = (TextView)
getView().findViewById(R.id.CategoryTxt);


        //Reference the editTexts of the Income Fragment.
        dateText        = (EditText) getView().findViewById(R.id.Date);
        accountText     = (EditText) getView().findViewById(R.id.Account);
        amountText      = (EditText) getView().findViewById(R.id.Amount);
        categoryText    = (EditText) getView().findViewById(R.id.Category);


        //Keyboard View
        keyListener = amountText.getKeyListener();
        amountText.setKeyListener(null);


        //Constructor for the class that handles the database.
        myDBHandler     =  new dbHandler(getActivity(),null,null,1);


        /**
```

```
         * Setting all editTexts not Editable
         */
        dateText.setKeyListener(null);
        accountText.setKeyListener(null);
        amountText.setKeyListener(null);
        categoryText.setKeyListener(null);

        /**
         * Setting the click listeners used by the textViews.
         * Setting the popup windows that will appear upon click.
         */

        //Setting the click listener for the Date textView.
        dateDefault();
        dateTXT.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                new DatePickerDialog(getActivity(), date, myCalendar
                        .get(Calendar.YEAR), myCalendar.get(Calendar.MONTH),
                        myCalendar.get(Calendar.DAY_OF_MONTH)).show();
            }
        });


        accountTXT.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                accountDialog();
            }
        });



//##########################################################################################
######

//##########################################################################################
######
        amountTXT.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                //Creating custom dailog boxes to get the amount of money the user
wants to enter.
                dialogBuilder = new AlertDialog.Builder(getActivity());
                dialogBuilder.setTitle(R.string.amountTitle);
                dialogBuilder.setMessage(R.string.amountPrompt);
                dialogBuilder.setCancelable(false);

                //For the user to enter money an EditText field is used with input
type phone
                //so as not to display letters or other special characters.
                final EditText money_entered = new EditText(getView().getContext());
                money_entered.setInputType(InputType.TYPE_CLASS_PHONE);

                dialogBuilder.setView(money_entered);


                dialogBuilder.setPositiveButton(R.string.ok, new
DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        String salarySTR = money_entered.getEditableText().toString();
```

```
                        if (salarySTR.contains(",")){
                            salarySTR = salarySTR.replace(',','.');
                        }

                        if (amountValidation(salarySTR) == 1){
                            amountEntered = Double.parseDouble(salarySTR);
                            amountEntered =
twoDecimals.roundTwoDecimals(amountEntered);
                            amountText.setText(salarySTR);
                        }
                    }
                });

                dialogBuilder.setNegativeButton(R.string.cancel, new DialogInterface.
                        OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        dialog.cancel();
                    }
                });

                AlertDialog alert = dialogBuilder.create();
                alert.show();

            }
        });


//#########################################################################################
######

//#########################################################################################
######


        //By pressing on the category TextView a list of categories is displayed from
which the
        //use can only choose one at the time.
        categoryTXT.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                categoryDialog();
            }
        });


        /**
         * Setting the click listeners used by the buttons of the view.
         */
        saveButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                dateStr         = dateText.getText().toString();
                accountStr      = accountText.getText().toString();
                amountStr       = amountText.getText().toString();
                categoryStr     = categoryText.getText().toString();


                //Popup message in case where some of the fields are not filled.
                if (dateStr.equals("") || amountStr.equals("") ||
accountStr.equals("") ||
                        categoryStr.equals("")){
```

```
Toast.makeText(getActivity(),R.string.fields_required,Toast.LENGTH_SHORT)
                            .show();
                }
                //If all textFields are filled proceed(BUG FIXED).
                else {
                    if (amountStr.contains(",")){
                        amountStr.replace(',','.');
                    }

                    resultValidation = amountValidation(amountStr);


                    /**
                     * 2 decimal digits for all doubles.
                     */

                    if (resultValidation == 1) {
                        amountEntered = Double.parseDouble(amountStr);
                        amountEntered = twoDecimals.roundTwoDecimals(amountEntered);

                        Transactions transaction = new Transactions(dateStr,
categoryStr,
                                accountStr, amountEntered, incomeFrag,0.0);

                        myDBHandler.addRow(transaction);

                        Toast.makeText(getActivity(),R.string.saved_all,
                                Toast.LENGTH_SHORT).show();
                    }
                    else{
                        amountText.setText("");
                    }
                }
            }
        });

        //Setting the action performed by the DISCARD button on main screen.
        discardButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                clearData();
            }
        });

    }


/*********************************************************************************
*********/
    /**
     * Creating the methods used in this fragment.
     */


    //Creating the datePicker popup window.
    final DatePickerDialog.OnDateSetListener date = new
DatePickerDialog.OnDateSetListener() {
        @Override
        public void onDateSet(DatePicker view, int year, int monthOfYear,
                              int dayOfMonth) {
            // TODO Auto-generated method stub
            myCalendar.set(Calendar.YEAR, year);
            myCalendar.set(Calendar.MONTH, monthOfYear);
```

```
            myCalendar.set(Calendar.DAY_OF_MONTH, dayOfMonth);
            updateLabel();
        }
    };


    //Changing the text of the appropriate TextView.
    private void updateLabel() {
        String myFormat = "yyyy/MM/dd";
        SimpleDateFormat sdf = new SimpleDateFormat(myFormat, Locale.US);
        dateText.setText(sdf.format(myCalendar.getTime()));
    }

    //Creating the DISCARD Button for clearing all fields.
    private void clearData(){
        TypedArray arrayPtr = getResources().obtainTypedArray(R.array.arrayID);


        for (int counter = 0; counter < arrayPtr.length(); counter ++){
            text = (EditText)
getView().findViewById(arrayPtr.getResourceId(counter,0));
            text.setText("");
        }

    }


    //The Dialog box used for the account menu.
    private void accountDialog(){
        final TypedArray accountIncDial =
getResources().obtainTypedArray(R.array.accountIncDialog);

        dialogBuilder = new AlertDialog.Builder(getActivity());


        dialogBuilder.setTitle(R.string.account_select);
        dialogBuilder.setSingleChoiceItems(R.array.accountIncDialog, -1,
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        Toast.makeText(getActivity(), R.string.account_select_success,
                                Toast.LENGTH_SHORT).show();
                        accountText.setText(accountIncDial.getText(which));
                        dialog.dismiss();
                    }
                });
        AlertDialog accountSelectDialog = dialogBuilder.create();
        accountSelectDialog.show();

    }


    //The Dialog box used for the category menu.
    private void categoryDialog(){
        final TypedArray categoryIncDial = getResources().obtainTypedArray(R.array.
                categoryIncDialog);

        dialogBuilder = new AlertDialog.Builder(getActivity());

        dialogBuilder.setTitle(R.string.category_select);
        dialogBuilder.setSingleChoiceItems(R.array.categoryIncDialog, -1,
                new DialogInterface.OnClickListener() {
                    @Override
```

```
                    public void onClick(DialogInterface dialog, int which) {
                        Toast.makeText(getActivity(),
R.string.category_select_success,
                                Toast.LENGTH_SHORT).show();
                        categoryText.setText(categoryIncDial.getText(which));
                        dialog.dismiss();
                    }
                });
        AlertDialog accountSelectDialog = dialogBuilder.create();
        accountSelectDialog.show();
    }


    //Checking if the input amount is valid.
    private int amountValidation(String str){
        for (counter=0; counter<str.length();counter++){
            if (!Character.isDigit(str.charAt(counter))){
                if (str.charAt(counter) == '.' || str.charAt(counter) == ','){
                    continue;
                }
                else {
                    Toast.makeText(getActivity(),R.string.invalid_input,
Toast.LENGTH_SHORT).show();
                    return 0;
                }
            }
        }

        return 1;
    }


    //Displaying the current date in date text field.
    private void dateDefault(){
        Calendar cal = Calendar.getInstance();
        SimpleDateFormat df = new SimpleDateFormat("yyyy/MM/dd");
        String formattedDate = df.format(cal.getTime());
        dateText.setText(formattedDate);
    }



/*************************************************************************************
*********/


    // TODO: Rename method, update argument and hook method into UI event
    public void onButtonPressed(Uri uri) {
        if (mListener != null) {
            mListener.onFragmentInteraction(uri);
        }
    }

    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
        try {
            mListener = (OnFragmentInteractionListener) activity;
        } catch (ClassCastException e) {
            throw new ClassCastException(activity.toString()
                    + " must implement OnFragmentInteractionListener");
        }
    }
```

```
    @Override
    public void onDetach() {
        super.onDetach();
        mListener = null;
    }

    /**
     * This interface must be implemented by activities that contain this
     * fragment to allow an interaction in this fragment to be communicated
     * to the activity and potentially other fragments contained in that
     * activity.
     * <p/>
     * See the Android Training lesson <a href=
     * "http://developer.android.com/training/basics/fragments/communicating.html"
     * >Communicating with Other Fragments</a> for more information.
     */
    public interface OnFragmentInteractionListener {
        // TODO: Update argument type and name
        public void onFragmentInteraction(Uri uri);
    }

}
```

8.4.2 Κώδικας XML

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.dimitris.moneyger.IncomeFragment">

    <TableLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="left|top">

        <TableRow
            android:layout_gravity="left">
            <TextView
                android:layout_width="wrap_content"
                android:text="@string/dateField"
                android:id="@+id/DateTxt"
                android:textSize="25sp"
                android:layout_marginLeft="20dp"/>
            <EditText
                android:id="@+id/Date"
                android:layout_height="wrap_content"
                android:layout_marginLeft="20dip"
                android:layout_marginRight="20dip"
                android:layout_weight="1"
                android:background="@drawable/back"
                android:inputType="date">
            </EditText>

        </TableRow>


        <TableRow
            android:layout_gravity="left">
            <TextView
```

```
            android:layout_width="wrap_content"
            android:text="@string/accountField"
            android:id="@+id/AccountTxt"
            android:textSize="25sp"
            android:paddingTop="60dp"
            android:layout_marginLeft="20dp"/>

        <EditText
            android:id="@+id/Account"
            android:layout_height="wrap_content"
            android:layout_marginLeft="20dip"
            android:layout_marginRight="20dip"
            android:layout_weight="1"
            android:background="@drawable/back"
            android:inputType="text">
        </EditText>

    </TableRow>

    <TableRow
        android:layout_gravity="left">
        <TextView
            android:layout_width="wrap_content"
            android:text="@string/amountField"
            android:id="@+id/AmountTxt"
            android:textSize="25sp"
            android:paddingTop="60dp"
            android:layout_marginLeft="20dp"/>

        <EditText
            android:id="@+id/Amount"
            android:layout_height="wrap_content"
            android:layout_marginLeft="20dip"
            android:layout_marginRight="20dip"
            android:layout_weight="1"
            android:background="@drawable/back"
            android:inputType="numberDecimal"
            android:digits="1234567890.,"
            android:imeOptions="actionNext">
        </EditText>

    </TableRow>



    <TableRow
        android:layout_gravity="left">
        <TextView
            android:layout_width="wrap_content"
            android:text="@string/categoryField"
            android:id="@+id/CategoryTxt"
            android:paddingTop="60dp"
            android:textSize="25sp"
            android:layout_marginLeft="20dp"/>

        <EditText
            android:id="@+id/Category"
            android:layout_height="wrap_content"
            android:layout_marginLeft="20dip"
            android:layout_marginRight="20dip"
            android:layout_weight="1"
            android:background="@drawable/back"
            android:inputType="text">
```

```
            </EditText>
        </TableRow>
    </TableLayout>

    <Button
        android:text="@string/saveButton"
        android:clickable="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/save"
        android:layout_gravity="left|bottom"/>

    <Button
        android:text="@string/discardButton"
        android:clickable="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/discard"
        android:layout_gravity="right|bottom"/>

</FrameLayout>
```

8.5 ΚΩΔΙΚΑΣ ΓΙΑ EXPENSESFRAGMENT ΚΛΑΣΗ

8.5.1 Κώδικας Java

```
package com.example.dimitris.moneyger;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.DatePickerDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.res.Configuration;
import android.content.res.TypedArray;
import android.net.Uri;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.text.InputType;
import android.text.method.KeyListener;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Locale;



/**
 * A simple {@link Fragment} subclass.
```

```
 * Activities that contain this fragment must implement the
 * {@link ExpensesFragment.OnFragmentInteractionListener} interface
 * to handle interaction events.
 * Use the {@link ExpensesFragment#newInstance} factory method to
 * create an instance of this fragment.
 */
public class ExpensesFragment extends Fragment {

    //Variable saved in the DB to show if the current transaction is from Incomes or
Expenses.
    private static final int incomeFrag = 1;


    //Constructor for the class that handles the DB.
    private dbHandler myDBHandler;


    //Soft keyboard when amountTXT is clicked.
    private KeyListener keyListener;


    //Getting the calendar instance to use it in the datePicker.
    Calendar myCalendar = Calendar.getInstance();


    //Dialog builders for the popup windows.
    AlertDialog.Builder dialogBuilder;


    //Round Decimal class reference.
    private RoundDecimals twoDecimals;


    //References for use in SharedPreferences class.
    private EditText dateText,accountText,amountText,categoryText;
    private EditText text;
    private String dateStr,accountStr,amountStr,categoryStr;
    private double amountEntered;


    //Result from amountValidation method.
    int resultValidation;


    //Counter for save clicks.
    private int counter = 0;


    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;

    private OnFragmentInteractionListener mListener;

    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
     *
```

```
     * @param param1 Parameter 1.
     * @param param2 Parameter 2.
     * @return A new instance of fragment ExpensesFragment.
     */
    // TODO: Rename and change types and number of parameters
    public static ExpensesFragment newInstance(String param1, String param2) {
        ExpensesFragment fragment = new ExpensesFragment();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    public ExpensesFragment() {
        // Required empty public constructor
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        myDBHandler = new dbHandler(getActivity(),null,null,1);
        twoDecimals = new RoundDecimals();
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }


    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_expenses, container, false);
    }


    //Main method to initialize and utilize all the buttons and fields of the View
after it's
    //created using the onCreateView method.
    @Override
    public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

Toast.makeText(getActivity(),R.string.press_on_text,Toast.LENGTH_SHORT).show();
        initializeFields();
    }


    private void initializeFields(){

        //Reference the buttons of Income Fragment.
        Button saveButton    = (Button) getView().findViewById(R.id.save2);
        Button discardButton = (Button) getView().findViewById(R.id.discard2);

        //Reference the textViews of the Income Fragment.
        TextView dateTXT              = (TextView)
getView().findViewById(R.id.DateTxt);
        final TextView accountTXT     = (TextView)
getView().findViewById(R.id.AccountTxt);
        final TextView amountTXT      = (TextView)
getView().findViewById(R.id.AmountTxt);
```

```
        final TextView categoryTXT    = (TextView)
getView().findViewById(R.id.CategoryTxt);


        //Reference the editTexts of the Income Fragment.
        dateText        = (EditText) getView().findViewById(R.id.Date);
        accountText     = (EditText) getView().findViewById(R.id.Account);
        amountText      = (EditText) getView().findViewById(R.id.Amount);
        categoryText    = (EditText) getView().findViewById(R.id.Category);


        //Keyboard View
        keyListener = amountText.getKeyListener();
        amountText.setKeyListener(null);


        //Constructor for the class that handles the database.
        myDBHandler     =  new dbHandler(getActivity(),null,null,1);


        /**
         * Setting all editTexts not Editable
         */
        dateText.setKeyListener(null);
        accountText.setKeyListener(null);
        amountText.setKeyListener(null);
        categoryText.setKeyListener(null);


        /**
         * Setting the click listeners used by the textViews.
         * Setting the popup windows that will appear upon click.
         */

        //Setting the click listener for the Date textView.
        dateDefault();
        dateTXT.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                new DatePickerDialog(getActivity(), date, myCalendar
                        .get(Calendar.YEAR), myCalendar.get(Calendar.MONTH),
                        myCalendar.get(Calendar.DAY_OF_MONTH)).show();
            }
        });


        accountTXT.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                accountDialog();
            }
        });



//#############################################################################################
######

//#############################################################################################
######
        amountTXT.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
                dialogBuilder = new AlertDialog.Builder(getActivity());
                dialogBuilder.setTitle(R.string.amountTitle);
                dialogBuilder.setMessage(R.string.amountPrompt);
                dialogBuilder.setCancelable(false);


                final EditText money_entered = new EditText(getView().getContext());
                money_entered.setInputType(InputType.TYPE_CLASS_PHONE);

                dialogBuilder.setView(money_entered);


                dialogBuilder.setPositiveButton(R.string.ok, new
DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        String salarySTR = money_entered.getEditableText().toString();

                        if (salarySTR.contains(",")){
                            salarySTR = salarySTR.replace(',','.');
                        }

                        if (amountValidation(salarySTR) == 1){
                            amountEntered = Double.parseDouble(salarySTR);
                            amountEntered =
twoDecimals.roundTwoDecimals(amountEntered);
                            amountText.setText(salarySTR);
                        }
                    }
                });

                dialogBuilder.setNegativeButton(R.string.cancel,new DialogInterface.
                        OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        dialog.cancel();
                    }
                });

                AlertDialog alert = dialogBuilder.create();
                alert.show();

            }
        });


//#############################################################################################
######

//#############################################################################################
######


        categoryTXT.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                categoryDialog();
            }
        });


        /**
         * Setting the click listeners used by the buttons of the view.
         */
```

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

                dateStr         = dateText.getText().toString();
                accountStr      = accountText.getText().toString();
                amountStr       = amountText.getText().toString();
                categoryStr     = categoryText.getText().toString();


                //Popup message in case where some of the fields are not filled.
                if (dateStr.equals("") || amountStr.equals("") ||
accountStr.equals("") ||
                        categoryStr.equals("")){

Toast.makeText(getActivity(),R.string.fields_required,Toast.LENGTH_SHORT)
                            .show();
                }
                //If all textFields are filled proceed(BUG FIXED).
                else {
                    if (amountStr.contains(",")){
                        amountStr.replace(',','.');
                    }

                    resultValidation = amountValidation(amountStr);


                    /**
                     * 2 decimal digits for all doubles.
                     */

                    if (resultValidation == 1) {
                        amountEntered = Double.parseDouble(amountStr);
                        amountEntered = twoDecimals.roundTwoDecimals(amountEntered);

                        Transactions transaction = new Transactions(dateStr,
categoryStr,
                                accountStr, amountEntered, incomeFrag,0.0);

                        myDBHandler.addRow(transaction);

                        Toast.makeText(getActivity(),R.string.saved_all,
                                Toast.LENGTH_SHORT).show();
                    }
                    else{
                        amountText.setText("");
                    }
                }
            }
        });


        //Setting the action performed by the DISCARD button on main screen.
        discardButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                clearData();
            }
        });

    }
```

```
/*********************************************************************************
*********/
    /**
     * Creating the methods used in this fragment.
     */


    //Creating the datePicker popup window.
    final DatePickerDialog.OnDateSetListener date = new
DatePickerDialog.OnDateSetListener() {
        @Override
        public void onDateSet(DatePicker view, int year, int monthOfYear,
                              int dayOfMonth) {
            // TODO Auto-generated method stub
            myCalendar.set(Calendar.YEAR, year);
            myCalendar.set(Calendar.MONTH, monthOfYear);
            myCalendar.set(Calendar.DAY_OF_MONTH, dayOfMonth);
            updateLabel();
        }
    };


    //Changing the text of the appropriate TextView.
    private void updateLabel() {
        String myFormat = "yyyy/MM/dd";
        SimpleDateFormat sdf = new SimpleDateFormat(myFormat, Locale.US);
        dateText.setText(sdf.format(myCalendar.getTime()));
    }


    //Creating the DISCARD Button for clearing all fields.
    private void clearData(){
        TypedArray arrayPtr = getResources().obtainTypedArray(R.array.arrayID);


        for (int counter = 0; counter < arrayPtr.length(); counter ++){
            text = (EditText)
getView().findViewById(arrayPtr.getResourceId(counter,0));
            text.setText("");
        }

    }


    //The Dialog box used for the account menu.
    private void accountDialog(){
        final TypedArray accountExpDial =
getResources().obtainTypedArray(R.array.accountExpDialog);

        dialogBuilder = new AlertDialog.Builder(getActivity());


        dialogBuilder.setTitle(R.string.account_select);
        dialogBuilder.setSingleChoiceItems(R.array.accountExpDialog, -1,
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        Toast.makeText(getActivity(), R.string.account_select_success,
                                Toast.LENGTH_SHORT).show();
                        accountText.setText(accountExpDial.getText(which));
                        dialog.dismiss();
                    }
```

```
                });
        AlertDialog accountSelectDialog = dialogBuilder.create();
        accountSelectDialog.show();

    }


    //The Dialog box used for the category menu.
    private void categoryDialog(){
        final TypedArray categoryExpDial = getResources().obtainTypedArray(R.array.
                categoryExpDialog);

        dialogBuilder = new AlertDialog.Builder(getActivity());

        dialogBuilder.setTitle(R.string.category_select);
        dialogBuilder.setSingleChoiceItems(R.array.categoryExpDialog, -1,
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        Toast.makeText(getActivity(),R.string.category_select_success,
                                Toast.LENGTH_SHORT).show();
                        categoryText.setText(categoryExpDial.getText(which));
                        dialog.dismiss();
                    }
                });
        AlertDialog accountSelectDialog = dialogBuilder.create();
        accountSelectDialog.show();
    }


    //Checking if the input amount is valid.
    private int amountValidation(String str){
        for (counter=0; counter<str.length();counter++){
            if (!Character.isDigit(str.charAt(counter))){
                if (str.charAt(counter) == '.' || str.charAt(counter) == ','){
                    continue;
                }
                else {
                    Toast.makeText(getActivity(), R.string.invalid_input,
Toast.LENGTH_SHORT)
                            .show();
                    return 0;
                }
            }
        }

        return 1;
    }


    //Displaying the current date in date text field.
    private void dateDefault(){
        Calendar cal = Calendar.getInstance();
        SimpleDateFormat df = new SimpleDateFormat("yyyy/MM/dd");
        String formattedDate = df.format(cal.getTime());
        dateText.setText(formattedDate);
    }



    // TODO: Rename method, update argument and hook method into UI event
    public void onButtonPressed(Uri uri) {
        if (mListener != null) {
```

```
            mListener.onFragmentInteraction(uri);
        }
    }

    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
        try {
            mListener = (OnFragmentInteractionListener) activity;
        } catch (ClassCastException e) {
            throw new ClassCastException(activity.toString()
                    + " must implement OnFragmentInteractionListener");
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        mListener = null;
    }

    /**
     * This interface must be implemented by activities that contain this
     * fragment to allow an interaction in this fragment to be communicated
     * to the activity and potentially other fragments contained in that
     * activity.
     * <p/>
     * See the Android Training lesson <a href=
     * "http://developer.android.com/training/basics/fragments/communicating.html"
     * >Communicating with Other Fragments</a> for more information.
     */
    public interface OnFragmentInteractionListener {
        // TODO: Update argument type and name
        public void onFragmentInteraction(Uri uri);
    }

}
```

8.5.2 Κώδικας XML

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.dimitris.moneyger.ExpensesFragment">

    <TableLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="left|top">

        <TableRow
            android:layout_gravity="left">
            <TextView
                android:layout_width="wrap_content"
                android:text="@string/dateField"
                android:id="@+id/DateTxt"
                android:textSize="25sp"
                android:layout_marginLeft="20dp"/>
            <EditText
```

```
            android:id="@+id/Date"
            android:layout_height="wrap_content"
            android:layout_marginLeft="20dip"
            android:layout_marginRight="20dip"
            android:layout_weight="1"
            android:background="@drawable/back"
            android:inputType="date">
        </EditText>

    </TableRow>


    <TableRow
        android:layout_gravity="left">
        <TextView
            android:layout_width="wrap_content"
            android:text="@string/accountField"
            android:id="@+id/AccountTxt"
            android:textSize="25sp"
            android:paddingTop="60dp"
            android:layout_marginLeft="20dp"/>

        <EditText
            android:id="@+id/Account"
            android:layout_height="wrap_content"
            android:layout_marginLeft="20dip"
            android:layout_marginRight="20dip"
            android:layout_weight="1"
            android:background="@drawable/back"
            android:inputType="text">
        </EditText>

    </TableRow>

    <TableRow
        android:layout_gravity="left">
        <TextView
            android:layout_width="wrap_content"
            android:text="@string/amountField"
            android:id="@+id/AmountTxt"
            android:textSize="25sp"
            android:paddingTop="60dp"
            android:layout_marginLeft="20dp"/>

        <EditText
            android:id="@+id/Amount"
            android:layout_height="wrap_content"
            android:layout_marginLeft="20dip"
            android:layout_marginRight="20dip"
            android:layout_weight="1"
            android:background="@drawable/back"
            android:inputType="number">
        </EditText>

    </TableRow>



    <TableRow
        android:layout_gravity="left">
        <TextView
            android:layout_width="wrap_content"
            android:text="@string/categoryField"
```

```
                android:id="@+id/CategoryTxt"
                android:paddingTop="60dp"
                android:textSize="25sp"
                android:layout_marginLeft="20dp"/>

            <EditText
                android:id="@+id/Category"
                android:layout_height="wrap_content"
                android:layout_marginLeft="20dip"
                android:layout_marginRight="20dip"
                android:layout_weight="1"
                android:background="@drawable/back"
                android:inputType="text">
            </EditText>
        </TableRow>
    </TableLayout>

    <Button
        android:text="@string/saveButton"
        android:clickable="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/save2"
        android:layout_gravity="left|bottom"/>

    <Button
        android:text="@string/discardButton"
        android:clickable="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/discard2"
        android:layout_gravity="right|bottom"/>
</FrameLayout>
```

8.6 ΚΩΔΙΚΑΣ ΓΙΑ TOTALFRAGMENT ΚΛΑΣΗ

8.6.1 Κώδικας Java

```
package com.example.dimitris.moneyger;

import android.app.Activity;
import android.net.Uri;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.text.method.ScrollingMovementMethod;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ToggleButton;

import java.text.SimpleDateFormat;
```

```java
import java.util.Calendar;


/**
 * A simple {@link Fragment} subclass.
 * Activities that contain this fragment must implement the
 * {@link TotalFragment.OnFragmentInteractionListener} interface
 * to handle interaction events.
 * Use the {@link TotalFragment#newInstance} factory method to
 * create an instance of this fragment.
 */
public class TotalFragment extends Fragment {
    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;

    private OnFragmentInteractionListener mListener;

    //References for use in SharedPreferences class.
    private TextView loadArea;
    private TextView balanceView;
    private TextView incomeView;
    private TextView expensesView;

    //References for the toggleButton.
    private ToggleButton togButton;


    //Connection to the db's constructor.
    private dbHandler handler;


    //Variable for the distinction between Income and Expenses view(toggleButton).
    private int view;


    //Constructor reference for the RoundDecimals class.
    private RoundDecimals twoDecimals;


    //Constructor reference for the BalanceCalculator class.
    private BalanceCalculator calculatorBal;


    //Variables for the total balance,total income and total expenses.
    private double totalBal = 0.0;
    private double totalInc = 0.0;
    private double totalExp = 0.0;
    private double totalSal = 0.0;


    //String values for the comma bugFIX in the income and expenses textViews.
    private String incomeStr,expensesStr;


    //Total Income with salary increase.
    private MonthlyIncome monthlySalary;
```

```
    //Inserting the salary into the db.
    private InsertSalary insert_salary;


    private int check;

    private int counter;

    private int date;

//#########################################################################################
##########




    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
     *
     * @param param1 Parameter 1.
     * @param param2 Parameter 2.
     * @return A new instance of fragment TotalFragment.
     */
    // TODO: Rename and change types and number of parameters
    public static TotalFragment newInstance(String param1, String param2) {
        TotalFragment fragment = new TotalFragment();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    public TotalFragment() {
        // Required empty public constructor
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        twoDecimals = new RoundDecimals();
        counter = 0;
        //home = new HomeFragment();
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_total, container, false);
    }

    // TODO: Rename method, update argument and hook method into UI event
    public void onButtonPressed(Uri uri) {
        if (mListener != null) {
            mListener.onFragmentInteraction(uri);
        }
```

```
    }

    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
        try {
            mListener = (OnFragmentInteractionListener) activity;
        } catch (ClassCastException e) {
            throw new ClassCastException(activity.toString()
                    + " must implement OnFragmentInteractionListener");
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        mListener = null;
    }

    /**
     * This interface must be implemented by activities that contain this
     * fragment to allow an interaction in this fragment to be communicated
     * to the activity and potentially other fragments contained in that
     * activity.
     * <p/>
     * See the Android Training lesson <a href=
     * "http://developer.android.com/training/basics/fragments/communicating.html"
     * >Communicating with Other Fragments</a> for more information.
     */
    public interface OnFragmentInteractionListener {
        // TODO: Update argument type and name
        public void onFragmentInteraction(Uri uri);
    }


    @Override
    public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
        handler = new dbHandler(getActivity(),null,null,1);
        calculatorBal = new BalanceCalculator();
        monthlySalary = new MonthlyIncome();
        insert_salary = new InsertSalary();
        super.onViewCreated(view, savedInstanceState);
        setupFields();
    }


    //Determine the click actions for all the buttons in this fragment.
    private void setupFields(){
        Button loadButton = (Button) getView().findViewById(R.id.loadButton);
        Button clearButton = (Button) getView().findViewById(R.id.clearButton);

        togButton = (ToggleButton) getView().findViewById(R.id.toggleButton);

        loadArea = (TextView) getView().findViewById(R.id.loadText);

        /**
         * Setting the actions performed and results displayed in the framelayout of
the fragment.
         **/

        //Creating the reference for the textViews.
        balanceView  = (TextView) getView().findViewById(R.id.textViewBalanceBox);
        incomeView   = (TextView) getView().findViewById(R.id.textViewIncomeBox);
```

```
        expensesView = (TextView) getView().findViewById(R.id.textViewExpensesBox);


        //Getting the text from the textViews.
        incomeStr    = incomeView.getText().toString();
        expensesStr  = expensesView.getText().toString();


        /**Getting only the day from the current date to use it as a check in order to
         * determine if the month has changed (and it is the first day of the new
month) and add the
         * salary to the total income.
         */
        Calendar calendarView = Calendar.getInstance();
        String formattedDate =
String.valueOf(calendarView.get(Calendar.DAY_OF_MONTH));
        String currentMonth   = String.valueOf(calendarView.get(Calendar.MONTH) + 1);



        //Income textView box.
        totalInc = handler.totalIncome();
        //System.out.println("TOTAL INCOME -----> " +totalInc);

        /**
         * BUG FIXED: When a user used a greek keyboard layout and tried to enter a
new amount then
         * the application would crash as in greek layout keyboard, '.' is taken as
',' which was
         * not recognised. So i managed to find all the occurrences of character ','
and transform
         * it into character '.'.
         */
        if (incomeStr.contains(",")){
            incomeStr.replace(',','.');
        }

        //Having the income amount displayed allowing two decimals.
        totalInc = twoDecimals.roundTwoDecimals(totalInc);

        //Setting the view of the TextView to display the current income.
        incomeView.setText(String.valueOf(totalInc));



//#########################################################################################
######

//#########################################################################################
######

        //Getting the month of the current Calendar instance.
        SimpleDateFormat df = new SimpleDateFormat("M");
        String dateSTR = df.format(calendarView.getTime());

        //Getting the month's integer value.
        date = Integer.parseInt(dateSTR);

        //The counter is saved into a table in the database so it's value IS NOT LOST.
In the
        //beginning counter's value is set to 0 showing it's the first time we access
this PART
        //of the fragment.
```

```
        handler.saveCounter(counter,date);

        //If this is the FIRST day of the month counter is increased and it's NEW
value is saved in
        //the database using the same method as before.
        if (formattedDate.equals("1")){
            counter++;
            String str = handler.counterToString();
            handler.saveCounter(counter,date);

            //This is a variable used to show if there is a new month.To wit it is
used in order to
            //see if the month has changed. Using the checkRepeat method,which takes
the INTEGER
            //value of the month(current) as a parameter, we either get -1 if the
months in the db
            //are all <= with the current or a value different to -1 if there is even
one month in
            //the database whose value(integer) is > than the current month.
            check = handler.checkRepeat(Integer.parseInt(currentMonth));

            //Using the getCounter method we obtain all the entries from the counter
table which
            //include the value of the counter and the current month value. Using this
method that
            //stores counter values and month integer values i have managed to
increase the salary
            //once every month AND NOT every time the view was destroyed.
            /**
             * MAJOR BUG FIX: Without the implementation below,every time the user
changed view in
             * the fragment,say from Total Fragment to Income Fragment or simply
anywhere else,then
             * the income would increase by the salary he had stored in the Settings
Fragment.Now
             * the salary is ONLY changed when the month is changed and is it's FIRST
DAY.
             */
            if (check != -1 && (handler.getCounter(date) <= 1 &&
handler.getCounter(date) > 0)){
                totalSal = handler.getSalary();
                handler.totalSaveRepeat(totalSal);

            }


        }
        totalSal = totalInc + handler.gettotalSaveRepeat();


//##########################################################################################
######

//##########################################################################################
######



        //Expenses textView box.
        totalExp = handler.totalExpenses();
        if (expensesStr.contains(",")){
            expensesStr.replace(',','.');
        }
```

```
        totalExp = twoDecimals.roundTwoDecimals(totalExp);

        expensesView.setText(String.valueOf(totalExp));


        //Balance textView box.
        totalBal = calculatorBal.totalBalance(totalSal,totalExp);
        totalBal = twoDecimals.roundTwoDecimals(totalBal);
        balanceView.setText(String.valueOf(totalBal));

        if (totalBal < 0.0){
            Toast.makeText(getActivity(),R.string.great_expenses,
                    Toast.LENGTH_SHORT).show();
        }


/*****************************************************************************************
*****
         *
         */

        //Loading area for the transactions' history.
        loadArea.setMovementMethod(new ScrollingMovementMethod());


        //Actions when load button is clicked.
        loadButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                displayEntries();
            }
        });


        //Actions when the clear button is clicked.
        clearButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                loadArea.setText("");
            }
        });


        //Actions when the toggle button is clicked.
        togButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                viewChanger();
            }
        });


    }


    //Displaying entries according to whether expenses or income view is set.
    public void displayEntries(){
        togButton = (ToggleButton) getView().findViewById(R.id.toggleButton);

        if (togButton.getText().toString().equals("Expenses")) {
            view = 0;
            String entry = handler.dbToString(view);
            loadArea.setText(entry);
```

```
        }
        else {
            view = 1;
            String entry = handler.dbToString(view);
            loadArea.setText(entry);

        }

    }


    //Toast messages for the change of entries that will appear.
    public void viewChanger(){
        if (togButton.getText().toString().equals("Expenses")) {
            Toast.makeText(getActivity(), "View is changed to Income.",
Toast.LENGTH_SHORT)
                    .show();
            displayEntries();
        }
        else {
            Toast.makeText(getActivity(), "View is changed to Expenses.",
Toast.LENGTH_SHORT)
                    .show();
            displayEntries();
        }
    }

}
```

8.6.2 Κώδικας XML

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.dimitris.moneyger.TotalFragment">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:maxLines="5"
        android:scrollbars="vertical"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:id="@+id/loadText"
        android:layout_gravity="center_horizontal|top" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/loadButton"
        android:id="@+id/loadButton"
        android:layout_gravity="left|bottom" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/clearButton"
        android:id="@+id/clearButton"
        android:layout_gravity="right|bottom" />
```

```
<ToggleButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/toggleButton"
    android:id="@+id/toggleButton"
    android:textOff="@string/textOff"
    android:textOn="@string/textOn"
    android:layout_gravity="center_horizontal|bottom" />

<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="150dp"
    android:layout_gravity="center">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="@string/totalIncome"
        android:id="@+id/textViewInc"
        android:layout_gravity="left|top" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="@string/totalExpenses"
        android:id="@+id/textViewExp"
        android:layout_gravity="left|center_vertical" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="@string/totalBalance"
        android:id="@+id/textViewBal"
        android:layout_gravity="left|bottom" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="@string/totalTextInc"
        android:background="@drawable/back"
        android:id="@+id/textViewIncomeBox"
        android:layout_gravity="right|top" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="@string/totalTextExp"
        android:background="@drawable/back"
        android:id="@+id/textViewExpensesBox"
        android:layout_gravity="right|center_vertical" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="@string/totalTextBal"
        android:background="@drawable/back"
        android:id="@+id/textViewBalanceBox"
```

```
            android:layout_gravity="right|bottom" />

    </FrameLayout>

</FrameLayout>
```

8.7 Κωδικας Για StatisticsFragment Κλαση

8.7.1 Κώδικας Java

```
package com.example.dimitris.moneyger;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.res.TypedArray;
import android.net.Uri;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import java.text.SimpleDateFormat;
import java.util.Calendar;


/**
 * A simple {@link Fragment} subclass.
 * Activities that contain this fragment must implement the
 * {@link StatisticsFragment.OnFragmentInteractionListener} interface
 * to handle interaction events.
 * Use the {@link StatisticsFragment#newInstance} factory method to
 * create an instance of this fragment.
 */
public class StatisticsFragment extends Fragment {
    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;

    private OnFragmentInteractionListener mListener;

    //Dialog builders for the popup windows.
    AlertDialog.Builder dialogBuilder;

    //Integer for the dbHandler constructor.
```

```
private int settings = 2;

//Constructor for the class that handles the DB.
private dbHandler myDBHandler;


//Button in the statistics fragment.
private Button clear_button;


//Strings for start and end date.
private String start_date_str,end_date_str;


//TextView variable for the load "screen" in the statistics fragment.
private TextView load_area;


//String to display the transactions from each month.
private String entry;


//Integer to get the CURRENT year.
private int year;


//Integer to check if any transactions took place between the two dates.
private int existence;


/**
 * Use this factory method to create a new instance of
 * this fragment using the provided parameters.
 *
 * @return A new instance of fragment StatisticsFragment.
 */
// TODO: Rename and change types and number of parameters
public static StatisticsFragment newInstance(int position) {
    StatisticsFragment fragment = new StatisticsFragment();
    Bundle args = new Bundle();
    args.putString(ARG_PARAM1, String.valueOf(position));
    //args.putString(ARG_PARAM2, param2);
    fragment.setArguments(args);
    return fragment;
}

public StatisticsFragment() {
    // Required empty public constructor
}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    myDBHandler  = new dbHandler(getActivity(),null,null,settings);
    if (getArguments() != null) {
        mParam1 = getArguments().getString(ARG_PARAM1);
        mParam2 = getArguments().getString(ARG_PARAM2);
    }
}

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

```
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_statistics, container, false);
    }

    //Main method to initialize and utilize all the buttons and fields of the View
after it's
    //created using the onCreateView method.
    @Override
    public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        setupFields();
    }



//##########################################################################################
##########

    /**
     * When the fragment's view is created, the stats will be displayed. The stats
come from the db
     * and show EVERYTHING the user has done from the first day of the CURRENT year up
to the
     * current date. BUG FIXED: Now the year's value is automatically changed when we
have a new
     * year. The day and month values always stay the same.
     */

    private void setupFields(){

        clear_button = (Button) getView().findViewById(R.id.buttonMonthOk);

        load_area = (TextView) getView().findViewById(R.id.loadTextStatistics);

        showStatistics();
    }


//##########################################################################################
##########


//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$


//##########################################################################################
##########
    public void showStatistics(){
        Calendar c1 = Calendar.getInstance();
        Calendar c2 = Calendar.getInstance();

        //Setting the START date of CURRENT year.
        year = c1.get(Calendar.YEAR);
        c1.set(Calendar.YEAR,year);
        c1.set(Calendar.MONTH,1);
        c1.set(Calendar.DAY_OF_YEAR,1);

        SimpleDateFormat df = new SimpleDateFormat("yyyy/MM/dd");

        String date1 = df.format(c1.getTime());
        String date2 = df.format(c2.getTime());
```

```
        //Checking if there are any transactions up to NOW.
        existence = myDBHandler.yearlyTransactions(date1,date2);


        //If there are transactions in the database they are loaded.
        if (existence > 0){

Toast.makeText(getActivity(),R.string.transactions_found_success,Toast.LENGTH_SHORT)
                    .show();
            entry = myDBHandler.showTransactionsYearly(date1,date2);
            load_area.setText(entry);
        }
        else{

Toast.makeText(getActivity(),R.string.transactions_not_exist,Toast.LENGTH_SHORT)
                    .show();
        }

        //This button simply clears the screen from the statistics.
        clear_button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                load_area.setText("");
            }
        });
    }


//##########################################################################################
##########


//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$


    // TODO: Rename method, update argument and hook method into UI event
    public void onButtonPressed(Uri uri) {
        if (mListener != null) {
            mListener.onFragmentInteraction(uri);
        }
    }

    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
        try {
            mListener = (OnFragmentInteractionListener) activity;
        } catch (ClassCastException e) {
            throw new ClassCastException(activity.toString()
                    + " must implement OnFragmentInteractionListener");
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        mListener = null;
    }

    /**
     * This interface must be implemented by activities that contain this
     * fragment to allow an interaction in this fragment to be communicated
```

```
     * to the activity and potentially other fragments contained in that
     * activity.
     * <p/>
     * See the Android Training lesson <a href=
     * "http://developer.android.com/training/basics/fragments/communicating.html"
     * >Communicating with Other Fragments</a> for more information.
     */
    public interface OnFragmentInteractionListener {
        // TODO: Update argument type and name
        public void onFragmentInteraction(Uri uri);
    }

}
```

8.7.2 Κώδικας XML

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.dimitris.moneyger.StatisticsFragment">


    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:maxLines="20"
        android:scrollbars="vertical"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:id="@+id/loadTextStatistics"
        android:layout_gravity="center_horizontal|top" />


    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/clearButton"
        android:id="@+id/buttonMonthOk"
        android:layout_gravity="center_horizontal|bottom" />


</FrameLayout>
```

8.8 ΚΩΔΙΚΑΣ ΓΙΑ SETTINGSFRAGMENT ΚΛΑΣΗ

8.8.1 Κώδικας Java

```
package com.example.dimitris.moneyger;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.DatePickerDialog;
import android.content.DialogInterface;
```

```
import android.net.Uri;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.text.InputType;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.TextView;
import android.widget.Toast;

import org.w3c.dom.Text;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Locale;


/**
 * A simple {@link Fragment} subclass.
 * Activities that contain this fragment must implement the
 * {@link SettingsFragment.OnFragmentInteractionListener} interface
 * to handle interaction events.
 * Use the {@link SettingsFragment#newInstance} factory method to
 * create an instance of this fragment.
 */
public class SettingsFragment extends Fragment {

    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;


    private OnFragmentInteractionListener mListener;


    //Constructor for the class that handles the DB.
    private dbHandler myDBHandler;


    //Integer for the dbHandler constructor.
    private int settings = 2;


    //Dialog builders for the popup windows.
    AlertDialog.Builder dialogBuilder;


    //String to "hold" the salary entry from the user.
    public double salary_entry = 0.0;


    //Reference to the dialog box for the salary entered.
```

```
    private EditText user_salary;


    //Round Decimal class reference.
    private RoundDecimals twoDecimals;


    //Total Income with salary increase.
    private MonthlyIncome monthlySalary;


    //Getting the calendar instance to use it in the datePicker.
    Calendar myCalendar = Calendar.getInstance();


    //Text fields in find transaction alert box
    private EditText dateSTedit,dateENedit;
    private TextView dateStart,dateEnd;


    //Text fields in password alert box;
    private EditText password_set;


    //Locale variable to host the new view for the prompts,password
xml,confirm_password.xml.
    private View promptsView;
    private View passwordView;
    private View confirmView;
    private View confirmDropView;


    //String variables to store the dates--start and end-- from the find transaction
dialog box.
    private String start_date_str,end_date_str;


    //Default transaction id for the delete certain dialog.
    private int id_transaction = 0;


    //Date checkers for the integer part.
    private int start_day,start_month,start_year;
    private int end_day,end_month,end_year;


    //Date checkers for the string part.
    private String start_day_str,start_month_str,start_year_str;
    private String end_day_str,end_month_str,end_year_str;


    //String to return the transactions that belong between two dates.
    private String check;


    //Integer used for determining if a password is already set.
    private int existence;


    //Integer used for determining if a transaction exists between two given dates.
    private int transaction_exists;
```

```
    //String to get the old password and integer to check for equality with the one in
the database.
    private String old_password_entered,new_password_entered;
    private int password_equality;


    //EditTexts for the confirm_password.xml.
    private EditText edit_old,edit_new;


    //EditTexts for the confirm_drop_db.xml.
    private EditText confirm_pass;


    //String to get the password for dropping the database confirmation dialog box.
    private String drop_password;


    //Integer to show if we are talking about the same month (BUG FIX)
    private int same_month = -1;

    //Counter for save clicks.
    private int counter = 0;



//##########################################################################################
##########
    // TODO: Rename and change types and number of parameters
    public static SettingsFragment newInstance(int position) {
        SettingsFragment fragment = new SettingsFragment();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, String.valueOf(position));
        //args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    public SettingsFragment() {
        // Required empty public constructor
    }

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

        //An integer that equals to 2 is used in order to distinguish it
        //from the income and expenses entries.
        myDBHandler     = new dbHandler(getActivity(),null,null,settings);
        twoDecimals     = new RoundDecimals();
        monthlySalary   = new MonthlyIncome();
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

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

        // Inflate the layout for this fragment
        View v =
```

```
LayoutInflater.from(getActivity()).inflate(R.layout.fragment_settings, null);

        return v;
    }


    //Main method to initialize and utilize all the buttons and fields of the View
after it's
    //created using the onCreateView method.
    @Override
    public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

Toast.makeText(getActivity(),R.string.press_on_text,Toast.LENGTH_SHORT).show();
        setupFields();
    }

//##############################################################################################
##########


//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$


//##############################################################################################
##########

//##############################################################################################
##########
    private void setupFields(){

        //Setting the default layout for the dialog window through prompt.xml.
        LayoutInflater inflater = LayoutInflater.from(getActivity());
        promptsView = inflater.inflate(R.layout.prompts,null);


        //Setting the default layout for the dialog window through password.xml.
        LayoutInflater inflaterpass = LayoutInflater.from(getActivity());
        passwordView = inflaterpass.inflate(R.layout.password,null);


        //Setting the default layout for the dialog window through
confirm_password.xml.
        LayoutInflater inflaterconfirm = LayoutInflater.from(getActivity());
        confirmView = inflaterconfirm.inflate(R.layout.confirm_password,null);


        //Setting the default layout for dialog window through confirm_drop_dp.xml.
        LayoutInflater inflaterdropdb = LayoutInflater.from(getActivity());
        confirmDropView = inflaterdropdb.inflate(R.layout.confirm_drop_db,null);


        //Creating the references for the edit text views located in the promptsView.
        dateSTedit = (EditText) promptsView.findViewById(R.id.editTextStartDate);
        dateENedit = (EditText) promptsView.findViewById(R.id.editTextEndDate);


        //Creating the references for the edit text views int the passwordView.
        password_set = (EditText) passwordView.findViewById(R.id.editTextPassword);


        //EditText references for the confirm_password.xml.
```

```
        edit_old = (EditText) confirmView.findViewById(R.id.editTextOld);
        edit_new = (EditText) confirmView.findViewById(R.id.editTextNew);


        //Creating the references for the text views in the prompt xml file.
        dateStart = (TextView) promptsView.findViewById(R.id.textViewStartDate);
        dateEnd   = (TextView) promptsView.findViewById(R.id.textViewEndDate);


        //Creating the references for the edit text in the confirm_drop_db.xml.
        confirm_pass = (EditText)
confirmDropView.findViewById(R.id.editTextPasswordGet);


        //Creating the references for the xml file.
        ImageButton buttonMonth = (ImageButton) getView().
                findViewById(R.id.imageButtonMonth);
        ImageButton buttonclearDB = (ImageButton) getView().
                findViewById(R.id.imageButtonClearDB);
        ImageButton buttonclearTr = (ImageButton) getView().
                findViewById(R.id.imageButtonClearTrans);
        ImageButton buttonPass = (ImageButton) getView().
                findViewById(R.id.imageButtonPass);
        ImageButton buttonFind = (ImageButton) getView().
                findViewById(R.id.imageButtonFindTrans);



        //Adding animations.
        buttonMonth.startAnimation(AnimationUtils.loadAnimation(getActivity(),
                android.R.anim.slide_in_left));
        buttonclearDB.startAnimation(AnimationUtils.loadAnimation(getActivity(),
                android.R.anim.slide_in_left));
        buttonclearTr.startAnimation(AnimationUtils.loadAnimation(getActivity(),
                android.R.anim.slide_in_left));
        buttonPass.startAnimation(AnimationUtils.loadAnimation(getActivity(),
                android.R.anim.slide_in_left));
        buttonFind.startAnimation(AnimationUtils.loadAnimation(getActivity(),
                android.R.anim.slide_in_left));


        /**
         * Adding the actions performed by the Imagebuttons.
         */

        //Onclick actions performed by the clearDB button.
        buttonclearDB.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                clearDBDialog();
            }
        });


        //Onclick actions performed by the Month button.
        buttonMonth.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                monthDialog();
            }
        });
```

```
        //Onclick actions performed by the deleteCertain transaction button.
        buttonclearTr.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                delCertainDialog();
            }
        });


        //Onclick actions performed by the Find transaction button.
        buttonFind.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                findTransDialog();
            }
        });


        //Onclick actions performed by the Password button.
        buttonPass.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                passwordDialog();
            }
        });

    }

//##########################################################################################
##########

    /**
     * Implementing the methods used in the setup method above.
     */



//##########################################################################################
##########
    //DROP DB BUTTON
    //Dialog window to make sure if one is to drop the DB tables.
    private void clearDBDialog(){
        dialogBuilder = new AlertDialog.Builder(getActivity());
        dialogBuilder.setView(confirmDropView);
        dialogBuilder.setCancelable(false);

        dialogBuilder.setPositiveButton(R.string.proceed, new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                drop_password = confirm_pass.getText().toString();

                password_equality = myDBHandler.passwordChecking(drop_password);
                if (password_equality > 0) {
                    deleteConfirmedDialog();
                } else {
                    Toast.makeText(getActivity(), R.string.password_inequality,
Toast.LENGTH_SHORT)
                            .show();
                }
                confirm_pass.setText("");
                dialog.cancel();

                ((ViewGroup) confirmDropView.getParent()).removeView(confirmDropView);
```

```
            }
        });
        dialogBuilder.setNegativeButton(R.string.cancel, new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                confirm_pass.setText("");
                dialog.cancel();

                ((ViewGroup) confirmDropView.getParent()).removeView(confirmDropView);
            }
        });

        AlertDialog alert = dialogBuilder.create();
        alert.show();
    }

    //Drop all tables after confirmation.
    private void deleteConfirmedDialog(){
        dialogBuilder = new AlertDialog.Builder(getActivity());
        dialogBuilder.setTitle(R.string.eraseTitle);
        dialogBuilder.setMessage(R.string.eraseConfirm);
        dialogBuilder.setCancelable(false);
        dialogBuilder.setPositiveButton(R.string.yes, new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                myDBHandler.clearDB();
            }
        });
        dialogBuilder.setNegativeButton(R.string.no, new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                dialog.cancel();
            }
        });

        AlertDialog alert = dialogBuilder.create();
        alert.show();
    }

//##########################################################################################
##########


//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$


//##########################################################################################
##########
    //SALARY BUTTON
    //Dialog window for the monthly income(salary).
    public void monthDialog(){

        dialogBuilder = new AlertDialog.Builder(getActivity());
        dialogBuilder.setTitle(R.string.salaryEntryTitle);
        dialogBuilder.setMessage(R.string.salary_enter);
        dialogBuilder.setCancelable(false);

        final EditText input  =  new EditText(getView().getContext());

        input.setInputType(InputType.TYPE_CLASS_PHONE);
        dialogBuilder.setView(input);
```

```
        dialogBuilder.setPositiveButton(R.string.ok, new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                String salarySTR = input.getEditableText().toString();

                if  (salarySTR.contains(",")){
                    salarySTR = salarySTR.replace(",",".");
                }

                if (amountValidation(salarySTR) == 1){
                    salary_entry = Double.parseDouble(salarySTR);
                    salary_entry = twoDecimals.roundTwoDecimals(salary_entry);
                }

                insertMonthly(salary_entry);

            }
        });
        dialogBuilder.setNegativeButton(R.string.cancel, new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.cancel();
            }
        });

        AlertDialog alert = dialogBuilder.create();
        alert.show();
    }


    //Function to automatically insert the given amount in the db.
    public void insertMonthly(double salary){
        Calendar calendar = Calendar.getInstance();
        SimpleDateFormat df = new SimpleDateFormat("M");
        String formattedDate = df.format(calendar.getTime());
        myDBHandler.saveSalary(salary, formattedDate);
    }

//#########################################################################################
##########


//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$


//#########################################################################################
##########
    //DELETE CERTAIN BUTTON
    //Dialog window for certain transaction delete.
    public void delCertainDialog(){

        dialogBuilder = new AlertDialog.Builder(getActivity());
        dialogBuilder.setView(confirmDropView);
        dialogBuilder.setCancelable(false);

        dialogBuilder.setPositiveButton(R.string.proceed, new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                drop_password = confirm_pass.getText().toString();

                password_equality = myDBHandler.passwordChecking(drop_password);
```

```
                if (password_equality > 0) {
                    deleteCertainConfDialog();
                } else {
                    Toast.makeText(getActivity(), R.string.password_inequality,
Toast.LENGTH_SHORT)
                            .show();
                }
                confirm_pass.setText("");
                dialog.cancel();

                ((ViewGroup) confirmDropView.getParent()).removeView(confirmDropView);
            }
        });
        dialogBuilder.setNegativeButton(R.string.cancel, new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                confirm_pass.setText("");
                dialog.cancel();

                ((ViewGroup) confirmDropView.getParent()).removeView(confirmDropView);
            }
        });

        AlertDialog alert = dialogBuilder.create();
        alert.show();
    }

    //Delete certain transaction-if exists- after confirmation.
    public void deleteCertainConfDialog(){
        dialogBuilder = new AlertDialog.Builder(getActivity());
        dialogBuilder.setTitle(R.string.delete_transaction);
        dialogBuilder.setMessage(R.string.transaction_id);
        dialogBuilder.setCancelable(false);


        final EditText inputid = new EditText(getView().getContext());

        inputid.setInputType(InputType.TYPE_CLASS_PHONE);
        dialogBuilder.setView(inputid);


        dialogBuilder.setPositiveButton(R.string.ok , new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {

                /*
                 * BUG FIXED when pressing OK without any values entered.
                 */
                if (inputid.getText().toString().equals("")){
                    Toast.makeText(getActivity(),R.string.fields_required,
Toast.LENGTH_SHORT)
                            .show();
                }
                else {
                    id_transaction =
Integer.parseInt(inputid.getEditableText().toString());
                }


                if (myDBHandler.findTransaction(id_transaction) == 1) {
                    transactionDialog(id_transaction);
                } else {
                    Toast.makeText(getActivity(),R.string.transaction_error,
```

```
Toast.LENGTH_SHORT)
                                .show();
                    }

                }
            });

            dialogBuilder.setNegativeButton(R.string.cancel, new
DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    dialog.cancel();
                }
            });

            AlertDialog alert = dialogBuilder.create();
            alert.show();
        }


        //Dialog window for transaction status display.
        public void transactionDialog(final int id) {
            dialogBuilder = new AlertDialog.Builder(getActivity());
            dialogBuilder.setTitle(R.string.transaction_details);
            dialogBuilder.setMessage(R.string.warning_delete_transaction);
            dialogBuilder.setCancelable(false);


            String details;
            details = myDBHandler.transactionToString(id);

            final EditText transaction_details = new EditText(getView().getContext());

            transaction_details.setText(details);
            transaction_details.setKeyListener(null);
            dialogBuilder.setView(transaction_details);

            dialogBuilder.setPositiveButton(R.string.delete, new
DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    myDBHandler.removeRow(id);
                }
            });

            dialogBuilder.setNegativeButton(R.string.cancel, new
DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    dialog.cancel();
                }

            });

            AlertDialog alert = dialogBuilder.create();
            alert.show();

        }

//#########################################################################################
##########
```

```
//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$


//######################################################################################
##########
    //SEARCH TRANSACTION BUTTON
    //Dialog window for searching transactions,a dialog window for the date picker and
a dialog
    //window to display the results.
    public void findTransDialog() {

        dialogBuilder = new AlertDialog.Builder((getActivity()));
        dialogBuilder.setView(promptsView);
        dialogBuilder.setCancelable(false);

        //Setting the onclick actions for the text views.
        dateStart.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                new DatePickerDialog(getActivity(), dateST, myCalendar
                        .get(Calendar.YEAR), myCalendar.get(Calendar.MONTH),
                        myCalendar.get(Calendar.DAY_OF_MONTH)).show();
            }
        });

        dateEnd.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                new DatePickerDialog(getActivity(), dateEN, myCalendar
                        .get(Calendar.YEAR), myCalendar.get(Calendar.MONTH),
                        myCalendar.get(Calendar.DAY_OF_MONTH)).show();
            }
        });


        /**
         * MAJOR BUG FIXED: When the dialog box is destroyed either by cancel or by
search
         * it's parent still has this child as it is loaded in the onCreated method,so
if
         * --remaining in the settings fragment(parent)-- one pressed the Search icon
for
         * the second time that would lead the app to stop working. Thus when the
dialog is
         * closed the promptsView that it uses is removed FROM THE PARENT.
         */

        dialogBuilder.setPositiveButton(R.string.search, new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {

                if (dateSTedit.getText().toString().equals("") ||
dateENedit.getText().
                        toString().equals("")){
                    Toast.makeText(getActivity(),R.string.fields_required,
Toast.LENGTH_SHORT)
                            .show();
                    dateSTedit.setText("");
                    dateENedit.setText("");
                }
```

```
                /**
                 * Need the check if the start date picked is greater than the ending
one. If not it
                 * may cause serious trouble when searching in the database causing
the application
                 * to stop.Splitting the string from the editText threeways to get a
value for the
                 * day,the month and the year.
                 */

                else{
                    start_date_str = dateSTedit.getText().toString();
                    end_date_str   = dateENedit.getText().toString();

                    //Year.
                    start_year = Integer.parseInt(start_date_str.substring(0,4));
                    end_year   = Integer.parseInt(end_date_str.substring(0,4));


                    //Month.
                    start_month = Integer.parseInt(start_date_str.substring(5,7));
                    end_month   = Integer.parseInt(end_date_str.substring(5,7));


                    //Day.
                    start_day = Integer.parseInt(start_date_str.substring(8,10));
                    end_day   = Integer.parseInt(end_date_str.substring(8,10));


                    if (start_year > end_year){
                        Toast.makeText(getActivity(),R.string.year_error
                                ,Toast.LENGTH_SHORT).show();
                    }
                    else{
                        if (start_month == end_month){
                            //same_month = 1;
                        }
                        else{
                            if (start_month > end_month){
                                Toast.makeText(getActivity(),R.string.month_error
                                        ,Toast.LENGTH_SHORT).show();
                            }
                        }
                    }



//*************************************************************************
                    //      Method to check the database for transactions between the
two dates.

                    /**MAJOR BUG FIXED:
                      *If the user used e.g 1/1/2015 as a start date and 30/1/2015 as
                      *an end date then SQL call would display all results from the
database,inlcu-
                      *ding results that relate to OTHER MONTHS. This is because of
the SQLite itse-
                      *lf which supports every date format BUT works properly only
with the
                      *yyyy/mm/dd one.When i changed the format WRITTEN in the
database(in Income-
                      * Fragment and in ExpensesFragment) everything worked as
supposed.
```

```
                     */

                    transaction_exists =
myDBHandler.transactionsExistBetweenDates(dateSTedit.
                            getText().toString(),dateENedit.getText().toString());

                    if (transaction_exists > 0 ){
                        check =
myDBHandler.searchTransaction(dateSTedit.getText().toString(),
                                dateENedit.getText().toString());
                        transBetweenDatesDialog(check);
                    }
                    else{
                        Toast.makeText(getActivity(),R.string.trans_existence,
                                Toast.LENGTH_SHORT).show();
                    }


//*****************************************************************************


                    /**
                     * After the dialog is closed either by cancel or by search
click,all the fields
                     * it contains must become empty again in order to begin a new
search if a user
                     * wants to.
                     */
                    dateSTedit.setText("");
                    dateENedit.setText("");
                }

                ((ViewGroup) promptsView.getParent()).removeView(promptsView);
            }
        });
        dialogBuilder.setNegativeButton(R.string.cancel, new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dateSTedit.setText("");
                dateENedit.setText("");
                dialog.cancel();
                ((ViewGroup)promptsView.getParent()).removeView(promptsView);

            }
        });

        AlertDialog alert = dialogBuilder.create();
        alert.show();
    }


    //Creating the datePicker popup window for the starting date.
    final DatePickerDialog.OnDateSetListener dateST = new
DatePickerDialog.OnDateSetListener() {
        @Override
        public void onDateSet(DatePicker view, int year, int monthOfYear,
                              int dayOfMonth) {
            // TODO Auto-generated method stub
            myCalendar.set(Calendar.YEAR, year);
            myCalendar.set(Calendar.MONTH, monthOfYear);
            myCalendar.set(Calendar.DAY_OF_MONTH, dayOfMonth);
            updateLabelST();
```

```
        }
    };


    //Creating the datePicker popup window for the ending date.
    final DatePickerDialog.OnDateSetListener dateEN = new
DatePickerDialog.OnDateSetListener() {
        @Override
        public void onDateSet(DatePicker view, int year, int monthOfYear,
                              int dayOfMonth) {
            // TODO Auto-generated method stub
            myCalendar.set(Calendar.YEAR, year);
            myCalendar.set(Calendar.MONTH, monthOfYear);
            myCalendar.set(Calendar.DAY_OF_MONTH, dayOfMonth);
            updateLabelEN();

        }
    };


    //Changing the text of the appropriate TextView.
    private void updateLabelST() {

        //Creating the references for the edit texts in the prompt xml file.
        dateSTedit = (EditText) promptsView.findViewById(R.id.editTextStartDate);

        String myFormat = "yyyy/MM/dd";
        SimpleDateFormat sdf = new SimpleDateFormat(myFormat, Locale.US);
        dateSTedit.setText(sdf.format(myCalendar.getTime()));
    }


    //Changing the text of the appropriate TextView.
    private void updateLabelEN() {
        //Creating the references for the edit texts in the prompt xml file.
        dateENedit = (EditText) promptsView.findViewById(R.id.editTextEndDate);

        String myFormat = "yyyy/MM/dd";
        SimpleDateFormat sdf = new SimpleDateFormat(myFormat, Locale.US);
        dateENedit.setText(sdf.format(myCalendar.getTime()));
    }




    //Dialog box to display the transactions between two given dates.
    public void transBetweenDatesDialog(String result){
        dialogBuilder = new AlertDialog.Builder(getActivity());
        dialogBuilder.setTitle(R.string.trans_between_datesTITLE);
        dialogBuilder.setMessage(R.string.trans_between_datesMSG);
        dialogBuilder.setCancelable(false);


        final EditText transaction_result = new EditText(getView().getContext());

        transaction_result.setText(result);
        transaction_result.setKeyListener(null);

        dialogBuilder.setView(transaction_result);

        dialogBuilder.setPositiveButton(R.string.ok, new
DialogInterface.OnClickListener() {
```

```
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.cancel();
            }
        });


        dialogBuilder.setNegativeButton(R.string.cancel, new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.cancel();
            }

        });

        AlertDialog alert = dialogBuilder.create();
        alert.show();

    }

//#############################################################################################
##########


//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$


//#############################################################################################
##########
    //PASSWORD BUTTON
    //Dialog box for the Password button,dialog box for setting new password
,confirmation dialog
    //box for the old password validation.
    public void passwordDialog(){
        dialogBuilder = new AlertDialog.Builder(getActivity());
        dialogBuilder.setView(passwordView);
        dialogBuilder.setCancelable(false);

        //Checking if a password is already stored in the database. This button has
two uses,first
        //if the password is not already stored in the db it creates a new one and
stores it in the
        //db.Secondly,if the password exists then it is used to update the password to
a new one.


        dialogBuilder.setPositiveButton(R.string.new_pass, new
DialogInterface.OnClickListener(){
            @Override
            public void onClick(DialogInterface dialog, int which) {

                existence = myDBHandler.passwordExistToDb();
                if (existence == 0) {

                    if (password_set.getText().toString().equals("")){

Toast.makeText(getActivity(),R.string.fields_required,Toast.LENGTH_SHORT)
                                .show();
                    }
                    else {
```

```
myDBHandler.addPasswordToDb(password_set.getText().toString());
                        Toast.makeText(getActivity(), R.string.password_add,
Toast.LENGTH_SHORT)
                                .show();
                    }
                }
                else{

Toast.makeText(getActivity(),R.string.password_exists,Toast.LENGTH_SHORT)
                            .show();
                    checkOldPasswordDialog();
                }
                password_set.setText("");
                dialog.cancel();
                ((ViewGroup) passwordView.getParent()).removeView(passwordView);
            }
        });

        dialogBuilder.setNegativeButton(R.string.cancel, new
DialogInterface.OnClickListener(){
            @Override
            public void onClick(DialogInterface dialog, int which) {
                password_set.setText("");
                dialog.cancel();
                ((ViewGroup) passwordView.getParent()).removeView(passwordView);
            }
        });


        AlertDialog alert = dialogBuilder.create();
        alert.show();

    }


    //Creating dialog box in order to check if the user is a legit one. This is done
by checking if
    //the old password that he will enter in this dialog box is equal to the one
stored in the data-
    //base. If they are equal then we proceed to password update.
    public void checkOldPasswordDialog() {

        dialogBuilder = new AlertDialog.Builder(getActivity());
        dialogBuilder.setView(confirmView);
        dialogBuilder.setCancelable(false);

        dialogBuilder.setPositiveButton(R.string.update, new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {

                old_password_entered = edit_old.getText().toString();
                new_password_entered = edit_new.getText().toString();
                password_equality =
myDBHandler.passwordChecking(old_password_entered);

                if (password_equality > 0) {
                    myDBHandler.updatePasswordToDb(new_password_entered);
                    Toast.makeText(getActivity(), R.string.update_success,
Toast.LENGTH_SHORT).show();
                } else {
                    Toast.makeText(getActivity(), R.string.password_inequality,
Toast.LENGTH_SHORT)
```

```
                        .show();
                }

                edit_old.setText("");
                edit_new.setText("");

                dialog.cancel();
                ((ViewGroup) confirmView.getParent()).removeView(confirmView);

            }

        });

        dialogBuilder.setNegativeButton(R.string.cancel, new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                edit_old.setText("");
                edit_new.setText("");
                dialog.cancel();
                ((ViewGroup) confirmView.getParent()).removeView(confirmView);
            }
        });


        AlertDialog alert = dialogBuilder.create();
        alert.show();
    }



//#########################################################################################
##########


//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$


//#########################################################################################
##########
    //Checking if the input amount is valid.
    private int amountValidation(String str){
        for (counter=0; counter<str.length();counter++){
            if (!Character.isDigit(str.charAt(counter))){
                if (str.charAt(counter) == '.' || str.charAt(counter) == ','){
                    continue;
                }
                else {
                    Toast.makeText(getActivity(),R.string.invalid_input,
Toast.LENGTH_SHORT).show();
                    return 0;
                }
            }
        }

        return 1;
    }

//#########################################################################################
##########
```

```
    // TODO: Rename method, update argument and hook method into UI event
    public void onButtonPressed(Uri uri) {
        if (mListener != null) {
            mListener.onFragmentInteraction(uri);
        }
    }

    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
        try {
            mListener = (OnFragmentInteractionListener) activity;
        } catch (ClassCastException e) {
            throw new ClassCastException(activity.toString()
                    + " must implement OnFragmentInteractionListener");
        }
    }


    @Override
    public void onDetach() {
        super.onDetach();
        mListener = null;
    }


    /**
     * This interface must be implemented by activities that contain this
     * fragment to allow an interaction in this fragment to be communicated
     * to the activity and potentially other fragments contained in that
     * activity.
     * <p/>
     * See the Android Training lesson <a href=
     * "http://developer.android.com/training/basics/fragments/communicating.html"
     * >Communicating with Other Fragments</a> for more information.
     */
    public interface OnFragmentInteractionListener {
        // TODO: Update argument type and name
        public void onFragmentInteraction(Uri uri);
    }
}
```

8.8.2 Κώδικας XML

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.dimitris.moneyger.SettingsFragment">

    <!-- TODO: Update blank fragment layout -->

    <TextView android:layout_width="match_parent" android:layout_height="match_parent"
        android:layout_gravity="left|center_vertical" />

    <FrameLayout
        android:layout_width="138dp"
        android:layout_height="143dp"
        android:layout_gravity="left|top">
```

```
    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageButtonMonth"
        android:layout_gravity="center"
        android:background="@drawable/ic_action_monthly"
        android:contentDescription="@string/monthlyTransactions"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="@string/monthlyTransactions"
        android:id="@+id/textView2"
        android:layout_gravity="center_horizontal|bottom" />
</FrameLayout>

<FrameLayout
    android:layout_width="138dp"
    android:layout_height="143dp"
    android:layout_gravity="left|center_vertical" >

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageButtonClearDB"
        android:layout_gravity="center"
        android:background="@drawable/ic_action_deletedb"
        android:contentDescription="@string/clearDB"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="@string/clearDB"
        android:id="@+id/textView4"
        android:layout_gravity="center_horizontal|bottom" />
</FrameLayout>

<FrameLayout
    android:layout_width="138dp"
    android:layout_height="143dp"
    android:layout_gravity="right|top" >

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageButtonClearTrans"
        android:layout_gravity="center"
        android:background="@drawable/ic_content_cleartransaction"
        android:contentDescription="@string/clearTransactions"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="@string/clearTransactions"
        android:id="@+id/textView3"
        android:layout_gravity="center_horizontal|bottom" />
</FrameLayout>

<FrameLayout
    android:layout_width="138dp"
```

```
        android:layout_height="143dp"
        android:layout_gravity="right|center_vertical" >

        <ImageButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/imageButtonPass"
            android:layout_gravity="center"
            android:background="@drawable/ic_action_password"
            android:contentDescription="@string/password"/>

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="@string/password"
            android:id="@+id/textView5"
            android:layout_gravity="center_horizontal|bottom" />
    </FrameLayout>


    <FrameLayout
        android:layout_width="138dp"
        android:layout_height="143dp"
        android:layout_gravity="center_horizontal|bottom" >

        <ImageButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/imageButtonFindTrans"
            android:layout_gravity="center"
            android:background="@drawable/ic_action_find"
            android:contentDescription="@string/findTransaction"/>

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="@string/findTransaction"
            android:id="@+id/textView7"
            android:layout_gravity="center_horizontal|bottom" />
    </FrameLayout>


</FrameLayout>
```

8.9 Κωδικας Για AboutFragment Κλαση

8.9.1 Κώδικας Java

```
package com.example.dimitris.moneyger;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
```

```
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.View;
import android.view.ViewGroup;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.Toast;


/**
 * A simple {@link Fragment} subclass.
 * Activities that contain this fragment must implement the
 * {@link AboutFragment.OnFragmentInteractionListener} interface
 * to handle interaction events.
 * Use the {@link AboutFragment#newInstance} factory method to
 * create an instance of this fragment.
 */
public class AboutFragment extends Fragment {
    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";


    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;

    private OnFragmentInteractionListener mListener;


    //Buttons and images local variables.
    private ImageView   imgUth;
    private ImageButton facebook,twitter,gplus,gmail,help;
    private ImageButton author;


    //Dialog builders for the popup windows.
    AlertDialog.Builder dialogBuilder;

    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
     *
     * @param param1 Parameter 1.
     * @param param2 Parameter 2.
     * @return A new instance of fragment AboutFragment.
     */
    // TODO: Rename and change types and number of parameters
    public static AboutFragment newInstance(int position) {
        AboutFragment fragment = new AboutFragment();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, String.valueOf(position));
        //args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
```

```
        return fragment;
    }

    public AboutFragment() {
        // Required empty public constructor
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_about, container, false);
    }

    //Main method to initialize and utilize all the buttons and fields of the View
after it's
    //created using the onCreateView method.
    @Override
    public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

Toast.makeText(getActivity(),R.string.press_on_help,Toast.LENGTH_SHORT).show();
        setupFields();
    }

    public void setupFields(){
        imgUth = (ImageView) getView().findViewById(R.id.imageViewUth);

        facebook    =   (ImageButton)
getView().findViewById(R.id.imageButtonFacebook);
        twitter     =   (ImageButton)
getView().findViewById(R.id.imageButtonTwitter);
        gplus       =   (ImageButton)
getView().findViewById(R.id.imageButtonGPlus);
        gmail       =   (ImageButton)
getView().findViewById(R.id.imageButtonGMail);
        help        =   (ImageButton)    getView().findViewById(R.id.imageButtonHelp);
        author      =   (ImageButton)    getView().findViewById(R.id.imageButtonMe);



        //Adding animations.

        facebook.startAnimation(AnimationUtils.loadAnimation(getActivity(),
                android.R.anim.slide_in_left));
        twitter.startAnimation(AnimationUtils.loadAnimation(getActivity(),
                android.R.anim.slide_in_left));
        gplus.startAnimation(AnimationUtils.loadAnimation(getActivity(),
                android.R.anim.slide_in_left));
        gmail.startAnimation(AnimationUtils.loadAnimation(getActivity(),
                android.R.anim.slide_in_left));
        author.startAnimation(AnimationUtils.loadAnimation(getActivity(),
                android.R.anim.slide_in_left));
        help.startAnimation(AnimationUtils.loadAnimation(getActivity(),
```

```
            android.R.anim.slide_in_left));

    //Setting the actions performed by clicking the buttons and logo.

    imgUth.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            uthPage();
        }
    });


    facebook.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            facebookPage();
        }
    });


    twitter.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            twitterPage();
        }
    });


    gmail.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            gmailPage();
        }
    });


    gplus.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            gplusPage();
        }
    });



    help.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            helpPage();
        }
    });


    author.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            auhtorPage();
        }
    });

}
```

```
    /**
     * Implementing the functions used when clicking the buttons and logo.
     */

    //Clicking on the Uth Logo opens the uth website on the user's browser.
    public void uthPage(){
        Intent intent = new Intent();
        intent.setAction(Intent.ACTION_VIEW);
        intent.addCategory(Intent.CATEGORY_BROWSABLE);
        intent.setData(Uri.parse("http://www.inf.uth.gr"));
        startActivity(intent);
    }

    //Opening the facebook page for the creator's profile.
    public void facebookPage(){
        Intent intent = new Intent();
        intent.setAction(Intent.ACTION_VIEW);
        intent.addCategory(Intent.CATEGORY_BROWSABLE);
        intent.setData(Uri.parse("http://www.facebook.com/dimitris.karagiannis.73"));
        startActivity(intent);
    }

    //Opening the twitter page for the creator's profile.
    public void twitterPage(){
        Intent intent = new Intent();
        intent.setAction(Intent.ACTION_VIEW);
        intent.addCategory(Intent.CATEGORY_BROWSABLE);
        intent.setData(Uri.parse("https://twitter.com/Mitsonio"));
        startActivity(intent);
    }

    //Sending email to the creator.
    public void gmailPage(){

        String[] TO = {"dkaragiannis91@gmail.com"};

        Intent emailIntent = new Intent(Intent.ACTION_SEND);

        emailIntent.setData(Uri.parse("mailto:"));
        emailIntent.setType("text/plain");
        emailIntent.putExtra(Intent.EXTRA_EMAIL, TO);

        try {
            startActivity(Intent.createChooser(emailIntent, "Send mail to creator"));
            getActivity().finish();
        }
        catch (android.content.ActivityNotFoundException ex) {
            Toast.makeText(getActivity(), "Email client is dead!",
                    Toast.LENGTH_SHORT).show();
        }
    }

    //Opening the google plus page for the creator's profile.
    public void gplusPage(){
        Intent intent = new Intent();
        intent.setAction(Intent.ACTION_VIEW);
        intent.addCategory(Intent.CATEGORY_BROWSABLE);

intent.setData(Uri.parse("https://plus.google.com/u/0/+DimitrisKaragiannis91"));
        startActivity(intent);
    }

    //Opening the help menu.
```

```
    public void helpPage(){
        dialogBuilder = new AlertDialog.Builder(getActivity());
        dialogBuilder.setTitle(R.string.help_title);
        dialogBuilder.setMessage(R.string.help_purpose);
        dialogBuilder.setCancelable(false);

        dialogBuilder.setPositiveButton(R.string.ok,new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                dialog.cancel();
            }
        });

        AlertDialog alert = dialogBuilder.create();
        alert.show();
    }

    //Message about the application,the author and his instructors.
    public void auhtorPage(){
        dialogBuilder = new AlertDialog.Builder(getActivity());
        dialogBuilder.setTitle(R.string.about_app);
        dialogBuilder.setMessage(R.string.app_message);
        dialogBuilder.setCancelable(false);
        dialogBuilder.setPositiveButton(R.string.ok, new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                dialog.cancel();

            }
        });

        AlertDialog alert = dialogBuilder.create();
        alert.show();
    }

    // TODO: Rename method, update argument and hook method into UI event
    public void onButtonPressed(Uri uri) {
        if (mListener != null) {
            mListener.onFragmentInteraction(uri);
        }
    }

    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
        try {
            mListener = (OnFragmentInteractionListener) activity;
        } catch (ClassCastException e) {
            throw new ClassCastException(activity.toString()
                    + " must implement OnFragmentInteractionListener");
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        mListener = null;
    }

    /**
     * This interface must be implemented by activities that contain this
     * fragment to allow an interaction in this fragment to be communicated
     * to the activity and potentially other fragments contained in that
```

```
     * activity.
     * <p/>
     * See the Android Training lesson <a href=
     * "http://developer.android.com/training/basics/fragments/communicating.html"
     * >Communicating with Other Fragments</a> for more information.
     */
    public interface OnFragmentInteractionListener {
        // TODO: Update argument type and name
        public void onFragmentInteraction(Uri uri);
    }

}
```

8.9.2 Κώδικας XML

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.dimitris.moneyger.AboutFragment">

    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        />


    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="center">

        <FrameLayout
            android:layout_width="match_parent"
            android:layout_height="125dp"
            android:layout_gravity="left|top">

            <ImageView
                android:layout_width="117dp"
                android:layout_height="117dp"
                android:id="@+id/imageViewUth"
                android:layout_gravity="center_horizontal|bottom"
                android:background="@drawable/ic_about_me" />

        </FrameLayout>

        <FrameLayout
            android:layout_width="match_parent"
            android:layout_height="275dp"
            android:layout_gravity="center">

            <FrameLayout
                android:layout_width="match_parent"
                android:layout_height="106dp"
                android:layout_gravity="left|top">

                <ImageButton
                    android:layout_width="60dp"
```

```
            android:layout_height="60dp"
            android:id="@+id/imageButtonFacebook"
            android:layout_gravity="left|bottom"
            android:background="@drawable/ic_action_facebook" />

        <ImageButton
            android:layout_width="60dp"
            android:layout_height="60dp"
            android:id="@+id/imageButtonGMail"
            android:layout_gravity="right|bottom"
            android:background="@drawable/ic_action_gmail" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:text="@string/facebook"
            android:id="@+id/textView12"
            android:layout_gravity="left|top" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:text="@string/gmail"
            android:id="@+id/textView13"
            android:layout_gravity="right|top" />

    </FrameLayout>

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="171dp"
        android:layout_gravity="left|bottom">

        <ImageButton
            android:layout_width="60dp"
            android:layout_height="60dp"
            android:id="@+id/imageButtonGPlus"
            android:layout_gravity="left|bottom"
            android:background="@drawable/ic_action_gplus" />

        <ImageButton
            android:layout_width="60dp"
            android:layout_height="60dp"
            android:id="@+id/imageButtonTwitter"
            android:layout_gravity="right|bottom"
            android:background="@drawable/ic_action_twitter" />

        <ImageButton
            android:layout_width="70dp"
            android:layout_height="70dp"
            android:id="@+id/imageButtonHelp"
            android:layout_gravity="center"
            android:background="@drawable/ic_action_help" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:text="@string/help"
            android:id="@+id/textView14"
            android:layout_gravity="center_horizontal|top" />
```

```
                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:textAppearance="?android:attr/textAppearanceSmall"
                    android:text="@string/gplus"
                    android:id="@+id/textView15"
                    android:layout_gravity="left|center_vertical" />

                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:textAppearance="?android:attr/textAppearanceSmall"
                    android:text="@string/twitter"
                    android:id="@+id/textView16"
                    android:layout_gravity="right|center_vertical" />
            </FrameLayout>
        </FrameLayout>

        <FrameLayout
            android:layout_width="match_parent"
            android:layout_height="111dp"
            android:layout_gravity="left|bottom">


            <ImageButton
                android:layout_width="70dp"
                android:layout_height="70dp"
                android:id="@+id/imageButtonMe"
                android:layout_gravity="center_horizontal|bottom"
                android:background="@drawable/ic_action_info" />

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textAppearance="?android:attr/textAppearanceSmall"
                android:text="@string/info"
                android:id="@+id/textView"
                android:layout_gravity="center_horizontal|top" />

        </FrameLayout>

    </FrameLayout>


</FrameLayout>
```

8.10 Κωδικας Για dbHandler Κλαση

```
package com.example.dimitris.moneyger;


import android.content.ContentValues;
import android.content.Context;
import android.content.res.Resources;
import android.database.Cursor;
```

```
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteException;
import android.database.sqlite.SQLiteOpenHelper;

/**
 * Main class that is used to handle the database in and outs. Everything
that is loaded or
 * written from/to the database is using a reference to the constructor of
this class.
 */

public class dbHandler extends SQLiteOpenHelper {

    private double totalIncDB = 0.0;
    private double totalExpDB = 0.0;
    private double totalMonth = 0.0;
    private double totalSal   = 0.0;

    private float graph_data  = 0;

    private int month_current = 0;

    private static final  int     DATABASE_VERSION    = 1;
    public static  final  String  DATABASE_NAME       = "AREA56.db";


    public static   final  String TABLE_TRANSACTIONS  = "tempTable";
    public static   final  String TABLE_TOTALSAL      = "salaryTable";
    public static   final  String TABLE_SAVESALARY    = "saveSalTable";
    public static   final  String TABLE_COUNTER       = "saveCounterTable";
    public static   final  String TABLE_PASSWORD      = "passwordTable";


    public static   final  String COLUMN_ID           = "_id";
    public static   final  String COLUMN_CATEGORY     = "category";
    public static   final  String COLUMN_AMOUNT       = "amount";
    public static   final  String COLUMN_DATE         = "date";
    public static   final  String COLUMN_ACCOUNT      = "account";
    public static   final  String COLUMN_TRANSACTION  = "incomeExpenses";


    public static   final  String COLUMN_SALARY       = "salary";

    public static   final  String COLUMN_PASSWORD     = "password";


    public dbHandler(Context context, String name,
SQLiteDatabase.CursorFactory factory,
                     int version) {
        super(context, DATABASE_NAME, factory, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {

        //Main table that handles incomes/outcomes,transactions etc.
        String query1 = "CREATE TABLE " + TABLE_TRANSACTIONS + "(" +
```

```
                COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
                COLUMN_DATE + " TEXT, " +
                COLUMN_CATEGORY + " TEXT, " +
                COLUMN_AMOUNT + " DOUBLE, " +
                COLUMN_ACCOUNT + " TEXT, " +
                COLUMN_TRANSACTION + " TEXT " +
                ");";

        //Table to store the total remaining salary.
        String query2 = "CREATE TABLE " + TABLE_TOTALSAL + "(" +
                COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
                COLUMN_SALARY + " DOUBLE, " +
                COLUMN_DATE + " INTEGER " +
                ");";

        //Table to store the user's salary.
        String query3 = "CREATE TABLE " + TABLE_SAVESALARY + "(" +
                COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
                COLUMN_SALARY + " DOUBLE " +
                ");";

        //Table to store a counter to check the number of times total
fragment is loaded.
        String query4 = "CREATE TABLE " + TABLE_COUNTER + "(" +
                COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
                COLUMN_TRANSACTION + " INTEGER, " +
                COLUMN_DATE + " INTEGER " +
                ");";

        //Table to store the user's password.
        String query5 = "CREATE TABLE " + TABLE_PASSWORD + "(" +
                COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
                COLUMN_PASSWORD + " TEXT " +
                ");";



        db.execSQL(query1);
        db.execSQL(query2);
        db.execSQL(query3);
        db.execSQL(query4);
        db.execSQL(query5);


    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
{
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_TRANSACTIONS);
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_TOTALSAL);
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_SAVESALARY);
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_COUNTER);
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_PASSWORD);

        onCreate(db);
```

```
    }

    //Adding new row in the table.
    public void addRow(Transactions transaction){
        ContentValues values  = new ContentValues();
        ContentValues values1 = new ContentValues();
        ContentValues values2 = new ContentValues();


        //First table
        values.put(COLUMN_DATE,transaction.get_date());
        values.put(COLUMN_CATEGORY,transaction.get_category());
        values.put(COLUMN_ACCOUNT,transaction.get_account());
        values.put(COLUMN_AMOUNT,transaction.get_amount());
        values.put(COLUMN_TRANSACTION,transaction.get_transaction());


        //Second table
        values1.put(COLUMN_SALARY, transaction.get_salary());
        values1.put(COLUMN_DATE,transaction.get_date());


        //Third table
        values2.put(COLUMN_SALARY, transaction.get_salary());


        SQLiteDatabase db = getWritableDatabase();

        db.insert(TABLE_TRANSACTIONS, null, values);
        db.insert(TABLE_TOTALSAL, null, values1);
        db.insert(TABLE_SAVESALARY, null, values2);


        db.close();
    }

    //Printing the database results in the textViews that we have created.
    public String dbToString(int view){

        String dbString = "";
        SQLiteDatabase db = getWritableDatabase();
        String query = "SELECT * FROM " + TABLE_TRANSACTIONS + " WHERE " +
COLUMN_TRANSACTION +
                "=\"" + view + "\";";

        //Cursor points to a location in your results
        Cursor c = db.rawQuery(query, null);
        //Move to the first row in your results
        c.moveToFirst();

        //Position after the last row means the end of the results
        while (!c.isAfterLast()) {
            if (c.getString(c.getColumnIndex("_id")) != null) {
                dbString += c.getString((c.getColumnIndex("_id")));
                dbString += "\t";
                dbString += c.getString(c.getColumnIndex("date"));
                dbString += "\t";
```

```
                dbString += c.getString(c.getColumnIndex("amount"));
                dbString += "€ \t";
                dbString += c.getString(c.getColumnIndex("account"));
                dbString += "\t";
                dbString += c.getString(c.getColumnIndex("category"));
                dbString += "\n";
            }
            c.moveToNext();
        }
        c.close();
        db.close();
        return dbString;

    }

    //Deleting all entries from the database without dropping the table.
    public void clearDB(){
        SQLiteDatabase db = getWritableDatabase();
        String query  = "DELETE FROM " + TABLE_TRANSACTIONS + " WHERE 1";
        String query2 = "DELETE FROM " + TABLE_SAVESALARY + " WHERE 1";
        String query3 = "DELETE FROM " + TABLE_TOTALSAL + " WHERE 1";

        db.execSQL(query);
        db.execSQL(query2);
        db.execSQL(query3);
    }


    //Calculating the total income for the total fragment.
    public double totalIncome(){
        SQLiteDatabase db = getWritableDatabase();
        String query = "SELECT SUM(" + COLUMN_AMOUNT + " ) " + " AS
IncomeTotal " +
                " FROM " + TABLE_TRANSACTIONS +
                " WHERE " + COLUMN_TRANSACTION + "=\"" + 0 + "\";";
        Cursor c  = db.rawQuery(query, null);
        c.moveToFirst();


        totalIncDB = c.getDouble(c.getColumnIndex("IncomeTotal"));

        c.close();
        db.close();

        return totalIncDB;
    }


    //Calculating the total expenses for the total fragment.
    public double totalExpenses(){
        SQLiteDatabase db = getWritableDatabase();
        String query = "SELECT SUM(" + COLUMN_AMOUNT + " ) " + " AS
ExpensesTotal " +
                " FROM " + TABLE_TRANSACTIONS +
                " WHERE " + COLUMN_TRANSACTION + "=\"" + 1 + "\";";
        Cursor c  = db.rawQuery(query,null);
        c.moveToFirst();
```

```
        totalExpDB = c.getDouble(c.getColumnIndex("ExpensesTotal"));

        c.close();
        db.close();

        return totalExpDB;
    }


    //Function to increase the incomes each first of the month by salary
using transaction_code = 4.
    public double increaseIncome(){
        SQLiteDatabase db = getWritableDatabase();
        String query = "SELECT SUM(" + COLUMN_AMOUNT + " ) " + "AS
IncomeMonthly " +
                " FROM " + TABLE_TRANSACTIONS +
                " WHERE " + COLUMN_TRANSACTION + "=\"" + 4 + "\";";
        Cursor c  = db.rawQuery(query,null);
        c.moveToFirst();

        totalMonth = c.getDouble(c.getColumnIndex("IncomeMonthly"));

        c.close();
        db.close();

        return totalMonth;
    }


    //Function to save the total income including the salary.
    public void saveSalary(double salary,String date){
        SQLiteDatabase db = getWritableDatabase();
        String query = "INSERT INTO " + TABLE_TOTALSAL +
                " (" + COLUMN_SALARY + " , " + COLUMN_DATE +
                " ) VALUES('"+salary+"','"+date+"');";
        Cursor c  = db.rawQuery(query,null);
        c.moveToFirst();


        c.close();
        db.close();
    }


    //Function to obtain the total income(including salary) from the db.
    public double getSalary(){
        SQLiteDatabase db = getWritableDatabase();
        String query = "SELECT SUM(" + COLUMN_SALARY +" ) " + "AS TotalSalary
" +
                " FROM " + TABLE_TOTALSAL +
                " WHERE 1";
        Cursor c  = db.rawQuery(query,null);
        c.moveToFirst();

        totalSal = c.getDouble(c.getColumnIndex("TotalSalary"));
        System.out.println("TOTAL_SAL ----> " + totalSal);
```

```
        c.close();
        db.close();


        return totalSal;
    }

    //Get the number of months in the database(in the TOTAL SALARY
specifically) that are in terms
    //of integer value lower than the current month passed as a parameter in
the method.
    public int checkRepeat(int month){
        int count;

        SQLiteDatabase db = getWritableDatabase();

        String query = "SELECT " + COLUMN_ID + " FROM " + TABLE_TOTALSAL +
                        " WHERE " + COLUMN_DATE + "<\"" + month + "\";";


        try {
            Cursor c  = db.rawQuery(query, null);
            count = c.getCount();

            c.close();
        }
        catch (Exception e){
            count = -1;
        }

        db.close();

        return count;
    }

    //Saving the salary once every month.
    public void totalSaveRepeat(double salary){
        SQLiteDatabase db = getWritableDatabase();
        String query = "INSERT INTO " + TABLE_SAVESALARY +
                "  (" + COLUMN_SALARY +
                "  ) VALUES('"+salary+"');";
        Cursor c  = db.rawQuery(query,null);
        c.moveToFirst();


        c.close();
        db.close();

    }


    //Getting the salary(if any) and adding it to the other incomes.
    public double gettotalSaveRepeat(){
        SQLiteDatabase db = getWritableDatabase();
        String query = "SELECT SUM(" + COLUMN_SALARY +" ) " + "AS saveSalary
" +
```

```
            " FROM " + TABLE_SAVESALARY +
            " WHERE 1";
    Cursor c  = db.rawQuery(query,null);
    c.moveToFirst();

    totalSal = c.getDouble(c.getColumnIndex("saveSalary"));
    System.out.println("SAVE_SAL ----> " + totalSal);

    c.close();
    db.close();
    return totalSal;

}


//Printing the total salary array for examination purposes.
public String salaryToString() {
    String dbString = "";
    SQLiteDatabase db = getWritableDatabase();
    String query = "SELECT * FROM " + TABLE_TOTALSAL + " WHERE 1";

    //Cursor points to a location in your results
    Cursor c = db.rawQuery(query, null);
    //Move to the first row in your results
    c.moveToFirst();

    while (!c.isAfterLast()) {
        if (c.getString(c.getColumnIndex("_id")) != null) {

            dbString += c.getString(c.getColumnIndex("salary"));
            dbString += " ";
            dbString += c.getString(c.getColumnIndex("_id"));
            dbString += " ";
            dbString += c.getString(c.getColumnIndex("date"));

        }

        c.moveToNext();
    }


    c.close();
    db.close();

    return dbString;
}


//Table used to store the value of the counter along with the MONTH.
public void saveCounter(int counter,int date){
    SQLiteDatabase db = getWritableDatabase();
    String query = "INSERT INTO " + TABLE_COUNTER +
            " (" + COLUMN_TRANSACTION + " , " + COLUMN_DATE +
            " ) VALUES('"+counter+"','"+date+"');";
    Cursor c  = db.rawQuery(query,null);
    c.moveToFirst();
```

```
        c.close();
        db.close();
    }


    //Printing the values of the counter table for examination purposes.
    public String counterToString(){
        String dbString = "";
        SQLiteDatabase db = getWritableDatabase();
        String query = "SELECT * FROM " + TABLE_COUNTER + " WHERE 1";
        Cursor c = db.rawQuery(query, null);
        //Move to the first row in your results
        c.moveToFirst();

        while (!c.isAfterLast()) {
            if (c.getString(c.getColumnIndex("_id")) != null) {

                dbString += c.getString(c.getColumnIndex("incomeExpenses"));
                dbString += " ";
                dbString += c.getString(c.getColumnIndex("_id"));
                dbString += " ";
                dbString += c.getString(c.getColumnIndex("date"));

            }

            c.moveToNext();
        }

        c.close();
        db.close();

        return dbString;
    }


    //Getting the entries from the counter array that correspond to the
parameters(to check if we
    //just changed view in the layout without a month change happening).
    public int getCounter(int date){
        int total;

        SQLiteDatabase db = getWritableDatabase();
        String query = "SELECT " + COLUMN_ID +
                " FROM " + TABLE_COUNTER +
                " WHERE " + COLUMN_DATE + " =\"" + date + "\"" + " AND " +
                COLUMN_TRANSACTION + "=\"" + 1 + "\";";

        try{
            Cursor c  = db.rawQuery(query, null);
            total = c.getCount();

            c.close();
            db.close();

            return total;
        }
```

```
        catch (Exception e){
            db.close();

            return -1;
        }

        //c.moveToNext();
    }


    //Unused method,initially purposed to delete the unused counters from the
database.
    public void counterDelete(){
        SQLiteDatabase db = getWritableDatabase();
        String query = "DELETE FROM " + TABLE_COUNTER +
                " WHERE " + COLUMN_TRANSACTION + "=\"" + 1 + "\";";

        Cursor c = db.rawQuery(query,null);
        c.moveToFirst();

        c.close();
        db.close();
    }


    //Find a transaction in the database.
    public int findTransaction(int transid){
        int score;

        SQLiteDatabase db = getWritableDatabase();
        String query = "SELECT * FROM " + TABLE_TRANSACTIONS +
                " WHERE " + COLUMN_ID + "=\"" + transid + "\";";

        Cursor c = db.rawQuery(query,null);
        c.moveToFirst();
        score = c.getCount();


        c.close();
        db.close();

        return score;
    }


    //Displaying the transactions based on the id the user provides.
    public String transactionToString(int transid) {

        String transString = "";
        SQLiteDatabase db = getWritableDatabase();
        String query = "SELECT * FROM " + TABLE_TRANSACTIONS + " WHERE " +
COLUMN_ID +
                "=\"" + transid + "\";";

        //Cursor points to a location in your results
        Cursor c = db.rawQuery(query, null);
        //Move to the first row in your results
```

```
        c.moveToFirst();

        //Position after the last row means the end of the results
        while (!c.isAfterLast()) {
            if (c.getString(c.getColumnIndex("_id")) != null) {
                transString += c.getString((c.getColumnIndex("_id")));
                transString += "\t";
                transString += c.getString(c.getColumnIndex("date"));
                transString += "\t";
                transString += c.getString(c.getColumnIndex("amount"));
                transString += "€ \t";
                transString += c.getString(c.getColumnIndex("account"));
                transString += "\t";
                transString += c.getString(c.getColumnIndex("category"));
                transString += "\n";
            }
            c.moveToNext();
        }

        c.close();
        db.close();

        return transString;

    }


    //Removing a row from the table.
    public void removeRow(int transid){
        SQLiteDatabase db = getWritableDatabase();
        db.execSQL("DELETE FROM " + TABLE_TRANSACTIONS + " WHERE " +
                COLUMN_ID + "=\"" + transid + "\";");

        db.close();
    }


    //Checking if there are any transactions between the two given dates.
    public int transactionsExistBetweenDates(String date1,String date2){
        int result;

        String transString = "";

        SQLiteDatabase db = getWritableDatabase();
        String query = "SELECT * FROM " + TABLE_TRANSACTIONS +
                " WHERE " + COLUMN_DATE + " BETWEEN '"+date1+"' AND
'"+date2+"'";

        Cursor c = db.rawQuery(query, null);

        //Move to the first row in your results
        c.moveToFirst();

        result = c.getCount();

        c.close();
        db.close();
```

```
        return result;
    }


    //Searching for specified transactions.
    public String searchTransaction(String date1, String date2){

        String transString = "";

        SQLiteDatabase db = getWritableDatabase();
        String query = "SELECT * FROM " + TABLE_TRANSACTIONS +
                " WHERE " + COLUMN_DATE + " BETWEEN '"+date1+"' AND
'"+date2+"'";

        Cursor c = db.rawQuery(query,null);

        //Move to the first row in your results
        c.moveToFirst();

        //Position after the last row means the end of the results
        while (!c.isAfterLast()) {
            if (c.getString(c.getColumnIndex("_id")) != null) {
                transString += c.getString((c.getColumnIndex("_id")));
                transString += "\t";
                transString += c.getString(c.getColumnIndex("date"));
                transString += "\t";
                transString += c.getString(c.getColumnIndex("amount"));
                transString += "€ \t";
                transString += c.getString(c.getColumnIndex("account"));
                transString += "\t";
                transString += c.getString(c.getColumnIndex("category"));
                transString += "\n";
            }
            c.moveToNext();
        }

        c.close();
        db.close();


        return transString;
    }


    //Checking if password is already in the database.
    public int passwordExistToDb(){
        int existence;

        SQLiteDatabase db = getWritableDatabase();
        String query = "SELECT * FROM " + TABLE_PASSWORD +
                " WHERE 1";

        Cursor c = db.rawQuery(query,null);
        c.moveToFirst();

        existence = c.getCount();
```

```
    c.close();
    db.close();

    return existence;
}


//Update an already stored password.
public void updatePasswordToDb(String password) {
    SQLiteDatabase db = getWritableDatabase();
    String query = "UPDATE " + TABLE_PASSWORD +
            " SET " + COLUMN_PASSWORD  + "=\"" + password + "\";";

    Cursor c  = db.rawQuery(query,null);
    c.moveToFirst();


    c.close();
    db.close();
}


//Add a new password in the database.
public void addPasswordToDb(String password){
    SQLiteDatabase db = getWritableDatabase();
    String query = "INSERT INTO " + TABLE_PASSWORD +
            " (" + COLUMN_PASSWORD + " ) VALUES('"+password+"');";
    Cursor c  = db.rawQuery(query,null);
    c.moveToFirst();


    c.close();
    db.close();
}


//Checking old password equality with the entered one.
public int passwordChecking(String password){
    int equality;

    SQLiteDatabase db = getWritableDatabase();
    String query = "SELECT * FROM " + TABLE_PASSWORD +
            " WHERE " + COLUMN_PASSWORD + "=\"" + password + "\";";

    Cursor c  = db.rawQuery(query,null);
    c.moveToFirst();

    equality = c.getCount();

    c.close();
    db.close();

    return equality;
}
```

```
    //Print password array.
    public String passwordPrint(){
        String transString = "";

        SQLiteDatabase db = getWritableDatabase();
        String query = "SELECT * FROM " + TABLE_PASSWORD +
                " WHERE 1";

        Cursor c = db.rawQuery(query,null);

        //Move to the first row in your results
        c.moveToFirst();

        while (!c.isAfterLast()) {
            if (c.getString(c.getColumnIndex("_id")) != null) {
                transString += c.getString((c.getColumnIndex("_id")));
                transString += c.getString(c.getColumnIndex("password"));
                transString += "\t";
            }
            c.moveToNext();
        }

        c.close();
        db.close();


        return transString;
    }


    //Public method to determine if there are any transactions for the month
selected.
    public int yearlyTransactions(String date1, String date2){
        int existence;

        System.out.println("DATE1 ---->"+date1);
        System.out.println("DATE2 ---->"+date2);

        SQLiteDatabase db = getWritableDatabase();
        String query = "SELECT * FROM " + TABLE_TRANSACTIONS +
                " WHERE " + COLUMN_DATE + " BETWEEN '"+date1+"' AND
'"+date2+"'";

        Cursor c = db.rawQuery(query,null);
        c.moveToFirst();

        existence = c.getCount();

        c.close();
        db.close();

        return existence;
    }


    //Public method to display yearly statistics.
    public String showTransactionsYearly(String date1,String date2){
```

```
        String dbString = "";
        String start_date = "From: ";
        String end_date = " To: ";
        String stats = " the statistics are: ";
        String total_income = "Total Income: ";
        String total_expenses = "Total Expenses: ";
        String total_transactions = "Total number of transactions is: ";
        String income_trans = "Total number of income transactions: ";
        String expenses_trans = "Total number of expenses transactions: ";

        double total_inc,total_exp;
        int transactions_number,inc_transactions,exp_transactions;

        SQLiteDatabase db = getWritableDatabase();


        String query = "SELECT * FROM " + TABLE_TRANSACTIONS +
                " WHERE " + COLUMN_DATE + " BETWEEN '"+date1+"' AND
'"+date2+"'";

        String query2 = "SELECT SUM(" + COLUMN_AMOUNT + " ) " + " AS
IncomeTotal " +
                " FROM " + TABLE_TRANSACTIONS +
                " WHERE " + COLUMN_TRANSACTION + "=\"" + 0 + "\";";

        String query3 = "SELECT SUM(" + COLUMN_AMOUNT + " ) " + " AS
ExpensesTotal " +
                " FROM " + TABLE_TRANSACTIONS +
                " WHERE " + COLUMN_TRANSACTION + "=\"" + 1 + "\";";

        String query4 = "SELECT * FROM " + TABLE_TRANSACTIONS +
                " WHERE " + COLUMN_DATE + " BETWEEN '"+date1+"' AND
'"+date2+"'";

        String query5 = "SELECT * FROM " + TABLE_TRANSACTIONS +
                " WHERE " + COLUMN_TRANSACTION + "=\"" + 0 + "\";";

        String query6 = "SELECT * FROM " + TABLE_TRANSACTIONS +
                " WHERE " + COLUMN_TRANSACTION + "=\"" + 1 + "\";";



        //Cursor points to a location in your results
        Cursor c  = db.rawQuery(query, null);
        Cursor c1 = db.rawQuery(query2, null);
        Cursor c2 = db.rawQuery(query3, null);
        Cursor c3 = db.rawQuery(query4, null);
        Cursor c4 = db.rawQuery(query5, null);
        Cursor c5 = db.rawQuery(query6, null);


        //Move to the first row in your results
        c.moveToFirst();
        c1.moveToFirst();
        c2.moveToFirst();
        c3.moveToFirst();
        c4.moveToFirst();
```

```
    c5.moveToFirst();


    total_inc = c1.getDouble(c1.getColumnIndex("IncomeTotal"));
    total_exp = c2.getDouble(c2.getColumnIndex("ExpensesTotal"));
    transactions_number = c3.getCount();
    inc_transactions = c4.getCount();
    exp_transactions = c5.getCount();

    start_date += date1;
    start_date += end_date;
    start_date += date2;
    start_date += stats;
    start_date += "\n";
    start_date += "##############################";
    start_date += "\n";
    start_date += total_income;
    start_date += total_inc;
    start_date += "\n";
    start_date += "-------------------------------------------------";
    start_date += "\n";
    start_date += total_expenses;
    start_date += total_exp;
    start_date += "\n";
    start_date += "-------------------------------------------------";
    start_date += "\n";
    start_date += total_transactions;
    start_date += transactions_number;
    start_date += "\n";
    start_date += "-------------------------------------------------";
    start_date += "\n";
    start_date += income_trans;
    start_date += inc_transactions;
    start_date += "\n";
    start_date += "-------------------------------------------------";
    start_date += "\n";
    start_date += expenses_trans;
    start_date += exp_transactions;
    start_date += "\n";
    start_date += "-------------------------------------------------";
    start_date += "\n";




    c.close();
    c1.close();
    c2.close();
    c3.close();
    c4.close();
    c5.close();

    db.close();


    return start_date;
}
```

```
}
```

8.11 Κωδικας για Transactions Κλαση

```
package com.example.dimitris.moneyger;


//Class that implements setter and getter methods used in the database loads and
writes.
public class Transactions {

    private     int         _id;
    private     String      _date;
    private     String      _category;
    private     String      _account;
    private     double      _amount;
    private     int         _transaction;
    private     double      _salary;

    //Constructor
    public Transactions(String date,String category,String account,double amount,
                        int transaction,double salary) {
        this._date          = date;
        this._category      = category;
        this._account       = account;
        this._amount        = amount;
        this._transaction   = transaction;
        this._salary        = salary;
    }

    public Transactions() {
    }

    //Setter methods to set the values of the java properties equal to UI entered
values.

//##############################################################################################
##########
    public void set_id(int _id) {
        this._id = _id;
    }

    public void set_date(String _date) {
        this._date = _date;
    }

    public void set_category(String _category) {
        this._category = _category;
    }

    public void set_amount(double _amount) {
        this._amount = _amount;
    }

    public void set_transaction(int _transaction) {
```

```
        this._transaction = _transaction;
    }

    public void set_account(String _account) {
        this._account = _account;}

    public void set_salary(double _salary) {
        this._salary = _salary;
    }


//#########################################################################################
##########


    //Getter methods to obtain values from the java properties.

//#########################################################################################
##########
    public int get_id() {
        return _id;
    }

    public String get_date() {
        return _date;
    }

    public String get_category() {
        return _category;
    }

    public double get_amount() {
        return _amount;
    }

    public int get_transaction() {
        return _transaction;
    }

    public String get_account() {
        return _account;
    }

    public double get_salary() {
        return _salary;
    }


//#########################################################################################
##########


}
```

8.12 ΚΩΔΙΚΑΣ ΓΙΑ ROUNDDECIMALS ΚΛΑΣΗ

```
package com.example.dimitris.moneyger;

public class RoundDecimals {

    public RoundDecimals() {
    }

    //Allowing only two decimals and rounding to the closest double.
    public double roundTwoDecimals(double number){

        return Math.floor(number * 100.0) / 100.0;
    }
}
```

8.13 ΚΩΔΙΚΑΣ ΓΙΑ MONTHLYINCOME ΚΛΑΣΗ

```
package com.example.dimitris.moneyger;


//Class that is used in order to calculate the TOTAL income including the salary.
public class MonthlyIncome {

    public MonthlyIncome() {
    }


    public double totalIncomeSal(double salary,double income){
        income = income + salary;

        return income;
    }
}
```

8.14 ΚΩΔΙΚΑΣ ΓΙΑ CONFIRM_DROP_DB.XML

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

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="250dp"
        android:layout_gravity="left|top">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceMedium"
```

```
            android:text="@string/entrance_authorization"
            android:id="@+id/textView9"
            android:textStyle="bold"
            android:layout_gravity="center_horizontal|top" />

        <EditText
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:inputType="textPassword"
            android:ems="10"
            android:id="@+id/editTextPasswordGet"
            android:layout_gravity="center_horizontal|bottom" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="@string/password_prompt"
            android:id="@+id/textView11"
            android:textStyle="bold"
            android:layout_gravity="center" />
    </FrameLayout>
</FrameLayout>
```

8.15 ΚΩΔΙΚΑΣ ΓΙΑ PASSWORD.XML

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

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="185dp"
        android:layout_gravity="left|top">

        <EditText
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:inputType="textPassword"
            android:ems="10"
            android:id="@+id/editTextPassword"
            android:layout_gravity="center" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="@string/password_prompt"
            android:id="@+id/textView6"
            android:layout_gravity="center_horizontal|top" />

    </FrameLayout>

</FrameLayout>
```

8.16 ΚΩΔΙΚΑΣ ΓΙΑ CONFIRM_PASSWORD.XML

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

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:layout_gravity="left|top">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="@string/old_title"
            android:textStyle="bold"
            android:id="@+id/textView8"
            android:layout_gravity="center_horizontal|top" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="@string/new_pass_msg"
            android:textStyle="bold"
            android:id="@+id/textViewNew"
            android:layout_gravity="left|center_vertical" />

        <EditText
            android:layout_width="150dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="50dp"
            android:inputType="textPassword"
            android:ems="10"
            android:id="@+id/editTextOld"
            android:layout_gravity="right|top" />

        <EditText
            android:layout_width="150dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:inputType="textPassword"
            android:ems="10"
            android:id="@+id/editTextNew"
            android:layout_gravity="right|center_vertical" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="50dp"
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="@string/old_pass_msq"
            android:textStyle="bold"
            android:id="@+id/textViewOld"
            android:layout_gravity="left|top" />
    </FrameLayout>

</FrameLayout>
```

8.17 ΚΩΔΙΚΑΣ ΓΙΑ PROMPTS.XML

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:weightSum="1">

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="189dp"
        android:layout_gravity="center_horizontal|top">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="@string/dialogEndDate"
            android:id="@+id/textViewEndDate"
            android:textStyle="bold"
            android:layout_gravity="left|bottom" />

        <EditText
            android:layout_width="160dp"
            android:layout_height="wrap_content"
            android:inputType="date"
            android:ems="10"
            android:id="@+id/editTextEndDate"
            android:background="@drawable/back"
            android:clickable="false"
            android:cursorVisible="false"
            android:focusable="false"
            android:focusableInTouchMode="false"
            android:layout_gravity="right|bottom" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textStyle="bold"
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="@string/dialogStartDate"
            android:id="@+id/textViewStartDate"
            android:layout_gravity="left|center_vertical" />

        <EditText
            android:layout_width="160dp"
            android:layout_height="wrap_content"
            android:inputType="date"
            android:ems="10"
            android:id="@+id/editTextStartDate"
            android:background="@drawable/back"
            android:clickable="false"
            android:cursorVisible="false"
            android:focusable="false"
            android:focusableInTouchMode="false"
            android:layout_gravity="right|center_vertical" />
```

```
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="@string/dialogTitle"
            android:textStyle="bold"
            android:id="@+id/textView10"
            android:layout_gravity="center_horizontal|top" />

    </FrameLayout>

</FrameLayout>
```

---

9 ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Επίσημος οδηγός προγραμματισμού σε Android Studio:
http://developer.android.com/tools/studio/index.html

[2] Επίσημος οδηγός προγραμματισμού για συσκευές Android:
http://developer.android.com/guide/index.html

[3] Beginning Android Application Development by Wei-Meng Lee

[4] Sams Teach Yourself Android Application Development in 24 Hours by Lauren Darcey & Shane Conder

[5] https://stackoverflow.com/

[6] http://www.mysamplecode.com/

[7] http://android-holo-colors.com/

[8] https://romannurik.github.io/AndroidAssetStudio/icons-launcher.html

[9] http://www.vogella.com/tutorials/android.html

[10] https://www.thenewboston.com/