# CashApp Documentation

1. **Introduction**

CashApp is a financial manager and budget analyzer built to help the user plan their budget ahead by making wishlists, manage credit cards and cash money. It can be used on the go every time a transaction is made or when the user has a new wish as well as to plan ahead and not make any bad decisions.

1. **Description**

The application is designed for daily use and has a simple and minimalist interface that any user can understand.

Regarding their cash money, the user can add, check, analyze and save to the database the list of transactions made. After they have added a cash transaction, they can see the date, the satisfaction (or rating) provided by that operation, the amount exchanged, check if the transaction was planned or spontaneous or if it was made from the user’s savings. The user can also see a graph that shows the correlation between the amount of cash money involved in a transaction and the satisfaction it provides and observes any abnormalities.

The management of Credit Cards is an important feature. The user can manage their accounts as well as their transactions on said accounts. The limit of the credit cards is a set number, updated every time the user spends money on that card. The Bank is also included with the accounts, helping the customer decide which card to use in different situations.

Another action that can be done using CashApp is adding wishes to the Wishlist and seeing them. Characteristics of a wish include the name, the importance (on a scale from 1 to 5), the deadline (for example, for a trip or a limited sale), the cost of that wish, the type as in Experience or Material Good and the frequency of that expenditure. Another feature of the CashApp is the ability to pick a holiday destination based on suggestions provided directly on Google Maps, with a simple click.

There are multiple ways to persist the data in the CashApp: databases with connections between them, .txt files and .csv files and a well-organized JSON file easy to read.

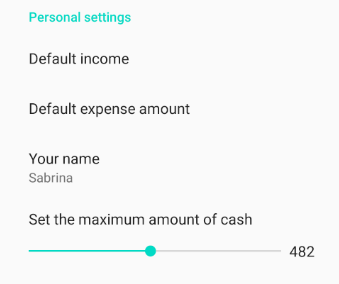
With the use of Preferences, the user can have a better experience using the application. This allows the customer to set default values and cash amount limits in order to make the addition of a transaction much faster and easier.

1. **Implementation**

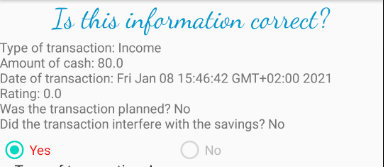
**The Cash activities**

The MainCashActivity contains the form used to input all the data needed for the Cash objects. There are various controls used for this, like RadioButtons, ProgressBar, RatingBar, Switch, etc. On that Activity, there is also an image animated to fade in/out when clicked, depending on its current state.

There are validations for all the controls and default values adjusted to help the user fill out the form faster; for example, the date, if left empty, is by default the date of completion of the form; the cash amount can be set by default by the user using Preferences which can be accessed from the main menu. The maximum amount that can be received/spent in a transaction can also be set using the user Preferences.



Submitting the form moves to the next activity where the user is asked if the selected information is correct. The data between activities/fragments is sent using Intents. If the data is not correct the user is asked once again, just to be sure, and if it is still incorrect the activity with the form appears again, allowing the user to edit their first input.

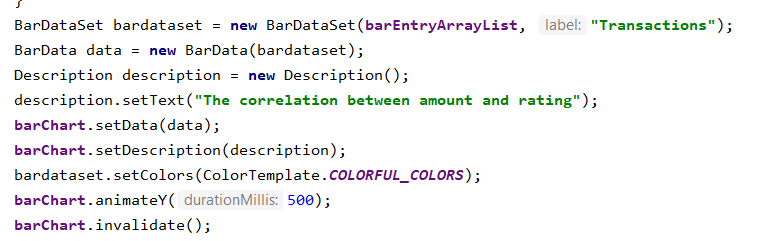


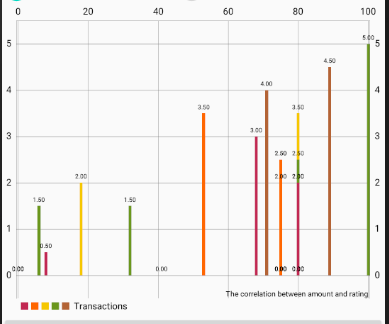
If the data is correct, then it is added to the ListView, to the database and to the JSON file, all containing the entire history. The JSON file has three levels, sorted on the type of Cash (Expense, Income or Other) and the database contains the table Cash. The database was formed using SQLiteOpenHelper (Doinea, 2020). The ListView appears in a Fragment.

From this point, the user can go back to add more Cash transactions or can view a Chart. If they click on the Show Chart button they can see the chart containing all Cash transactions or they can click on any ListView item and the chart will appear with the clicked item highlighted. The Bar Chart appears in a fragment.

The Bar Chart used for the Cash analysis is the MPAndroidChart library created by Philip Jahoda. (Jahoda, 2016)

It is a simple library that displays data made of BarEntry entities which take two arguments: the value on the X-axis and the value on the Y-axis. The chart can be customized by changing the colours of the bars, the description and the title, the duration of the animation, etc.





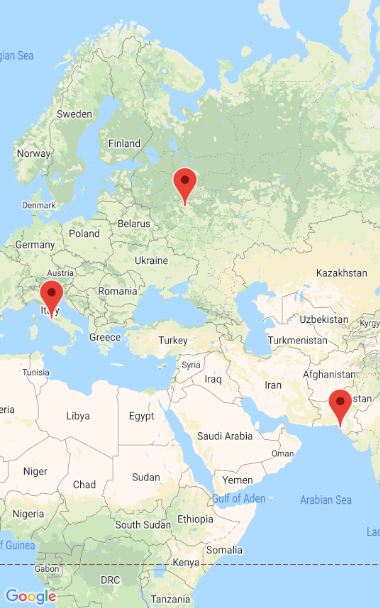
**The WishList activities**

The wishList functionality is made out of three activities: the main one in which the list of wishes is displayed; the form which is used to add a new wish and the Maps activity in which a few good vacation spots are marked, for the user to choose from.

When the main activity is created, it starts 2 threads that were implemented using Runnable interface, which are used to download a couple of pictures. These pictures are then sent to a custom adapter which is used to display two hard-coded wishes examples: an experience (a trip to USA) and a material good (a car). These are displayed inside a gridview. (Fig. 2)

Another thread which was made using AsyncTask is used to send a JSON object containing an array of the 2 hard-coded wishes to a server, using a POST request method.

Following the gridview, there is a button that starts the form activity to add a wish to the listview. For the listview I used an ArrayAdapter. The form has a few controllers such as EditText and spinner, and provides some simple data validation. It also has a cancel button to return to the main wish activity, and a “See Suggestions” button that shows the marked map. The user can select a marker and it will auto-complete the wish name with the name of the location.

****

When the user adds a wish, this is sent in a bundle back to the main activity, where it is added to the local array of Wish objects and to the database. The database was made with SQLite.

Problems/advantages using Android Studio

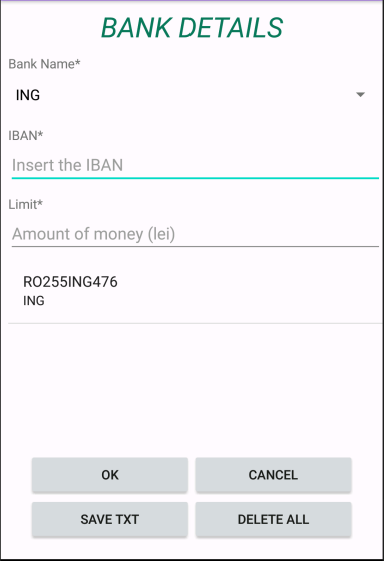
One of the problems of Android Studio encountered during the development of the application was the limited amount of controls, especially Graph controls (which have to be obtained from third-parties). Another problem was that it requires too many resources.

An advantage of Android Studio is that the designer is very intuitive.

**The Credit Cards activities**

In this section the user can use a credit card in order to manage his budget. For doing this, he has to enter the IBAN and a limit of account useful in the situation when he wants to reserve only a sum of money for expenses. For example, when a transaction occurs, the amount of that transaction will be reduced until the limit gets value 0. When a limit is 0 the only chance to use the card is to get an income.

On the first activity it is possible to see all the accounts used so far, but also, the application offers the opportunity to save the list of the accounts in a text file. It can be used as a report. A last option is to delete all the data from the list, but also from the database, where the accounts’ details are stored. All these options can be called from the buttons placed at the bottom of the page. This can be seen as follows:

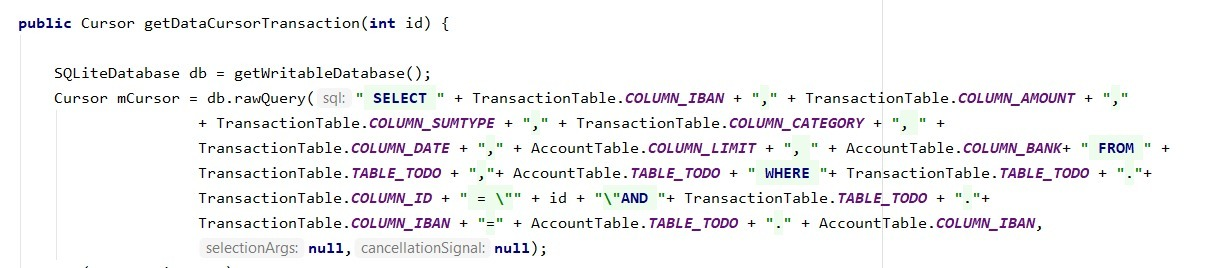


All the transactions can be done on three different categories: Food, Clothes or Services. Also, the user has to insert a specific date for when the transaction takes place. All the information is mandatory and when a field is not completed, the application will display errors regarding the problem. An example is the one presented in the figure 3.

The next activity asks for validation, if the data is correctly inserted. If the user choses “No”, then he has the possibility to edit his information, else, the information is displayed on the next page. There exist three buttons: one for delete, one for saving the information in a text file as report and one for returning to the bank details when the user has the opportunity to insert another transaction.

In this section, almost everything is related to the database which contains two tables: AccountTable, which contains the information associated to the accounts and TransactionTable, being correlated to the transactions. All the DLL(Create, Drop, Alter) and DML (Select, Update, Delete and Insert) are implemented. The Update option is not visible in the interface because it is automatically called when a transaction is done. It means whenever a sum of money is substracted or added to the account, the limit of it is also changed, depending on the operation.

For the list of the final activity we used a cursor to make a “select” query. This operation is used to display all the information from the tables. They are connected through the field IBAN, which is the id of the AccountTable. On this connection, we could make a join between tables and obtain all the information from the database. The code for this operation is presented below:



1. **Conclusions**

CashApp can be further improved by adding more interactive graphs, better forms to add transactions/wishes/accounts and overall more features. The application could include videos/tips/lessons for financial education and more efficient management of the money.

# Bibliography

Jahoda, P. (2016, March 20). *MPAndroidChart.* Retrieved from Github: https://github.com/PhilJay/MPAndroidChart

Annex

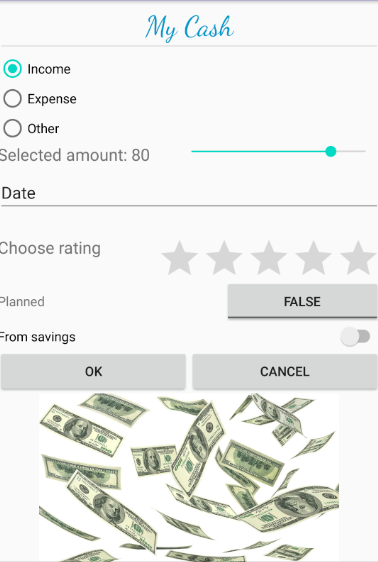


Figure 1: Form to input a Cash transaction

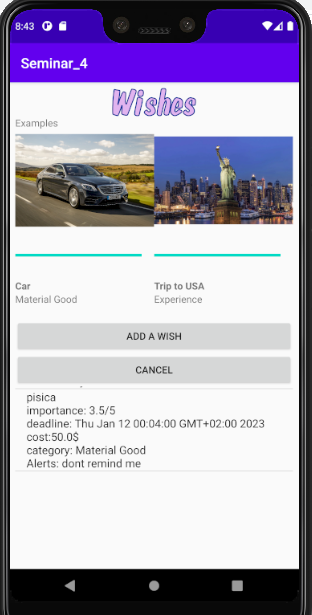


Figure 2: The gridview featuring the custom adapter

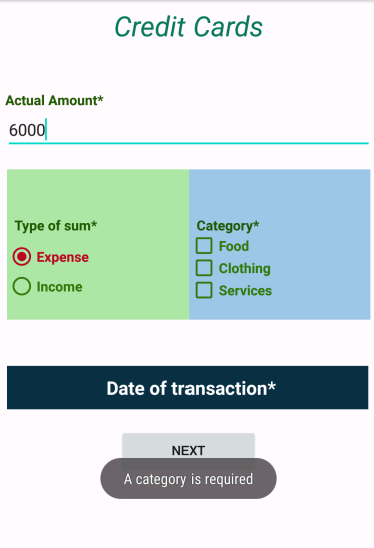


Figure 3: The error received

Developers:

Hriscu Bogdan

Lepîrda Damon-Gabriel

Lupșan Sabrina

Group 1093