Blackboard design document

Boriero Stefano 876106
Brunitti Simone 875039

April 2018

POLITECNICO
MILANO 1863

Contents

1 Introduction **4**
1.1 Scope . . . 4
1.2 Glossary . . . 4

2 Business Requirements **5**
2.1 Functionalities . . . 5
2.2 Goals . . . 5
2.3 Traceability Matrix . . . 6

3 Technical Requirements **7**
3.1 Device Requirements . . . 7
3.2 External Services . . . 7
3.2.1 Firebase Firestore . . . 7
3.2.2 Firebase Cloud Functions . . . 7
3.2.3 Firebase Cloud Messaging . . . 7
3.2.4 Firebase Cloud Storage . . . 7
3.2.5 Firebase Authentication . . . 8
3.2.6 Firebase Test Lab . . . 8
3.2.7 chron-job . . . 8
3.3 External Dependencies . . . 8
3.4 Multi Threading . . . 8
3.5 Device Sensors . . . 8

4 Application Flow **9**
4.1 Navigation Patterns . . . 9
4.2 Create an item . . . 9
4.3 Add a payment . . . 10
4.4 Take in charge an item . . . 10
4.5 Complete an item . . . 10
4.6 Change profile image . . . 11
4.7 Add a member to a house . . . 11
4.8 Leave a house . . . 11
4.9 Create a house . . . 12

5 Activities **13**
5.1 Main Activity . . . 13
5.1.1 Description . . . 13
5.1.2 Components . . . 13
5.2 Balance Activity . . . 15
5.2.1 Description . . . 15
5.2.2 Functionalities . . . 15
5.2.3 Components . . . 15
5.2.4 Goals achieved . . . 16
5.3 New Payment Activity . . . 17

5.3.1 Description . . . 17
5.3.2 Functionalities . . . 17
5.3.3 Components . . . 17
5.3.4 Goals achieved . . . 18
5.4 New To Do Task Activity . . . 19
5.4.1 Description . . . 19
5.4.2 Functionalities . . . 19
5.4.3 Components . . . 19
5.4.4 Goals achieved . . . 20
5.5 Settings Activity . . . 21
5.5.1 Description . . . 21
5.5.2 Components . . . 21
5.6 List Activity . . . 22
5.6.1 Description . . . 22
5.6.2 Functionalities . . . 22
5.6.3 Layout . . . 23
5.6.4 Behaviour logic . . . 24
5.6.5 Goals achieved . . . 24
5.7 Profile Activity . . . 25
5.7.1 Description . . . 25
5.7.2 Functionalities . . . 25
5.7.3 Components . . . 25
5.7.4 Goals achieved . . . 26
5.8 House Info Activity . . . 27
5.8.1 Description . . . 27
5.8.2 Functionalities . . . 27
5.8.3 Components . . . 27
5.8.4 Goals achieved . . . 28
5.9 Login Activity . . . 29
5.9.1 Description . . . 29
5.9.2 Components . . . 29
5.10 Sign up Activity . . . 30
5.10.1 Description . . . 30
5.10.2 Functionalities . . . 30
5.10.3 Components . . . 30
5.10.4 Goals achieved . . . 30

6 Database **31**
6.1 Description . . . 31
6.2 Data Structure . . . 31
6.3 Data Integrity . . . 32

7 References **33**

8 Future Works **34**

1 Introduction

1.1 Scope

The purpose of this document is to present the design decisions that had been taken during both the design and development phase of the application. In the first section we're going to present which are the requirement analysis that had been carried out: after that, we will focus on the requirements a device needs to meet in order to correctly run the application.
We will then delve into the details of the implementation, presenting the overall application flow alongside with some specific flows for the most relevant actions. A brief description of every activity is going to conclude the insight on the actual android application, then we're going to present the database structure underlying the project.
At the end of the document we'll provide useful links to technologies we used in the development process as well as some future works to improve our work.

1.2 Glossary

Here's a table containing definitions for terms relative to the application

Table 1: Glossary

TERM	**MEANING**
Roommate	Users belonging to the same house
House	Group of roommates sharing a list of items and payments
Item	Task that must be completed by a roommate
Payment	Item that requires money
Balance	Resume of all the payments

2 Business Requirements

2.1 Functionalities

The following are the main functionalities our application must have:

- **F.1** enable users to create a House
- **F.2** enable users to add an item
- **F.3** enable users to take in charge an item
- **F.4** enable users to complete an item
- **F.5** enable users to delete an item
- **F.6** enable users to give up an item (if previously taken)
- **F.7** enable users to access their balance information
- **F.8** enable users to add a payment
- **F.9** enable users to delete a payment
- **F.10** enable users to access their personal info
- **F.11** enable users to change profile picture
- **F.12** enable users to access their list of houses
- **F.13** enable users to add a member to a house
- **F.14** enable users to leave a house
- **F.15** enable users to access some useful stats and achievements concerning their use of the application
- **F.16** enable users to register a profile

2.2 Goals

The main goals this system is meant to offer are the following:

- **G.1** allow users to manage a house
- **G.2** allow users to manage their balance
- **G.3** allow users to manage their profile

2.3 Traceability Matrix

GOAL	FUNCTIONALITIES
G.1	**F.1**, **F.2**, **F.3**, **F.4**, **F.5**, **F.6**
G.2	**F.7**, **F.8**, **F.9**
G.3	**F.10**, **F.11**, **F.12**, **F.13**, **F.14**, **F.15**, **F.16**

3 Technical Requirements

3.1 Device Requirements

In developing Blackboard, we aim at supporting multiple classes of devices. In particular, the application is designed to run on devices with Android 5.0 or later versions.
In order to have a responsive UI, we decided to support different screen sizes, by having multiple layouts for some meaningful activities. The following are the breakpoints we identified, all referring to the smallest width:

- smallest width < 600dp ($\sim$ 4 to 7 inches) ->smartphone version
- smallest width > 600dp ($\sim$ 7 to 10 inches) ->tablet version

Regarding memory usage, we tested the application on different hardware configurations, and we suggest a minimum RAM requirement of 1GB. Finally, given the connected nature of our application, the device must have a reliable internet connection.

3.2 External Services

3.2.1 Firebase Firestore

Since we need to store many user data (e.g. items, payments etc...) we use Firebase Firestore to save documents on a cloud platform, which will be retrieved when the user needs to access his own information. Further details on the database can be found in the Database section of this document.

3.2.2 Firebase Cloud Functions

In order to mantain the referential integrity of our data, we need to exploit the Firebase Cloud Functions service. This allows us to write functions that will be triggered by events in our database, such as document insertion or modification.

3.2.3 Firebase Cloud Messaging

We take further advantage of the services offered by Firebase exploiting its messaging system: we'll use it to send notifications to the users when something meaningful happens.

3.2.4 Firebase Cloud Storage

Since we want users to have profile pictures, we use Firebase Cloud Storage to save their own images.

3.2.5 Firebase Authentication

In order to interact with the application, the user must be logged in. To support the login process, we use Firebase Authentication, both with e-mail and password or by using gmail authentication.

3.2.6 Firebase Test Lab

To test our app under stress conditions, we took advantage of the Firebase Test Lab service: this allowed us to test the app with a variety of devices, recording the whole test and producing a detailed output log file

3.2.7 chron-job

As we need some functions to be run every day, we took advantage of this service that enables to send an http request every given time interval.

3.3 External Dependencies

To facilitate the development process we take advantage of some existing libraries:

1. Faranjiit currency EditText: handle currency EditTexts
2. Glide: handle image loading

3.4 Multi Threading

One of our main concern regarding the application is that it has to be responsive: since its functionalities rely on data stored on the cloud, we need to handle internet requests on a thread different from the main one. Firebase SDK takes care of this by itself, running asynchronously each request on a different thread by default, and leaving us only to provide a listener that would run upon completion or failure of the request.
Another computationally intensive task that needs to be handled asynchronously is the image loading and rendering: the Glide library helps us to achieve this by integrating directly with Firebase Cloud Storage.

3.5 Device Sensors

Our app interacts with the device exploiting some sensors and listening to some broadcast events. To preserve battery life, we listen to changes in the battery level: when the level goes below a critical threshold, we disable live updates from the network. Moreover, in order to add a profile picture, the application has access to the camera and the gallery.

4 Application Flow

4.1 Navigation Patterns

In the following section we're going to illustrate all the possible navigation patterns a user can follow in our application. Then we'll focus on the navigation flows for the most relevant actions the user can perform

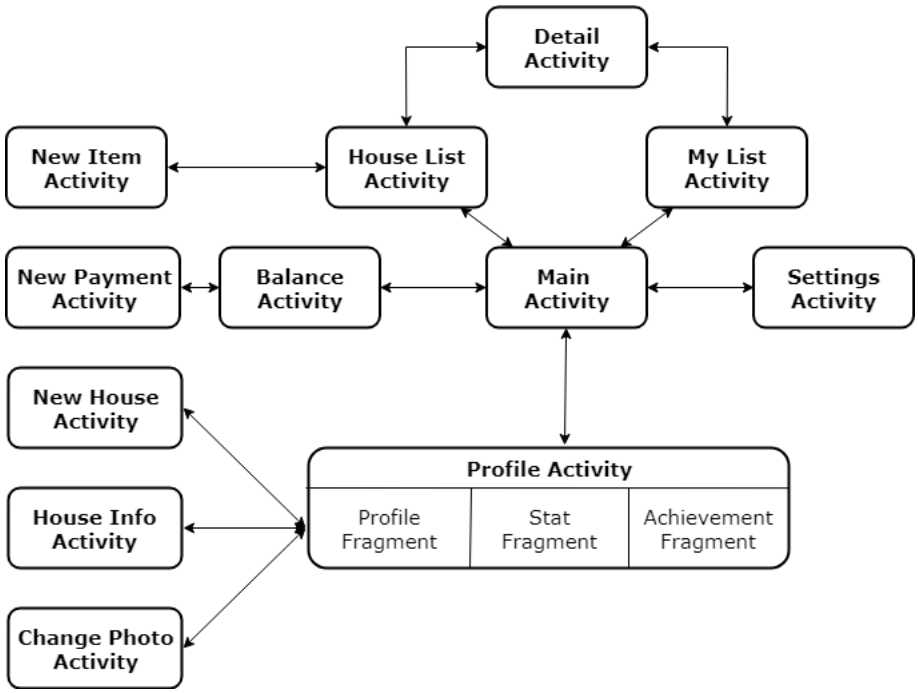


Figure 1: Complete navigation diagram

4.2 Create an item

Figure 2: Navigation flow to create an item

To create an item, the user should access the HouseList activity using the navbar in the main page. From here he could choose the house where the item has to be added, then has to click on the FAB to access another activity where he will be asked all the necessary info about the item. This transition will be highlighted by the FAB movement

4.3 Add a payment

Figure 3: Navigation flow to add a payment

To create a payment, the user should access the Balance activity using the navbar in the main page. From here, he must choose the house he wants to add the payment to and then click on the FAB to access the New Payment Activity. In this activity the user must fill a form with information about the payment. Finally, the user must click create payment to finalize the operation.

4.4 Take in charge an item

Figure 4: Navigation flow to take in charge an item

To take in charge an item, the user should access the HouseList activity using the navbar in the main page. From here he could directly take in charge an element in two ways:

- **Swipe**: performing a left swipe over the element will directly take it in charge. As the user might do it not on purpose, an explicit acknowledgment of his action is shown trough a Snackbar, offering as well the option to undo the action
- **Access the details**: clicking on an item will lead the user to its detail page: on smaller devices a visual transition will acknowledge the user on what's happening, while on larger devices a visual flag will help him remembering which item he has clicked. From here, the user can click on the accept button on the bottom of the fragment

4.5 Complete an item

Figure 5: Navigation flow to complete an item

The navigation flow to complete an item is the same as that to take in charge an item, the user should only access MyList instead of HouseList.

4.6 Change profile image

Figure 6: Navigation flow to change the profile picture

To change profile image, the user should access the Profile Info activity using the navbar in the main page. Once here, he must click on his own profile image, where the application will allow the user to choose between taking a photo or choosing one from the gallery.

4.7 Add a member to a house

Figure 7: Navigation flow to add a member to a house

To add a member to a house, the user should access the Profile Info activity using the navbar in the main page. From here, he must click on the house he wants to add a member to. This action will open the House Info activity, where the user can click on the "add member" button, which opens a dialog box. The user needs to insert the email of a registered user. If the user has entered a valid email, the application will add this member to the selected house.

4.8 Leave a house

Figure 8: Navigation flow to leave a house

To leave a house, the user should access the Profile Info activity using the navbar in the main page. From here, he must click on the house he wants to leave. This action will open the House Info activity, where the user can click on the "leave" button, which allows the user to leave the house. This action adds a payment to the balance of the selected house in order to reflect the change.

4.9 Create a house

Figure 9: Navigation flow to create a house

To create a house, the user should access the Profile Info activity using the navbar in the main page. From here, he must click on three dots to open a popup menu. The user must click on the "create a house" item, which opens a dialog box where the user can create a new house by entering all the required information (house name and email of all the members)

5 Activities

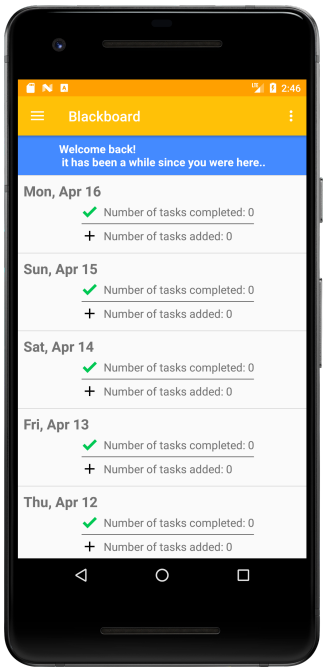
5.1 Main Activity

5.1.1 Description

Main Activity is shown upon each launch of the application: its purpose is to present a brief recap of the latest interactions the user had with the application, paginating the results weekly.

5.1.2 Components

Figure 10: Main Activity for small devices



A ConstraintLayout hosts a RecyclerView that presents a resume of the last 7 days; each item shows the number of items completed and added during the given day. User can navigate through the paginated weeks by clicking the extremes of the RecyclerView.
Navigation is provided by an hidden left navbar and can grant acces to any other activity of the app

Figure 11: Main Activity for larger devices

On larger devices, the RecyclerView will be sided by two graphs visualizing the same information

5.2 Balance Activity

5.2.1 Description

Balance Activity is in charge of showing the current balance of the user in each house. The activity also shows all the payments made by each roommate, divided in positive payments (those made by the user) and negative payments (those made by all other roommates). By pressing a button located at the top right of the screen, it is possible to access a menu, which can be used to select one of the houses the user is part of. Upon selection, the activity will show the balance of the user in the selected house. Finally, using a button located on the App Bar, the user is able to add a payment he has performed to the currently selected house.

5.2.2 Functionalities

- **F.7** enable users to access their balance information
- **F.9** enable users to delete a payment

5.2.3 Components

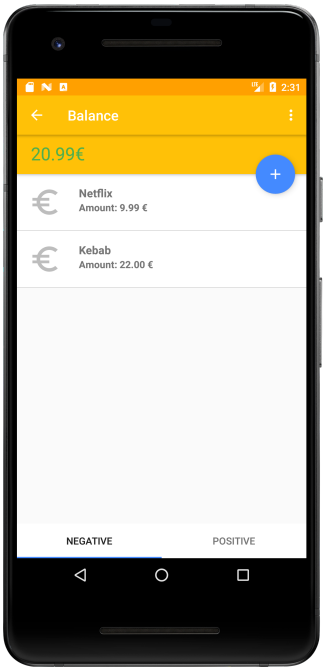


Figure 12: Balance Activity for small devices

The activity uses an AppBarLayout where the title of the AppBar shows the current balance. This AppBarLayout contains two tabs, one for the positive payments (the ones performed by the user) and one for the negative payments (the ones performed by other users). All the items in each tab are contained in a RecyclerView. If live synchronization is not enabled in the settings menu, the user can refresh the list of items by swiping down.

5.2.4 Goals achieved

- **G.2** through functionalities **F.7**, **F.9**

5.3 New Payment Activity

5.3.1 Description

New Payment Activity is shown each time the user presses the Floating Action Button in Balance Activity. By means of this activity, the user can add a new payment to the house that was previously selected.

5.3.2 Functionalities

- **F.8** enable users to add a payment

5.3.3 Components

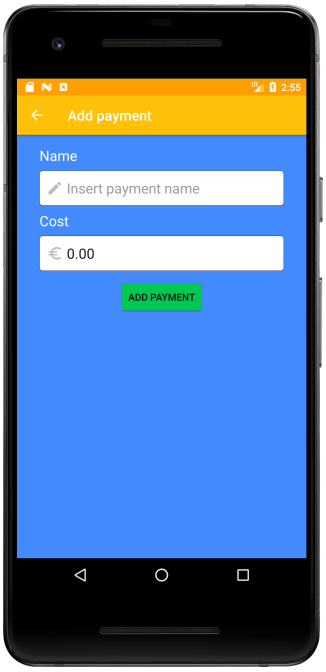


Figure 13: New Payment Activity for small devices

This activity contains a form the user must fill in order to create a new payment. In particular, the user is required to

1. Enter a name for the payment
2. Enter the amount of money he payed

Upon completion, the user must press the Add Payment button. If one of the fields hasn't been filled, the activity doesn't create a new payment and shows an error message.

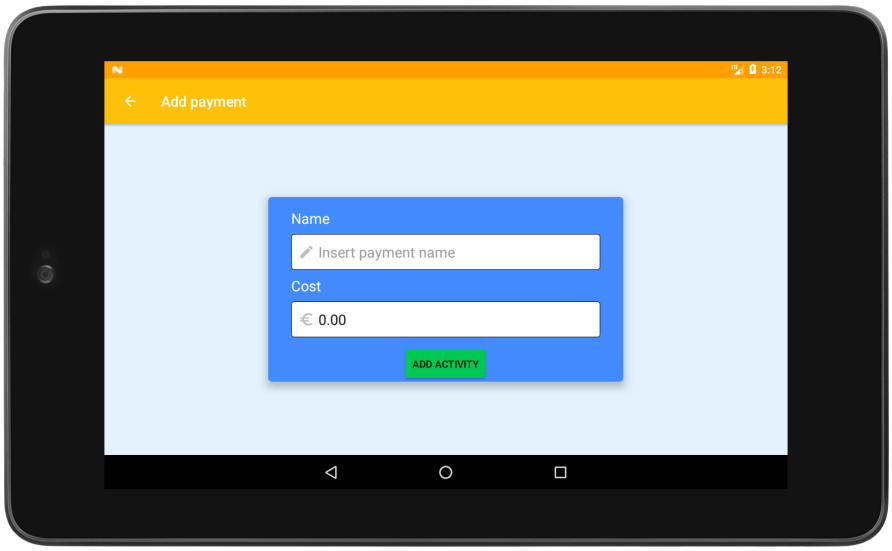


Figure 14: New Payment Activity for larger devices

On larger devices, the whole view is centered and surrounded by a box.

5.3.4 Goals achieved

- **G.2** through functionalities **F.8**

5.4 New To Do Task Activity

5.4.1 Description

New To Do Task Activity is shown each time the user presses the Floating Action Button in House List Activity. By means of this activity, the user can add a new task to the house that was previously selected.

5.4.2 Functionalities

- **F.2** enable users to add an item

5.4.3 Components

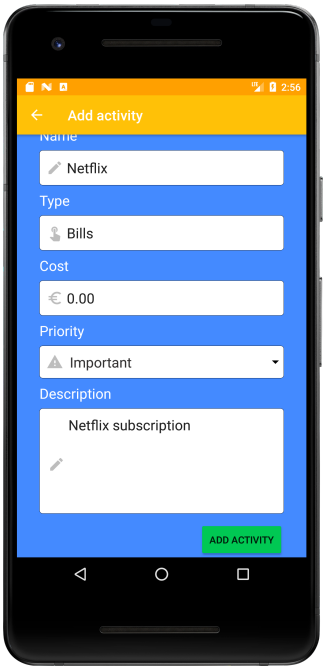


Figure 15: New To Do Task Activity for small devices

This activity contains a form the user must fill in order to create a new task. In particular, the user is required to:

1. Enter a name for the task
2. Select the type of the task (it is possible to choose from the following: Housekeeping, Billing or Shopping)
3. Select a priority level

4. Enter a description of the task

If the user selects Billing as the type of the task, an additional field requiring to enter the price of the bill appears. Upon completion, the user must press the Add Activity button. If one of the fields hasn't been filled, the activity doesn't create a new task and shows an error message.

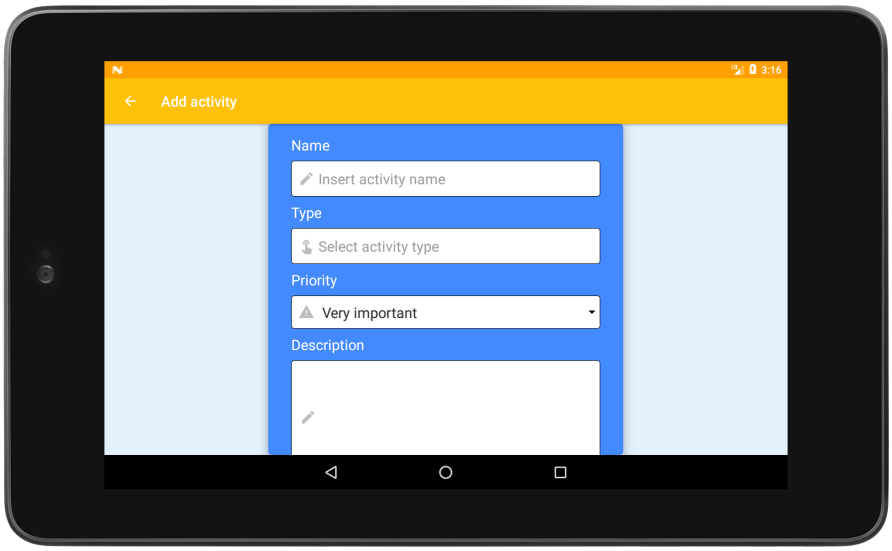


Figure 16: New Payment Activity for larger devices

On larger devices, the whole view is centered and surrounded by a box.

5.4.4 Goals achieved

- **G.1** through functionalities **F.2**

5.5 Settings Activity

5.5.1 Description

Settings Activity can be accessed by pressing the corresponding entry in Main Activity navigation bar. Its purpose is to set some values useful for the interaction between the application and the user.

5.5.2 Components

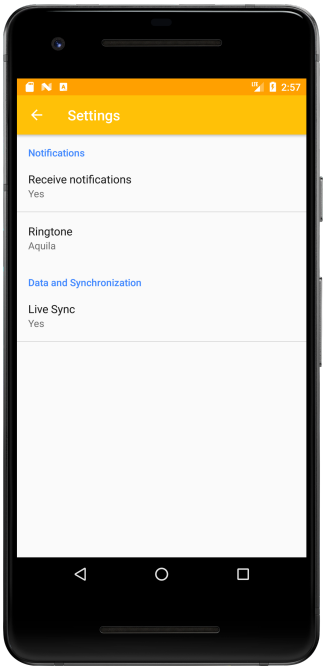


Figure 17: Settings Activity for small devices

The activity contains three settings values:

1. **Receive notifications:** allows the user to decide whether to receive notifications or not
2. **Ringtone:** allows the user to set the notification ringtone
3. **Live sync:** allows the user to decide whether to have live sync enabled or not

5.6 List Activity

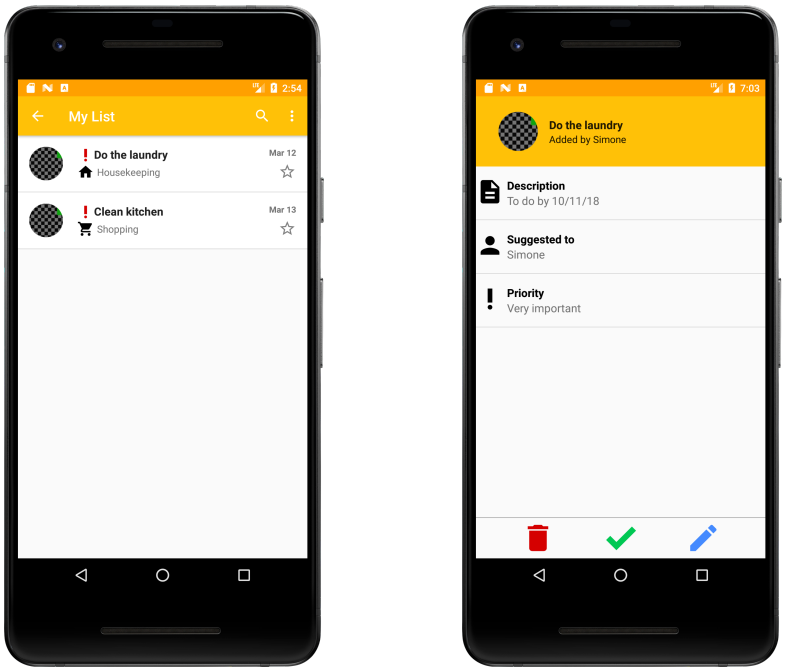
5.6.1 Description

This activity presents the user the list of items that had been proposed by the users of the Houses he has subscribed to. The items are divided by House: the user can access the list of houses by clicking on the top-left corner. The user can search for an item using a textual search that will be matched again the activity name or type. An item can be clicked, allowing the user to access its detailed information, or swiped, enabling the user to accept the item directly.

5.6.2 Functionalities

- **F.2** enable a user to add an item
- **F.3** enable a user to take in charge an item
- **F.4** enable a user to complete an item
- **F.5** enable a user to delete an item
- **F.6** enable a user to give up an item

5.6.3 Layout



(a) List fragment (b) Detail fragment

Figure 18: ListActivity for small devices

One fragment contains a RecyclerView, where items are listed: for each item is shown the picture of the person who added it and some short information about it. On the AppBar we have a SearchView to search trough the items and a button to choose the house to show. In the second fragment are contained the details of one item: on the top header the short info are presented, while the details are organized in a RecyclerView underneath it. On the bottom footer there are 3 buttons to allow the user to perform 3 actions. On smaller screens, the two fragments are contained in two separated activities.

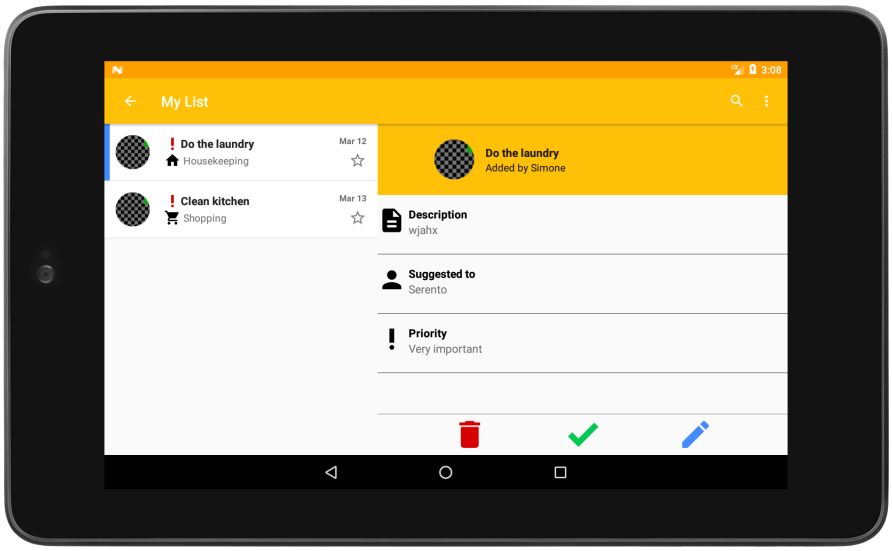


Figure 19: List Activity for larger devices

On larger screen, the list fragment is sided by the detail fragment, being both contained in the same activity.

5.6.4 Behaviour logic

Depending on the device size, the behaviour of the activity is slightly different:

- Small device: when clicking on an item, the user will be prompted to another activity presenting its details
- Large device: when clicking on an item, the detail fragment will be updated

With both layout, a left swipe will automatically take in charge/ complete the item

5.6.5 Goals achieved

- **G.1** through functionalities **F.2**, **F.3**, **F.4**, **F.5**, **F.6**

5.7 Profile Activity

5.7.1 Description

Profile info is the activity where the user can access all the houses he is part of and find some useful information about his own profile. The activity is divided in three sections:

- **Info:** contains a brief overview of the user profile info and a list of all the houses the user is part of
- **Stats:** contains a collection of statistics about the user usage of the application
- **Achievements:** contains a list of achievement earned by the user while using the application

5.7.2 Functionalities

- **F.1** enable users to create a House
- **F.10** enable users to access their personal info
- **F.11** enable users to change profile picture
- **F.12** enable users to access their list of houses
- **F.15** enable users to access some useful stats and achievements concerning their use of the application

5.7.3 Components

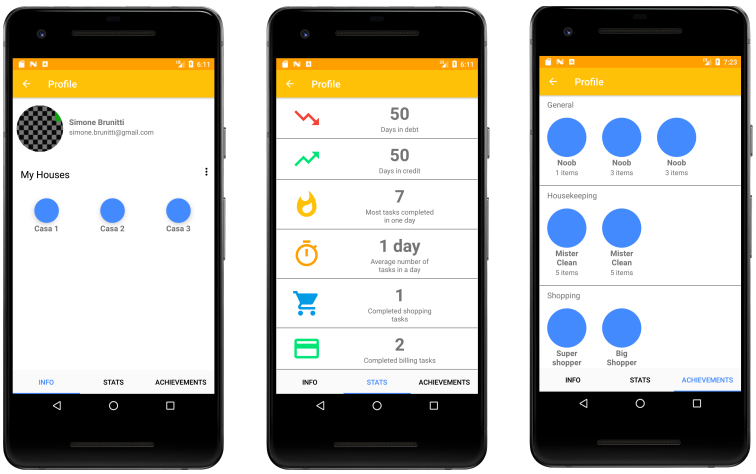


Figure 20: Info, Stats and Achievements for small devices

The whole activity is divided in three fragments, which are:

- **Info:** This fragment contains the profile image of the current user on the top of the screen and a RecyclerView of houses on the bottom of the screen. Both the profile picture and each house can be clicked. By clicking on the profile picture the user can change it, while by clicking on a house the user can access a brief overview of each one. Finally, the fragment contains an option menu that allows the creation of a new house.
- **Stats:** This fragment contains a list of statistics
- **Achievements:** this fragment contains a list of achievements

5.7.4 Goals achieved

- **G.1** through functionalities **F.1**, **F.12**
- **G.3** through functionalities **F.10**, **F.11**, **F.15**

5.8 House Info Activity

5.8.1 Description

House info activity allows the user to see a list of the roommates of a house. Moreover, the activity gives the user the possibility to add a member to the house or leave the house.

5.8.2 Functionalities

- **F.13** enable users to add a member to a house
- **F.14** enable users to leave a house

5.8.3 Components

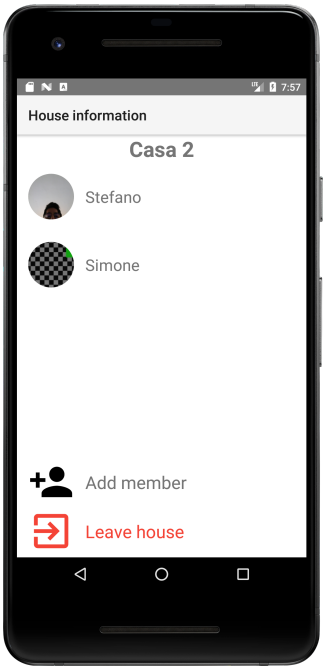


Figure 21: House Info Activity for small devices

This activity contains a list of all the members of the house on the top part of the screen. On the bottom of the screen there are two buttons, respectively for adding a member and leaving the house. If a member of the house leaves it, the balance of the house will be updated in order to reflect this change (the member of the house will then need to give money to other members if he was in debt or receive if he had a credit).

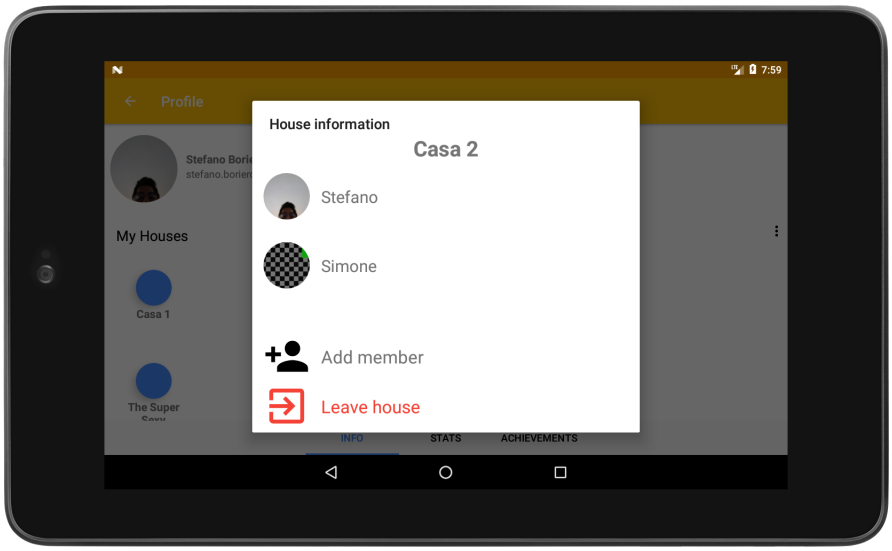


Figure 22: House Info Activity for larger devices

On larger devices, the view becomes a dialog box.

5.8.4 Goals achieved

- **G.3** through functionalities **F.13**, **F.14**

5.9 Login Activity

5.9.1 Description

Login activity is the first screen the user is directed to when he opens the app. On this screen, the user can login if he is already registered(either using e-mail and password or by using gmail). If the user is not already registered, he can do so by clicking on the link on the bottom part of the screen.

5.9.2 Components

Figure 23: Login Activity for small devices

The activity contains two EditText for entering e-mail and password, together with two buttons and one link. The two buttons are respectively for signing in with email and password or with Gmail. The link, on the other hand, redirects the user to the sign up activity.

5.10 Sign up Activity

5.10.1 Description

Sign up activity is used when the user wants to register using username and password.

5.10.2 Functionalities

- **F.16** enable users to register a profile

5.10.3 Components

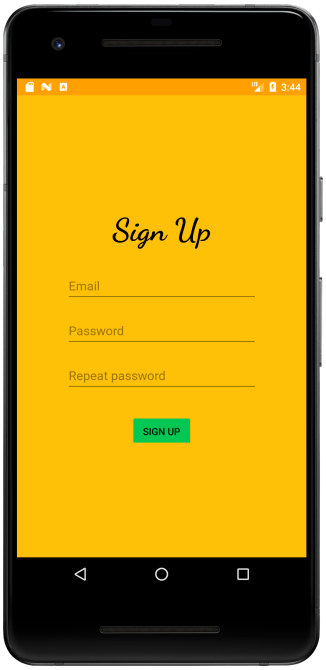


Figure 24: Sign up Activity for small devices

The activity contains three EditText, all used for user registration. The user is prompted to enter the password twice, in order to be sure he didn't make errors while entering it. Once the user has filled the form, he can hit the submit button to register.

5.10.4 Goals achieved

- **G.3** through functionalities **F.16**

6 Database

6.1 Description

The application data is hosted remotely thanks to the Firebase Firestore service. It required to structure the data according to a NoSQL data model: we therefore had to identify the collection and document structure that would best fit our needs.

6.2 Data Structure

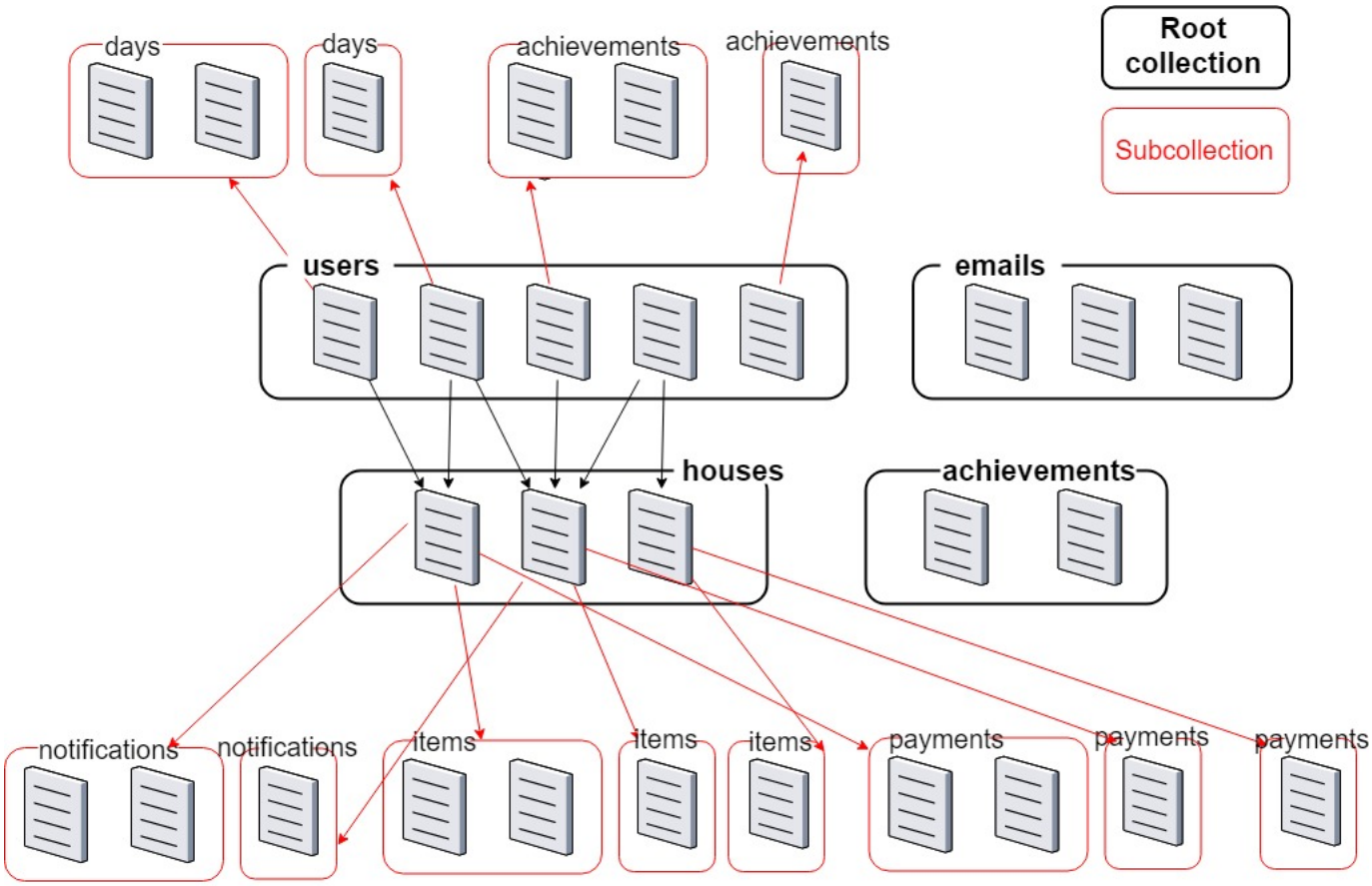


Figure 25: Data structure

We identified 4 root collections:

1. **houses**: collection of documents representing an house. Each document has then 3 subcollections:

- items: collection of documents representing an item
- payments: collection of documents representing payments
- notifications: collection of documents representing the notification info of each user

2. **users**: collection of documents representing a user. Each document has two subcollections:
   - achievements: collection of unlocked achievements
   - day: collection of summurized information about one day of interaction
3. **email**: collection of information about emails
4. **achievements**: collection of possible achievements to unlock

6.3 Data Integrity

Since the data model is of a NoSQL nature, Firebase doesn't actually provide a robust infrastructure to support the integrity of the data: a common workaround for this problem is to implement several Cloud Functions, that can be triggered by events in the database.

7 References

Here can be found some useful links for the external services or libraries that had beeen used in the development process

- **Firebase**: `https://firebase.google.com/`
- **Glide**: `https://github.com/bumptech/glide`
- **Currency EditText**: `https://github.com/faranjit/currency_edittext`
- **Cron Job**: `https://cron-job.org/en/`

8 Future Works

At the end of our work, we tried to identify which could be some improvements from which our users could benefit

- **Modify Item:** we could let the user modify an item after its insertion on a list
- **Enable/disable notifications:** in order to improve the control over the functionalities, we could let the user decide whether or not to receive notifications from the application