A3-Android Prototype

Developing Mobile Applications (ID2216)

Marvin Koselnik, Sylvain Roudiere, Corty Suss, Shi Yu, Matthias Probst

December 6, 2018

1 Web App Prototype

The implemented Android Application has a similar AI as the previous Web App Prototype, which can be seen in figure 1.

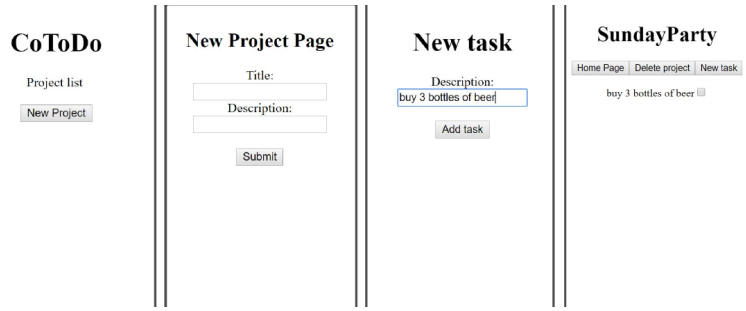

Figure 1: AI of the Web App Prototype

The user test with the Web App Prototype showed following things to improve:

- Our design missing pages as a complete use flow, including: friends list page, my tasks list, empty task list (compare to non-empty task list page, empty task list page need graphical to guide users)
- Our design missing next/confirm button for some pages
- We could also add a feature to suggest "common task lists"
- Our App should not ask user to add member twice (at the beginning when they are creating a project and when they assigning tasks)

This and the results from the user test lead to developing the Android native app, as described in the next section.

2 Android App

We decided to use Android studio in java to develop the app because the graphical user interface of Android Studio when creating new screen is not so complicated to understand and because Android studio allows us to implement a Model-View-Controller architecture. We also chose this tool because we could use Java to develop the app.

2.1 Model-View-Controller Concept

We have chosen the Model-View-Controller Concept for implementing the App. We chose it because it seems to be a good way to separate the three instances and to have a modular work. The principle of the concept is shown in figure 2.

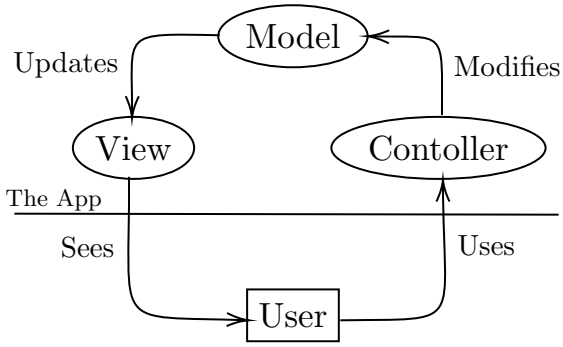

Figure 2: Model-View-Controller Concept

The principle of this architecture is simple: the User sees the View. On the view there are several possible actions (often buttons). The user can use and activate those actions. For example, by pressing a button, the Controller will then be called to act accordingly to the action that has been made. Then, the controller will send the user request to the model which will process it. The model contains all the data and procedures. The model will then update the view. For instance, if the user wants to create a new task, he will click on a button "Create a New task" that he sees in the View, then, by clicking on it, will call a Controller's function which will send the message to the model, which will load the "New task screen".

2.2 Model Class Structure

As already mentioned there exists an underlaying model, which holds the data and manages it. This is shown in figure 6. From the figure the function and hierarchy is quite clear. In order to let it work properly and also fulfill the most important pinpoints of section 1 we implemented according methods for each class. More concretely speaking, the model classes are the entities we want to manipulate in our app which are: Project, Persons, Tasks. That's why we have one class for each entity and several functions for each class such as addProject, addTask etc... The model is implemented in Java.

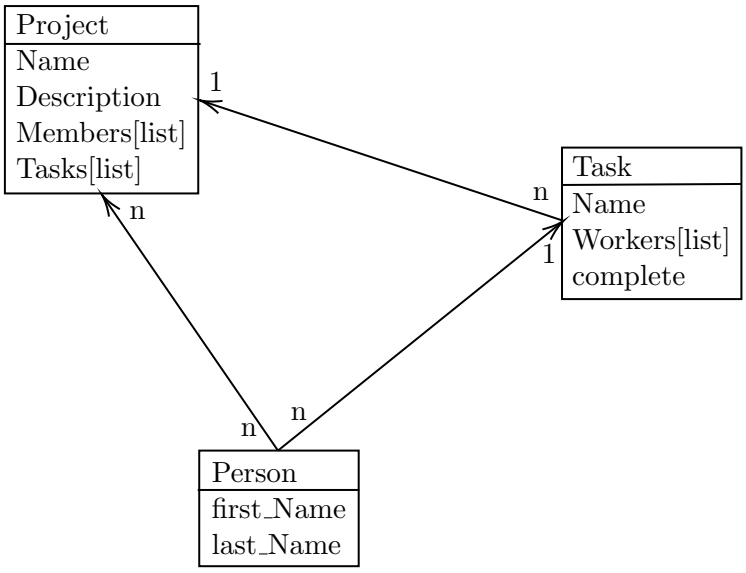

Figure 3: Class structure of the model

2.3 Implementation of the Controllers

The controllers are also implemented in Java. Every different screen has its own controller-class also called activity class in Android Studio. Thus there exist as many different controllers as different screen GUIs. When a new project is created, there gets a new object of every controller initiated. More concretely speaking, there is one main controller that creates several sub controllers, which can create several sub controllers etc... In our app, the activity-main launches the add-project-activity and also the project-activity which is the project's main page. Those objects are specific to each project. All this is a logical way to ensure, that only the chosen project is modified and not all others because every project generates its own controllers.

2.4 Programming the GUI

The graphical user interface (GUI) one will later see is programmed in XML. It represents the view-part from figure 2. In Android-studio the GUI can be done visually, but dynamic modifications such as adding a task to a project needs to be done in Java. So, basically, the View contains the objects that appears on the screen, and by adding in Java (in the model), the elements to the view's objects, it is shown on the screen.
Each button is connected to a controller that does specific actions.

3 Result and User Test

In general, we managed to have a better looking application, and implemented the features about adding people and tasks, and assigning people to tasks. What lacks our application is the possibility to remove projects, tasks or people.
We had difficulties creating a fragmentList to refresh the project lists after adding one, so we chose to add a refresh button to refresh the screen. This is the main screen of the app and the screen to add a new project (after clicking on the green plus):

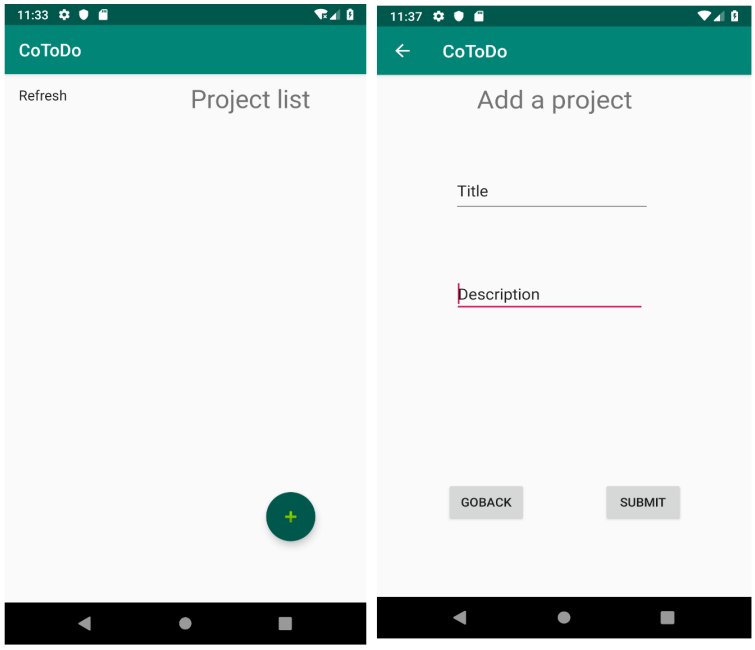

Figure 4: Welcome and add project screens

We chose to implement first all the features but removing features because we wanted to have all of them tested by users, because the removing action is not the first thing a user will do on the app.

We tried to make it as modular and ergonomic as possible. For example for the task manager screen, one needs just to click on the name of the participant or on the name of the non participant to change its status and the screen is automatically updated. The persons are all either in the participants list or the non participants lists for every project and for every task, so one doesn't need to add himself twice.
We had our app tested by several people and the overall negative feed back we had is the fact that it is not possible to remove anything, and that adding new people to projects could be more ergonomic also. The positive feed back we had is that our app can be intuitive once we know what's its use. (otherwise the user is a bit lost).

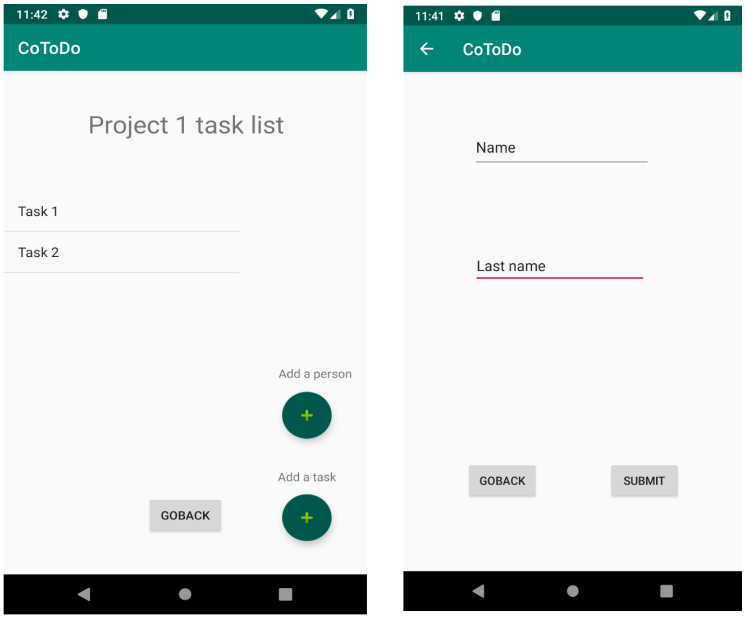

Figure 5: Task list and add people screens

So for the future native app, what we can do to better this app is to add the removing features, try to correct and remove the refresh button in the first screen and try to make the person adding easier and quicker.

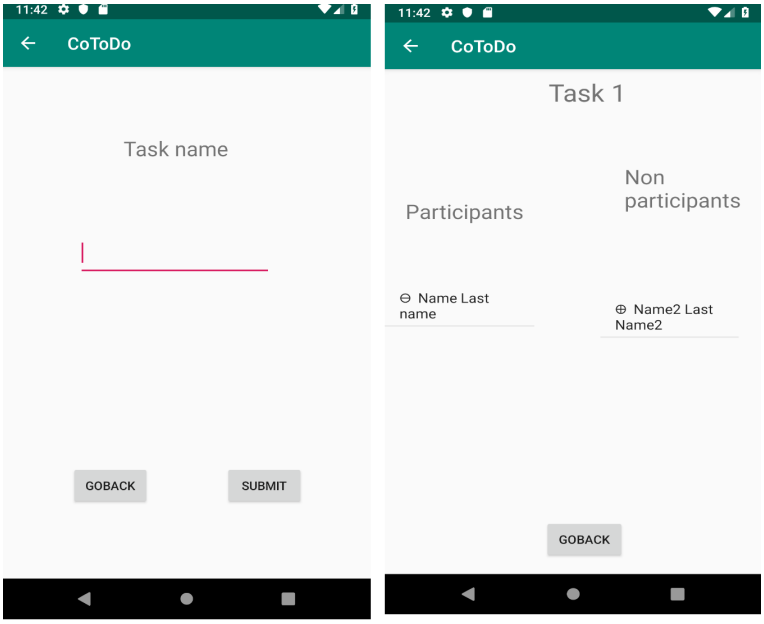

Figure 6: Task manager and add task screen