Developer Documentation

Determinator

Getting started

In order to get started, some prerequisites are required. These are amongst others:

- Java SE Development Kit 7
- Android SDK, version 22
- A (virtual) Android phone
- Android Studio

And if you want to use your own server:

- PHP and MySQL server for the API, and the instructions to install this is found in separate repository, see API further down.

To get the code for the project, use git:

```
git clone https://github.com/PINOMG/determinator.git
```

After the code is copied to the local filesystem, simply open the project in Android Studio and you are good to go. If you use your own server, remember to switch the *BASE_URL* variable in ApiHandler.java to match your server's URL. Right now the application uses the PINOMG organization's server at http://79.99.3.112/pinomg/.

Design decisions

The minimum SDK level is 21, and the target SDK is 22. The minimum was chosen to focus the development on the functionality the application requires, instead of laying time on backwards compatibility. Another reason was that all team members' phones had an SDK level of 21 or higher.

The Volley framework was found late in the development process, and had several benefits against the previous way of handling HTTP calls. A future vision is to implement Volley for all future API calls, and refactor the current api related code to use this framework. More about this under chapter API.

Building and release

This project uses Gradle to build the application. We recommend that you use Android Studio and the built in Gradle functionality. Building can be done by running (in the root directory of the app):

```
$ ./gradlew assemble
```

Unit tests

The text based tests available are found in docs/tests.pdf.

The automated unit tests are located in app/src/test directory. To run the unit tests just run the test task (in the root directory of the app):

```
$ ./gradlew test --continue
```

For more info on how the unit tests are set up, see:
http://tools.android.com/tech-docs/unit-testing-support

Architecture

The system contains of several packages which are grouped as follows: activities, database, helpers, models and net. The classes inside a package can be dependent upon each other, but also of a class of another package. For an overview of the dependencies, see Figure 1.

The database package is not used at the moment, but will be a necessary part of the application to use for caching, and was also used during the development of the API (determinator_server).

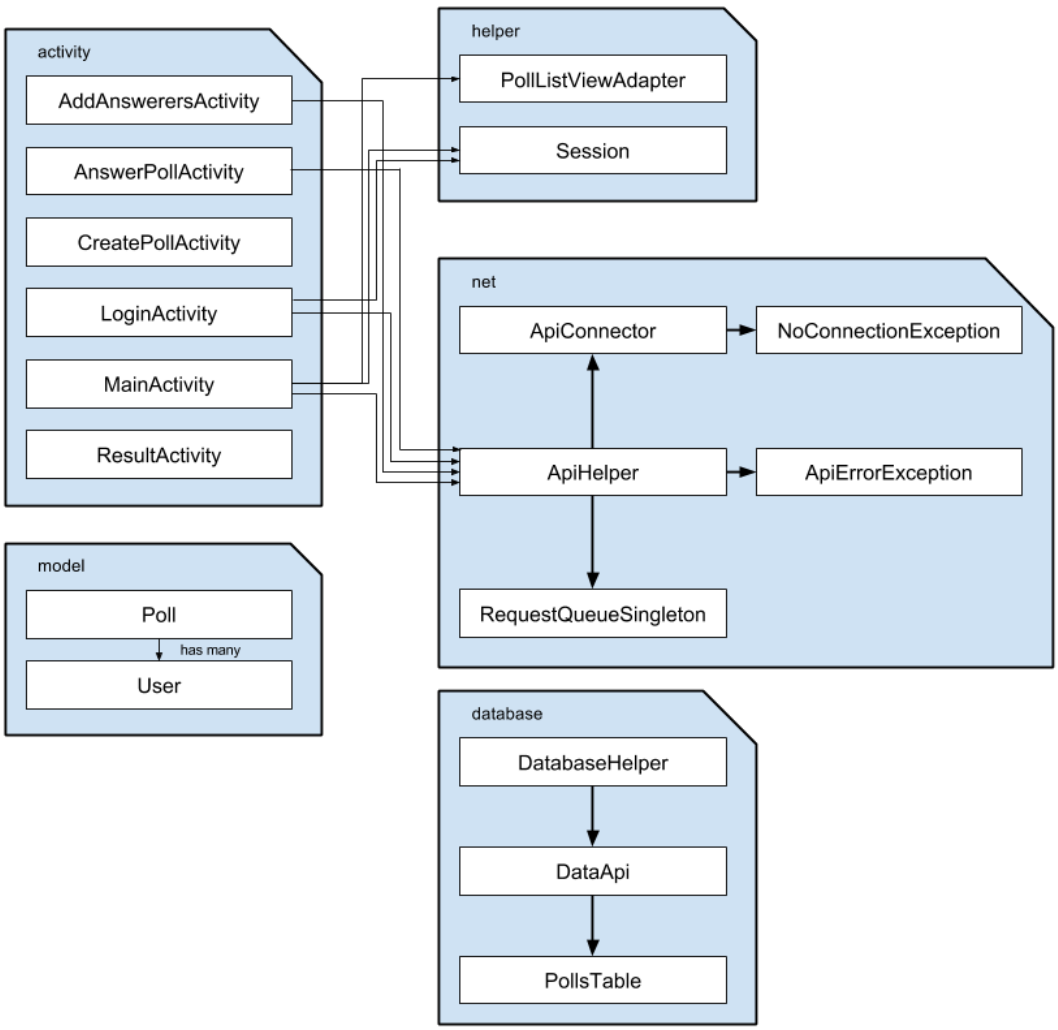

Figure 1. An overview of the packages and classes. Not all dependencies are shown, only the most important.

Flowchart

Figure 2 shows how the user navigates between the activities in the application. The application starts to check if the user is logged in. If so, the user will be redirected to MainActivity, where all the main functionality is available. If the user is not logged in, the user will be redirected to LoginActivity, where the user can authenticate.

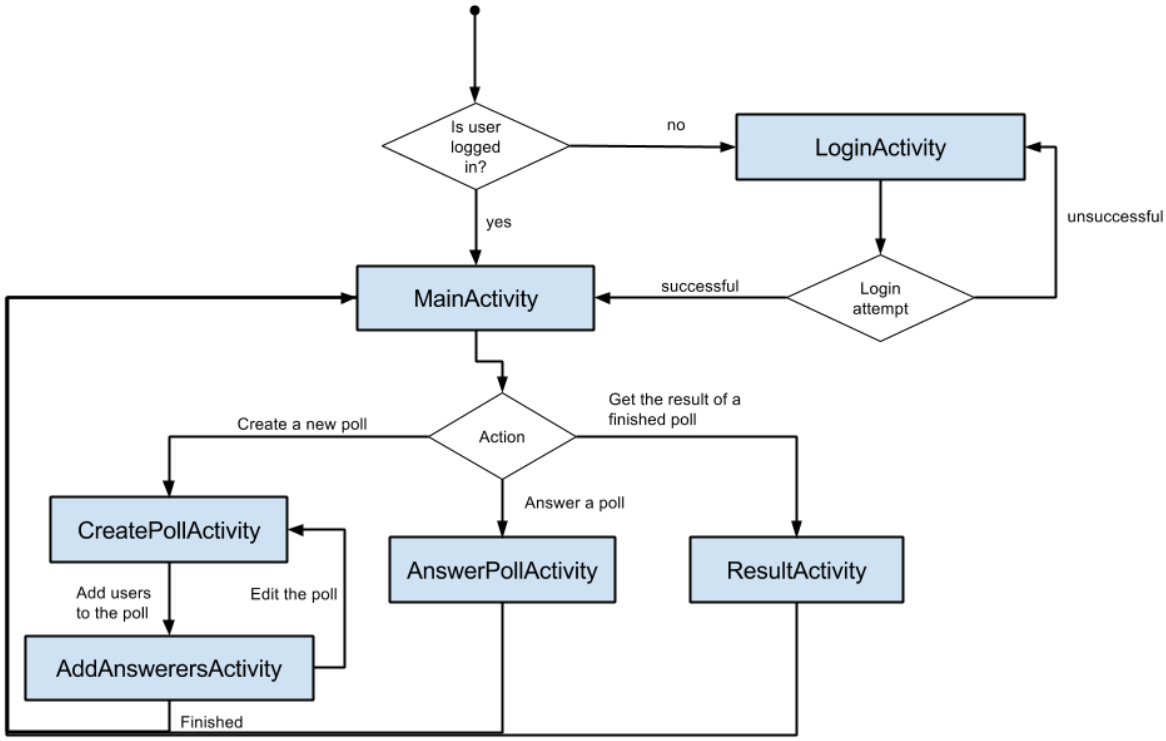

Figure 2. The relationship between the activities in the application.

Database

This is the application's database schema, which contains only one table, polls. Note that the database is not in use at the moment, but will be used as caching in the future, and it was also used earlier in the development process.

```
polls (
        id                  INTEGER
        question            TEXT
        alternative_one     TEXT
        alternative_two     TEXT
)
```

API

The application uses the Determinator server REST API to handle creation of polls, answer polls, get poll results and user authentication. The API is found at `https://github.com/PINOMG/determinator_server.` The documentation for the API and how to use it is found at this page.

The application parses the JSON response from the server, and handles error responses sent from the server.

The application uses two different methods to communicate with the server. The first method is via the Volley framework, which sends asynchronous calls. This method is used to get arrays of data, which is only used when getting the list of polls belonging to the user. By using the asynchronous method the application can be used during the api call and this method is essential to use when fetching large amounts of data. This is the preferred method to use for all api calls, to give a better user experience.

The second method is sending http calls via the Java net HttpUrlConnection. Although the http task itself is asynchronous, the implementation of the ApiHandler makes it run on the main thread. The main thread will therefore wait for the call to finish until continuing. This method is used for all API connections except when fetching all polls.