# Developer manual for the Odin app

## Dependencies

- Google Maps
- Slim
- Android SDK target level 17
- Java JDK 7

## Installation and building process

1. Se till att Android Studio är installerat
2. Klona Git-reposotory:n från https://github.com/fonfalleh/Odin.git
3. Se till att Android SDK (API level 17), en (virtuell) Android enhet och Java Development Kit 7 utvecklingsmiljö är installerat.

## Design decisions

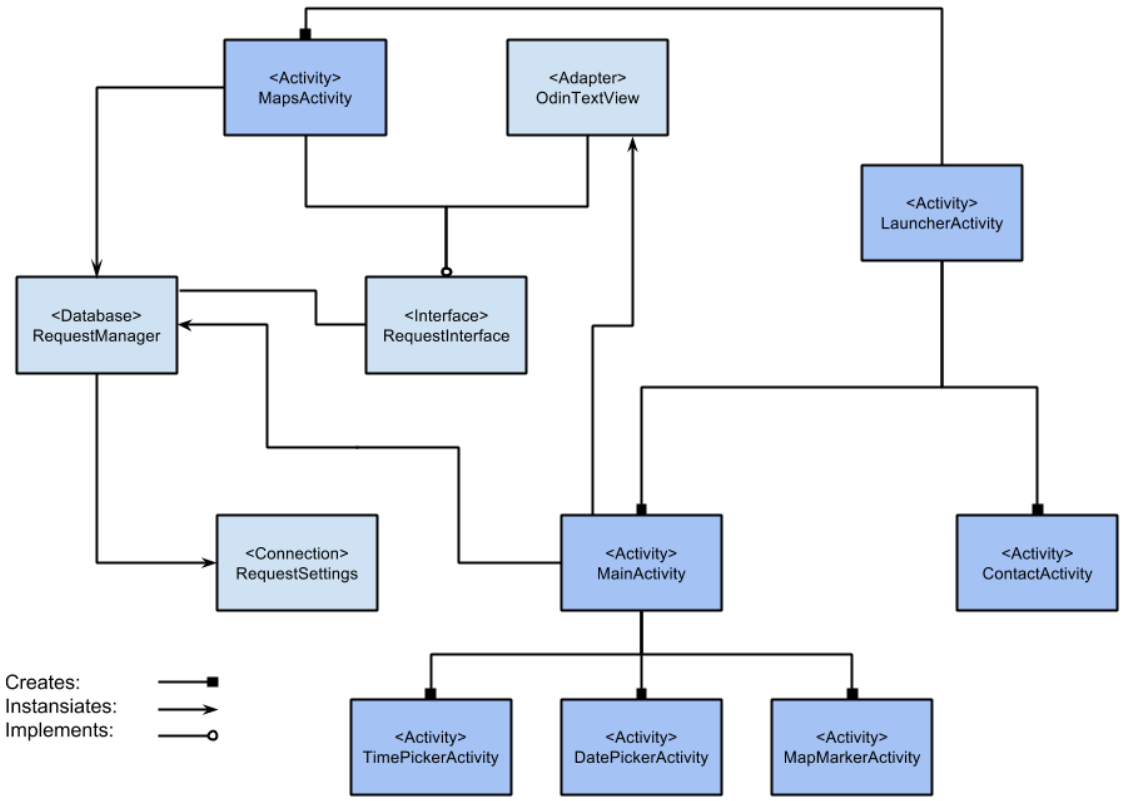
In order to make the application available to a sufficient number of devices, without having to use deprecated functions and classes, the Android API version 17 has been used. Another reason for choosing API version 17 is that it increased our possibilities to test the application on actual an actual device and not do all testing in a virtual emulator.

Since one of the applications main features is to view geographic locations on a map, Google Maps was chosen as an external service because of a number of reasons. First and foremost it is an application that is preinstalled in most android devices and it is relatively simple to connect to an android application. Apart from Google maps the application is not dependant on any other external services or libraries other than the Android API version 17.

On the server side PHP has been used to handle Requests and database integration. MySQL is used for the database and Apache is used for web services. The main reason for these choices is the simplicity of the LAMP installation on the server. There has also been some preferences within the group in regard to PHP and MySQL.

To handle HTTP requests a micro framework in PHP called Slim has been used. This choice was made in order to decrease the time spent on development on the server.

# Major components and UML



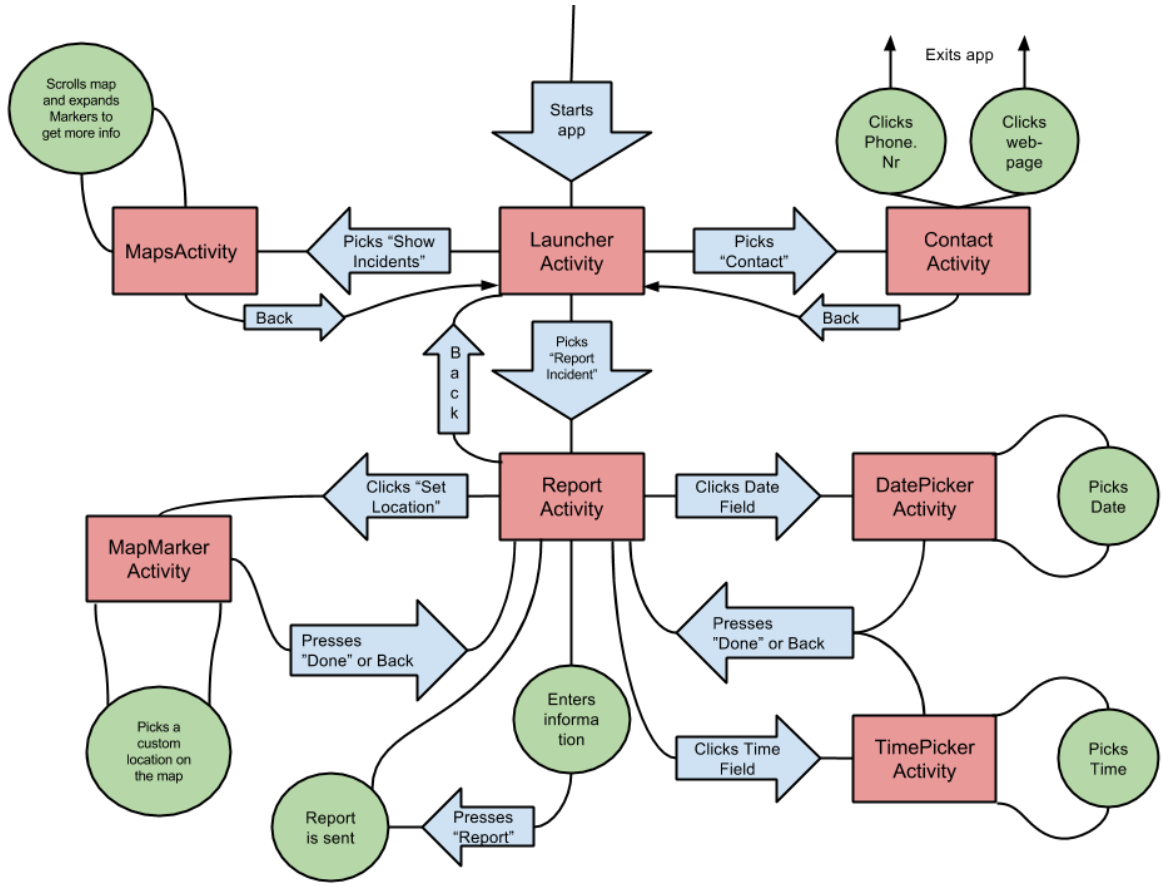

The app is structured around a number of Activities, all of which extend the Android Superclass *Activity* in some way, mostly by extending *AppCompatActivity*. These activities create and send data to each other in a structured manner using *Intents*.

All communication with the database is handled by the *RequestManager*, which is used both by *MainActivity*, to send a POST-request with the entered report information to be stored in the database, and by *MapsActivity*, to send a GET-request to get all of the reported incidents from the database to visualize on the map. *RequestManager* also uses the class *RequestSettings* to handle the details of the URL-connection to the database.

*RequestManager* also takes an implementation of the interface *RequestInterface*, which is implemented both by *MapsActivity*, to process the recieved data, and by the class *OdinTextView*, which connects the *RequestManager* directly to a *TextView* in *MainActivity* which it uses to inform the user of report failure or success.

## User experience Flowchart

The following chart illustrates the users navigation through the app's different activities and features. A red box represents an activity screen, a blue arrow represents a user action or decision, and a green balloon represents a user activity.




## Protocol (Björn)

A microframework for PHP called Slim is being used for easier management of HTTP Requests on the server. The client can send three different types of standard HTTP Requests to the server.

| URL | Parameter | Type |
|---|---|---|
| http://188.166.95.224/incident_manager/v1/incidents | (none) | GET |
| http://188.166.95.224/incident_manager/v1/incidents/**:id** | (none) | DELETE |
| http://188.166.95.224/incident_manager/v1/incidents | name (string)<br>incident (string)<br>lat (float) | POST |

| | | |
|---|---|---|
| | long (float)<br>timestamp (string) | |

The GET-request is used to get all the incidents from the server along with geographic positions. The POST-request is used for creating an incident with the data in the parameters. The client does not use the DELETE-request in the current version but the functionality is there to delete an incident. To delete an incident the client must replace the **:id**-tag with an integer that represents the id of the incident.

Every request returns a multidimensional associative JSON-array that includes a boolean value named “error” that tells the client if an error has occurred and an error message that tells the client what has happened in case of error.

## Known bugs

- The server architecture currently does not fully support all non-standard latin alphabet characters, and usage of characters such as “ÅÄÖ” may therefore cause unpredictable string representation in the app.
- The GPS uses fairly old location information, and might not always reflect the user’s current location very accurately.

## Possible future work

- As previously mentioned, some bugs still exist with the app and its accompanying database. Correcting these should be given the highest priority.
- To completely fulfil the vision for this project, it may be of value to create a more automated and simple approach to export and share all collected information in the database with chosen organizations.
- The user interface is currently based entirely on Android Studio’s basic XML-editor parts, and use no custom parts, images or textures. As a result the look of the app is very basic and not very appealing. During the development process, this was considered unimportant considering what the vision for the app was, and that the functionality of the app has not yet been fully specified. Development time was therefore instead spent on developing the functionality of the app. A future update of the user interface to make it more clear and visually appealing could however contribute to the app’s usability and spread.
- Currently, the contact functionality of the app only refers the user to the Swedish help organization BRIS (Barnens rätt i samhället). This functionality should of course be extended to enable easy contact with more organizations. Once the number of organizations is so large that looking through them all in a glance is no longer trivial, one could also imagine a system that lets the user sort the organizations along different dimensions, or does so automatically according to identified user location or details of their sent report.
- A point of discussion that surfaced at several points during the project was whether or not the incident-mapping-functionality of the app should display the entered incident-report text.

- A “Heat Map” that visualizes which areas that have a higher frequency of incidents was one feature that was scrapped during development due to a high work-load. It is still considered to be an useful and attractive feature that could be added in the future.