Developer documentation

How does the build process work?

Our application use one necessary build process which is Gradle. Gradle is all one need to install and build our application.

System design:

Overview: The application uses an android MVP structure, “package-by-feature” structure and services classes for communication with a database.

External dependencies

- We use the Google+ API. Google+ allows us to have an easy login system, where the user just needs to choose their Google account and sign in.
- The database service Parse has its own SDK.

MVP structure combined with package by feature:

We use package by feature in our project. It’s a package structure which put classes relevant for one feature in the same package. For example; all classes which are relevant to the buslogging component is located in the same package. The classes within these feature-packages are structured by more packages following the Model-View-Presenter design pattern. The view classes are plain .XML files while the presenters and models are java classes. In some feature-packages there are also other packages like service or utils to further divide the classes within the feature-package, (shown in figure 1).

The way the application was built was to have every package to be able to function as independent as possible. This way the packages become highly modular. Some dependencies are necessary between the packages, but by using interfaces as gateways between them we have made it more modular and less coupled. The application is open for extensions, but closed for modification.

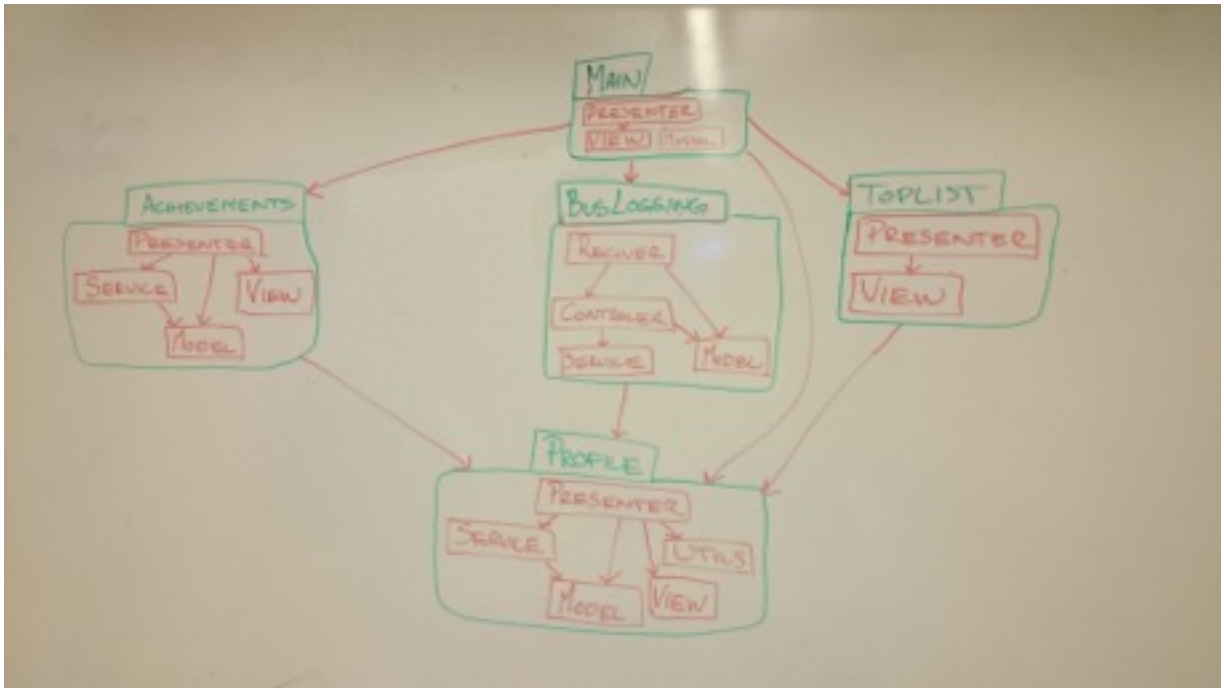
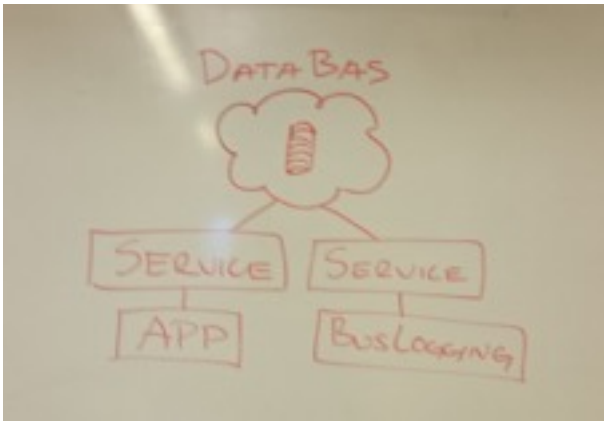


Figure 1 (high level package structure).

Services for database

To retrieve and write data to Parse, our database, we need to use the Parse SDK.

To communicate with the database without compromising the sustainability or exchangeability of the program, we have made Service classes. These classes handles all communication with the database, and therefore all Parse-code is contained within them. These classes also have their own interfaces which cover the base functionality of the classes. To change "storage type" from our database Parse to some other database or local storage, (almost) all one needs to do is change or replace the Service class and implement the service interface. This simplifies the usage of the subsystem Parse. This also makes the program almost independent from parse API, since it is only the interface of the Services that is used in the rest of the application.



What major parts / components are there in the application?

Main

Contains the login class that handles the functionality from the Google+ API to improve and simplify the login process. It also contains the classes that setups the TabActivity which is the main layout class. TabActivity handles the swipe and tab menu. By creating different fragments, TabActivity (using viewPagerAdapter and myViewPager) easily creates new tabs for the application. The main package also consist of a search function which is represented by a model.

BussLogging

This system is a background process, and is responsible for logging bus trips whenever the phone is in range of one of the WiFi-connections available on the Electricity busses. The module only consists of model, control and service classes because it doesn't show any graphical design. It functions by having a broadcastreciever which listens to an alarmanager on a one minute interval. The receiver starts identifytravelservice when appropriate. The server communicates to the models to calculate data. When the service is done, it gets data from the models and sends it to the database through the service package.

Profile

Contains all classes and methods relevant for a user's profile. The Profile package has its own service package responsible for communication with our database, where all the profiles are accessible. The (presenter) ProfileFragment in the profile package handles the user front page. In the model package the profile model represents one user. ProfileHolder holds all the users in a singelton. The user class is a sublcass to profile and represents the user currently logged in. The user is globaly reachable through a singelton called Userhandler.

Toplist

The toplist modules consists of presenters which assignment is to present profiles. Each one of the profiles has its assigned information from the ToplistAdapter. The ToplistAdapter matches each profile with the corresponding information from the Profile package. From the Toplist you can also reach a more informative user profile, where detailed information are presented by a Profile fragment. The adapter sets an onClickListener to each of the objects in the Toplist, and when clicked upon, a new fragment is presented. The new fragment is presented by the Toplistfragment.

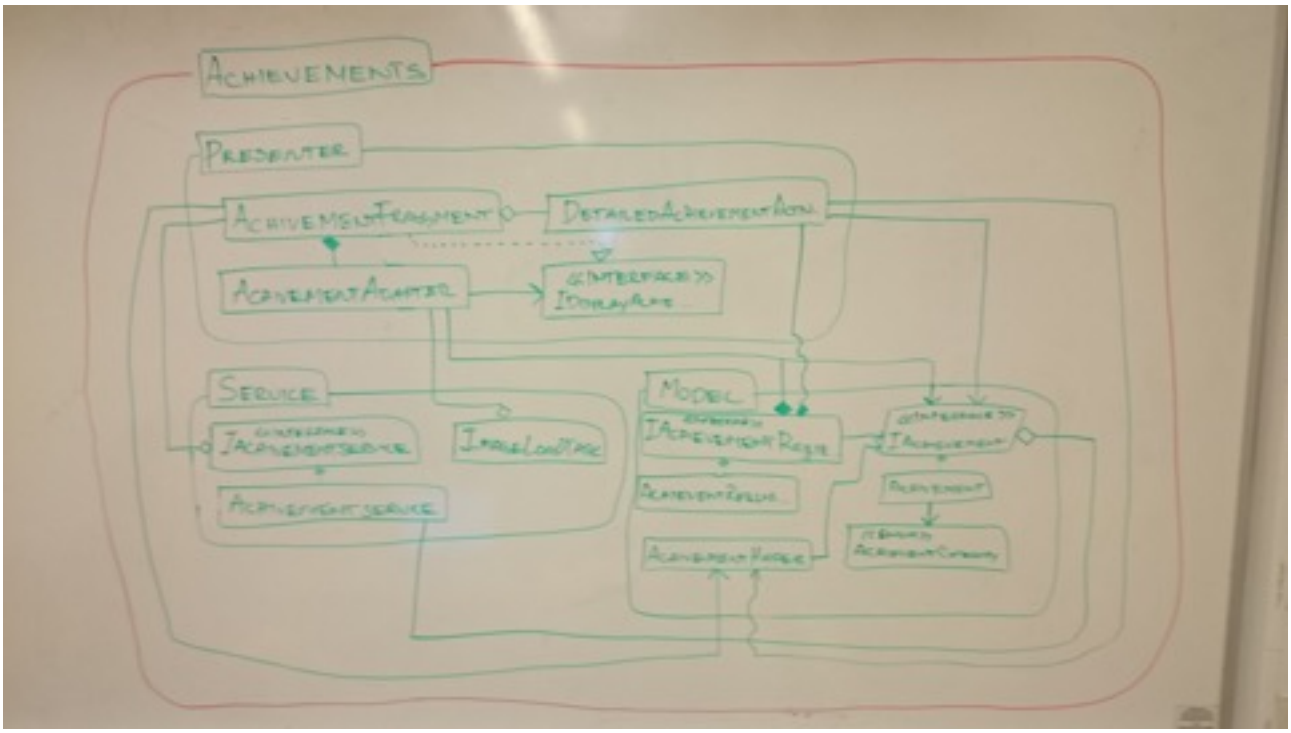
Achievement

This package displays a user's travel data in form of progress. By fetching the users travelinfo from the database, AchievementRequirements evaluates if a user have completed a specific task(there is two different requirements, distance or amount of travels). All the achievement requirements and information about the achievement is located in the database so its easy to add new achievements. The AchievementFragment then displays the progress in form of graphical components for the user.

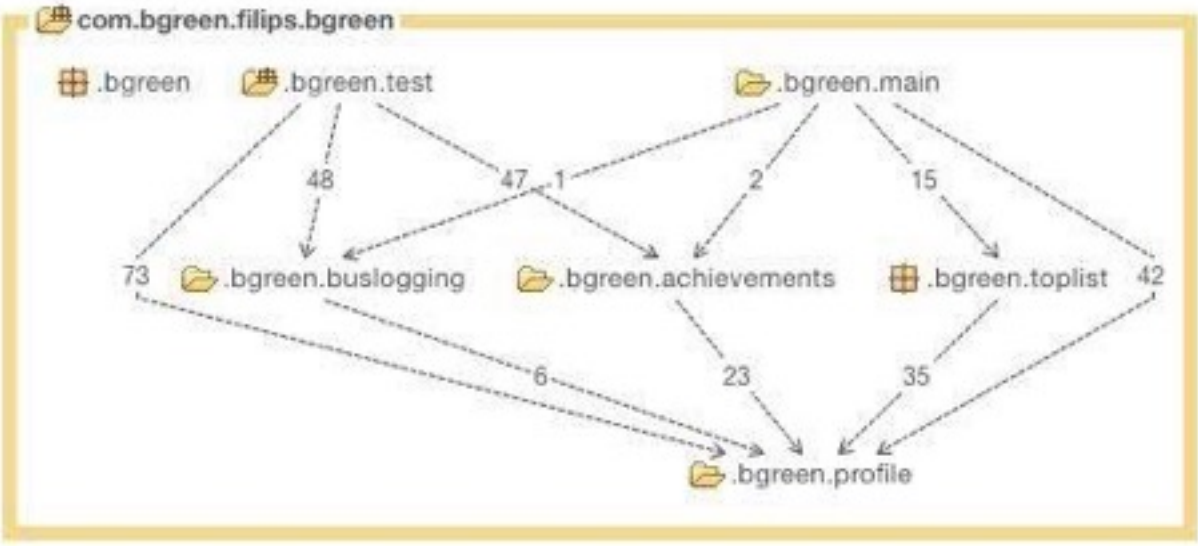
Design decisions (such as API level, etc.)

The minimum SDK is currently 17. In the beginning we set the minimun SDK version to 15, so that people who uses ice cream sandwich 4.0.3 had the ability to use the application. Later in the implementation phase a problem accured that we couldn't find a good solution for so we decided to set the minimun SDK version to 17 where a method that solved the problem was available. 17 is still is quite low, so it offers a lot of users the possibility to use the application.

Appendix



UML of the Achievement package



Dependence Analysis