Overview

Software decomposition

Eventify is decomposed into the following package structure. Each line indicates a dependency. The goal has been to maintain a clean code while facilitating debugging and further development. There is a minor circular dependency between the eventutil and activity package since the class NotificationUtil in eventutil references DetailAcitivty in the activity package in order to open a detail view based on the notification clicked. This dependency has been ignored due to its minor nature but should be looked at if further development is made.

- **Activity,** contains all the activity classes
- **Manager,** contains all the classes close to the Android system
- **Model,** contains all the model classes
- **Helper,** contains smaller classes used to help others
- **Eventutil,** contains classes that provide functionality regarding events
- **Util,** contains classes that provides functionality. E.g. location services
- **View,** contains all the UI classes
    - **Adapter,** contains all the adapter classes

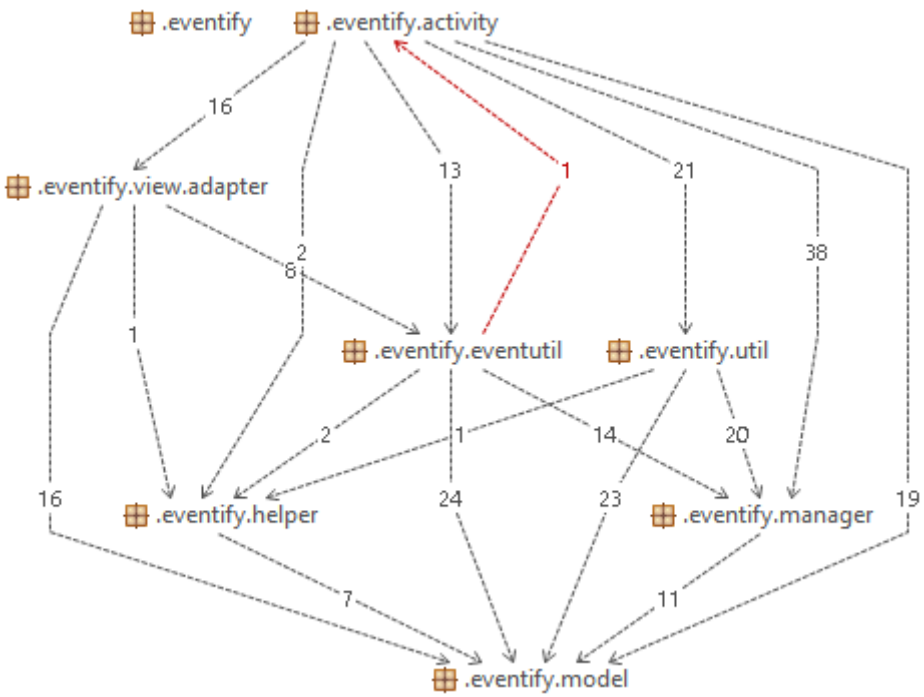

Figure 1. Package structure.

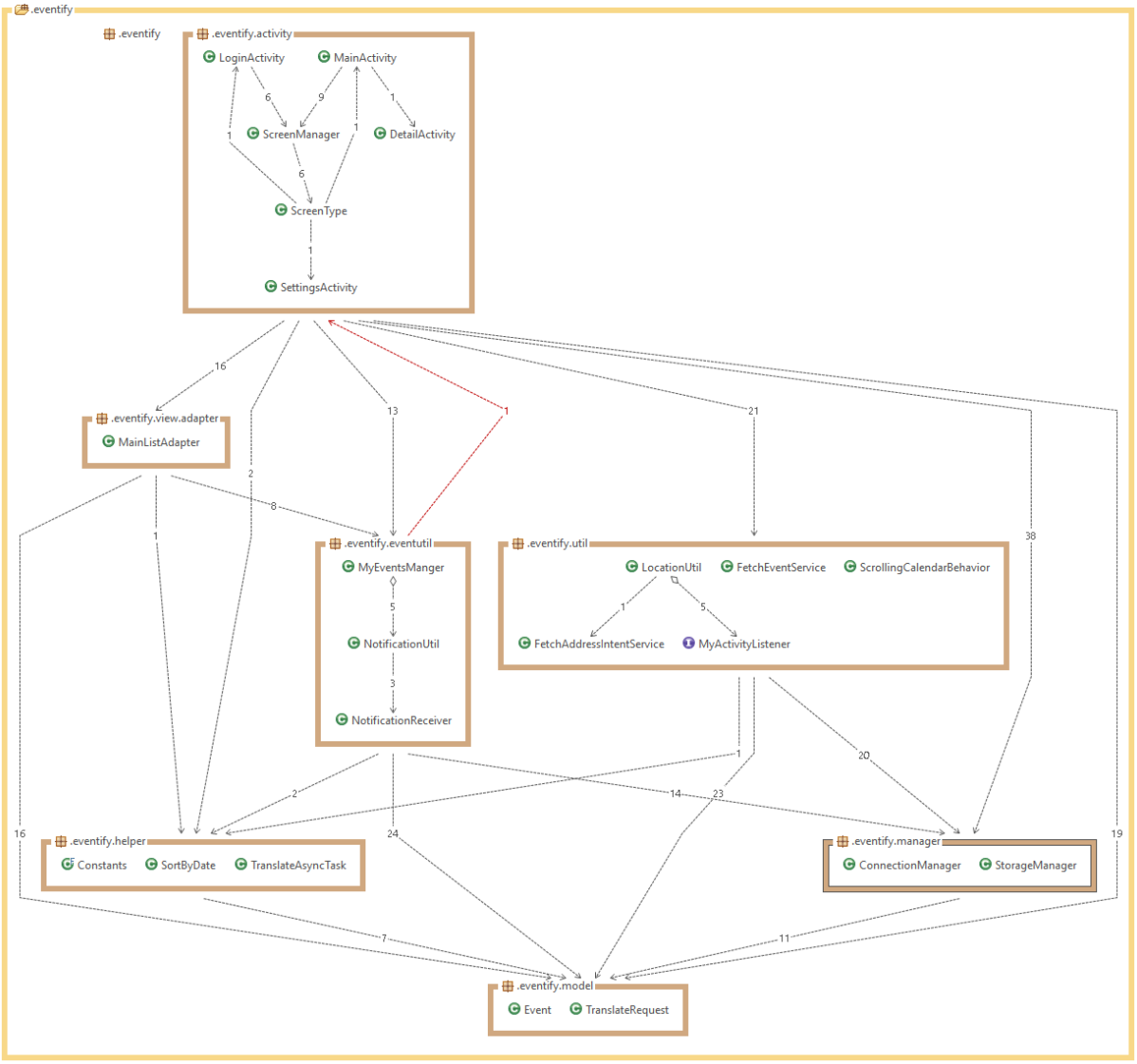

Figure 2. Detailed view of package structure.

Design patterns

Eventify uses the standard android MVC-like design pattern which separates back-end and front-end with activity and manager classes acting as controllers, .xml files and their java classes acting as views and the model classes acting as models. In addition to MVC the design pattern Singleton is widely used in this project. This pattern is used in classes that only should exist as one instance, the reason for this is that the affected classes could otherwise cause runtime exceptions and major bugs. Another reason why Singleton is used is that some classes simply do not need to exist as more than one instance. Like every other design pattern Singleton have both pros and cons but in the case of Eventify it was determined to be a good choice.