System design document for PocketPolitics

Version: 2.0

Date: 2013-05-25

Authors: Group 13
Björn Bergqvist
Harry Eriksson
Johan Hasselqvist
Leif Schelin

This version overrides all previous versions.

Table of Contents

1 Introduction

1.1 Design goals

An Android app targeting minimum API level 14. It should properly retrieve data from the Swedish Riksdag and display it in the app. User data should be saved on and retrieved from a server in a secure encrypted way.

The application is aimed to be secure, extendable and fast enough to work well on common Android devices.

1.2 Definitions, acronyms and abbreviations

- Article, an event, decision or proposition discussed or voted upon in the Swedish Riksdag. Articles are displayed in a list and are likeable by the user.
- Comment, a short text written to comment an article or as a reply to a previous comment.
- Username, the name that will be displayed next to a comment referring to the author of the comment. Login will use email instead of username.
- PUL, Person-Uppgifts-Lagen. The Swedish law for individual privacy rights on the internet.
- Party, political party represented in the Swedish Riksdag.
- Politician, member of the Riksdag who has authority to vote and debate on Riksdag issues.
- GUI, Graphical User Interface.
- MVC, architecture pattern model-view-controller.

2 System design

2.1 Overview

The project will have two sides: one client-side and one server-side. The client-side will work mainly with user interaction but also net-operations like retrieving data directly from the Riksdag. The client should also be able to connect to the server in order to login, post comments and get statistics.

The server-side will mainly handle the requests from the client: updating the database and passing the necessary data to the client. The server should be used minimally to store data and perform calculations; as much as possible should be done on the devices.

2.2 Software decomposition

In order to ensure stability and extendability, the project design is to follow the architecture pattern of model-view-controller: MVC. Each package has a specific task to solve using classes to solve subtasks of that task. Each package should only provide external access to the minimal number of classes necessary, ie. as much logic as possible should be handled internally in a package.

Since the target platform is Android, all client code is written in Java and XML. For the server, requests and handling of the database is made by php-files.

There are several test-classes in a separate source folder. These test-classes are declared in the same packages as the classes they are testing. However they are not displayed in any UML diagram in this document.

2.2.1 General

The MVC layer contains the Model, the Layout, the View and the Control. The Model is composed of pure data-classes and is divided into subpackages depending on the domain. The Layout is a bundle of XML-layout files for Android, some of them need adapters which are found in View. Control is the centerpiece, connecting the packages in MVC with net and make them work together. Control is typically composed of Activity-classes and some pure Controller-class.

The Net package is responsible for all net operations. It retrieves and parses information from the Riksdag and the server to the device. It also makes the queries to the server when something should change in the database, for example a new registered user.

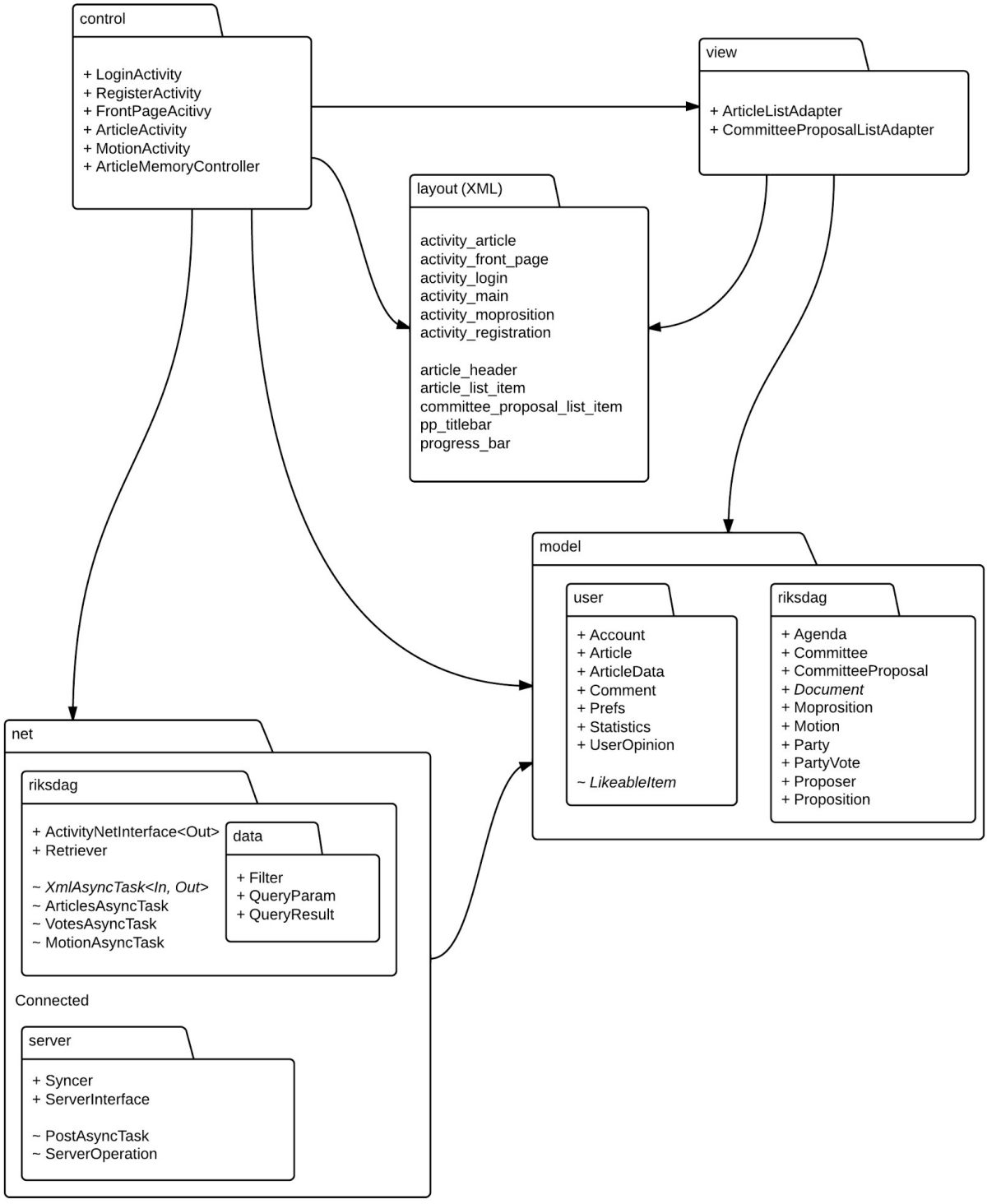


Fig 1. Package diagram. For each package, except layout, there is an UML class diagram in Appendix

2.2.2 Decomposition into subsystems

The Model has two subpackages: one for the Model classes based on the domain of the Riksdag such as Committee, Motion etc and one subpackage for Model classes dealing with the user and the application such as Account and Comment.

The Net package has two subpackages: one for handling net-operations against the Riksdag and one for handling net-operations against the database on the project server. They both use asynchronous methods for sending, retrieving and parsing data. They also have a similar exterior with only a listener-Interface and a control-Singleton visible. Internally however, they are using two different formats and libraries for data-transfer (XML and Json). They also have different architectures. Net.riksdag uses inheritance and generics while Net.server uses enumerations. There are plans for combining the two architectural patterns with multiple interfaces for an even more extensible solution.

2.2.3 Layering

The top layer, closest to the user, are the MVC packages View, Model and Control. This layer handles the user interaction and local data.

The Net packages are running on a tier below, closer to the machine: sending, retrieving and parsing data.

2.2.4 Dependency analysis

Dependency is analyzed using the STAN (stan4j.com) plugin for Eclipse.

Some packages like Model and Layout are entirely independent, like leaves, (see Fig. 1 above). The leaf packages are used or controlled by packages closer to the root like Control and Net.

2.3 Concurrency issues

All network operations on the client are running in a different asynchronized thread from the UI. Concurrency is handled by the Android libraries.

2.4 Persistent data management

Client:
Possibility to save email and password to auto-login. Eventual device settings should be saved locally on the device such as the last used filter.

Server:
User account should be remembered (email, username and a way to verify password securely). User choices such as opinions on articles and comments should be remembered on the server.

2.5 Access control and security

The SQL database on the server should only be accessible through a controlled interface. It

should not accept SQL queries, only specific method calls. All personal information used for verification should be hashed so the user data cannot be compromised even during the event of a breach. If there is time, a system of authentication tokens will be implemented.

2.6 Boundary conditions

Startup: login

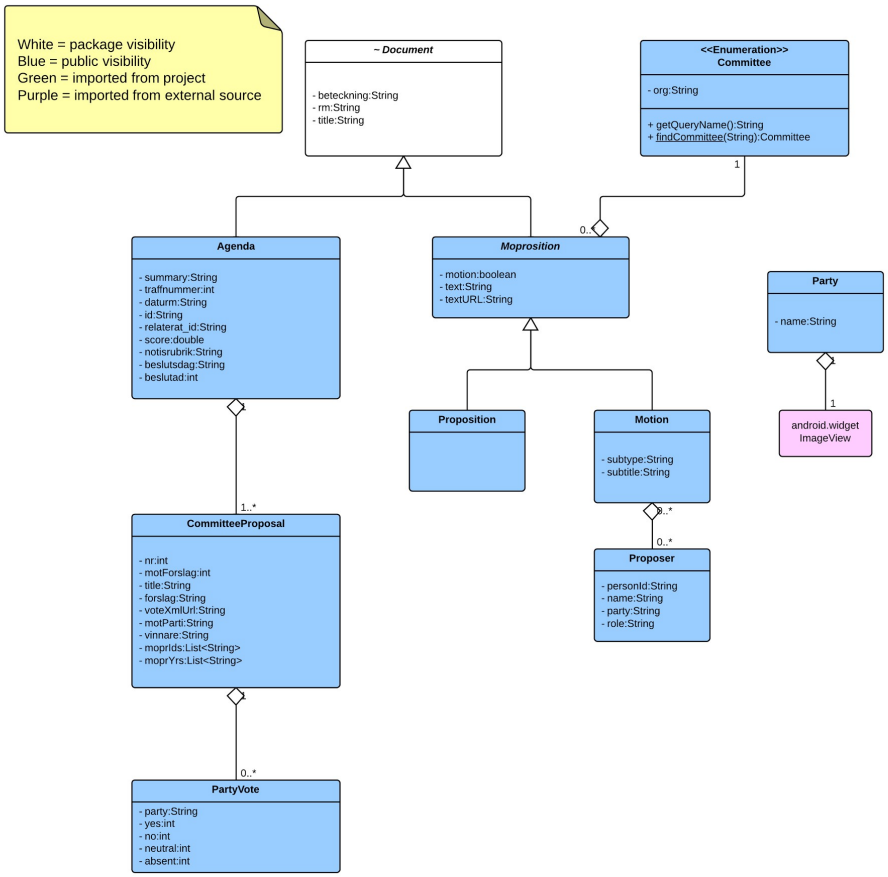
3 References

RAD document v 2.0

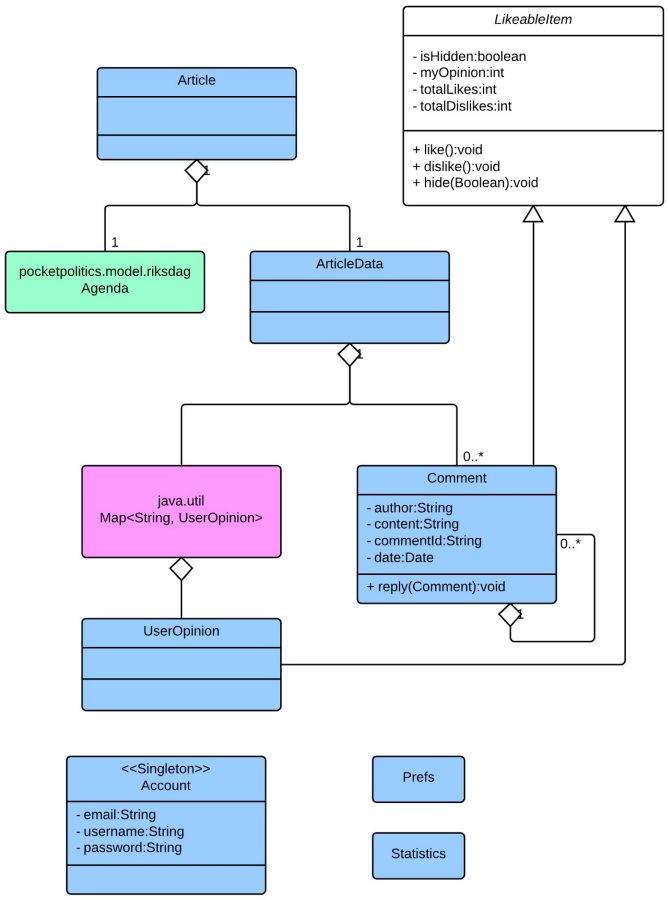
4. Appendix

4.1 UML class diagrams

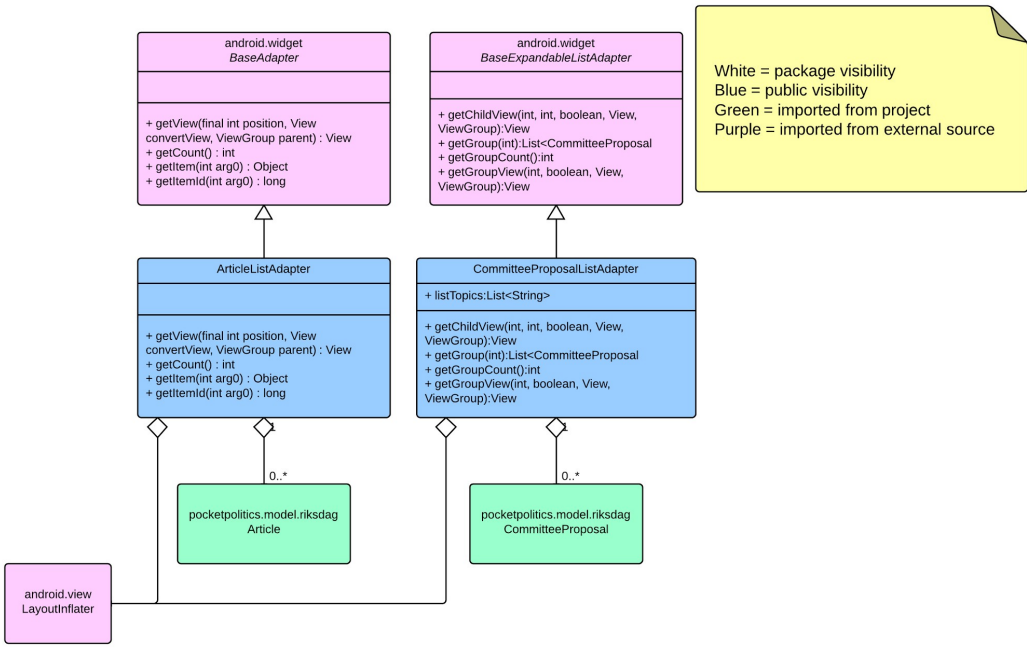
4.1.1 Package pocketpolitics.model.riksdag



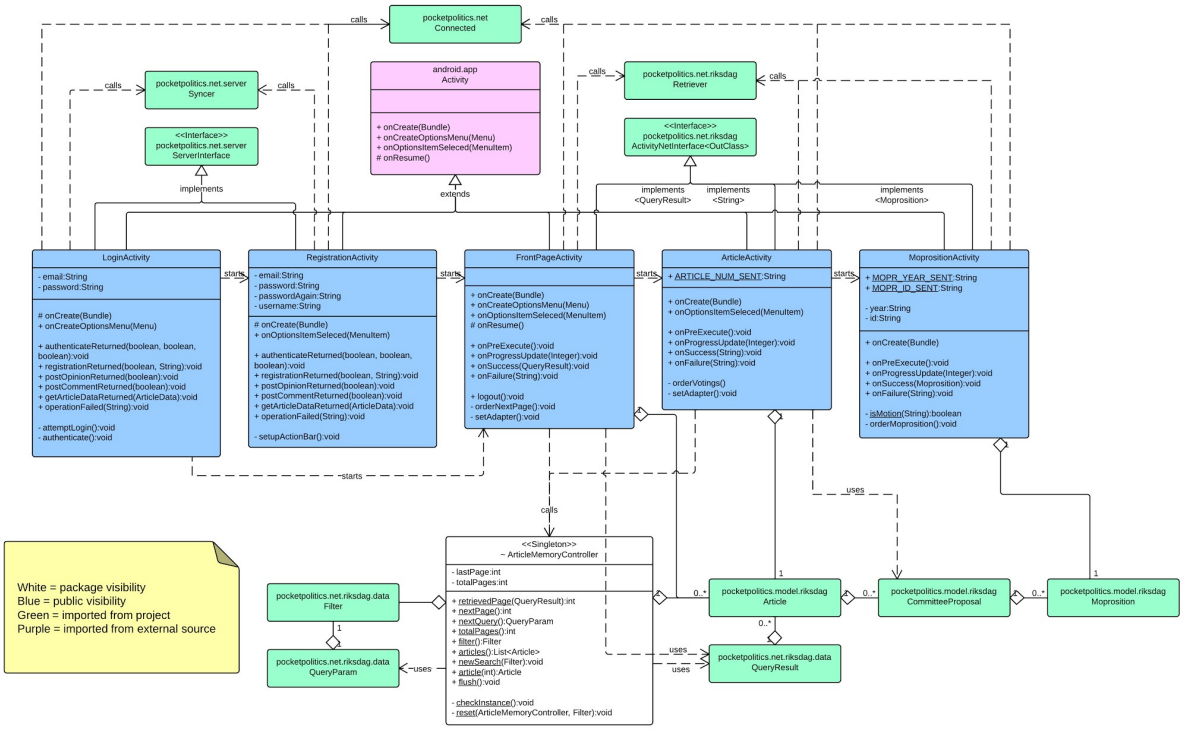
4.1.2 Package pocketpolitics.model.user



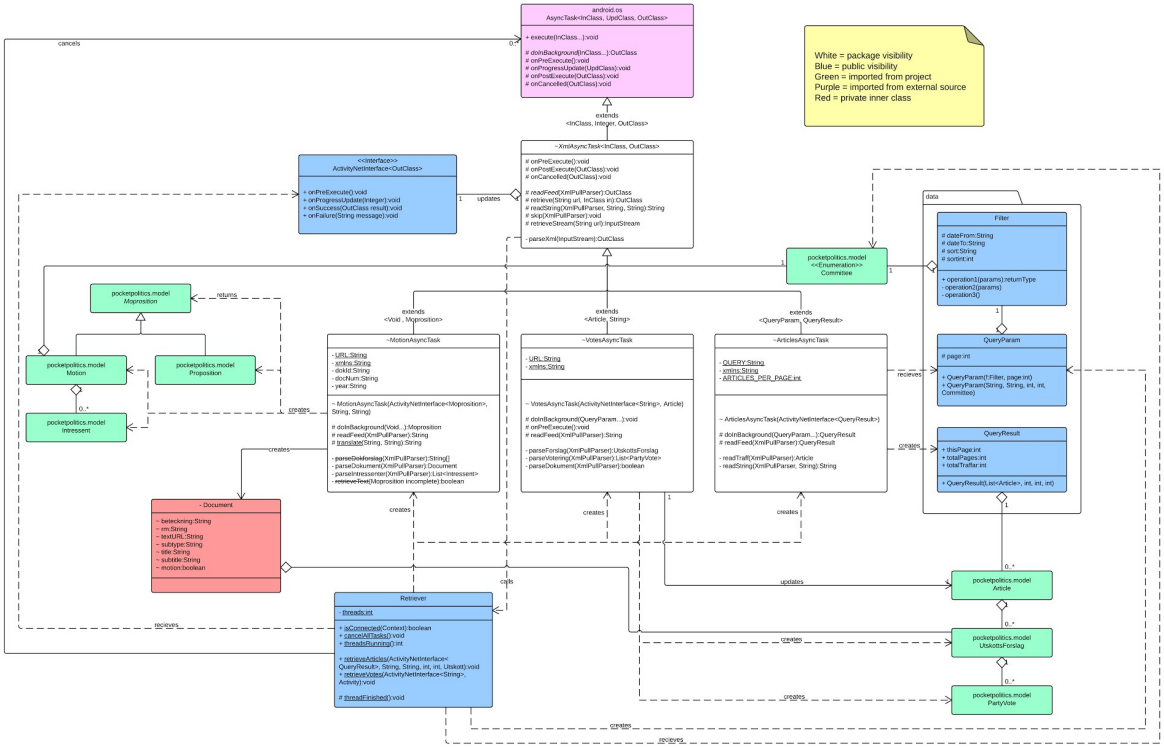
4.1.3 Package pocketpolitics.view



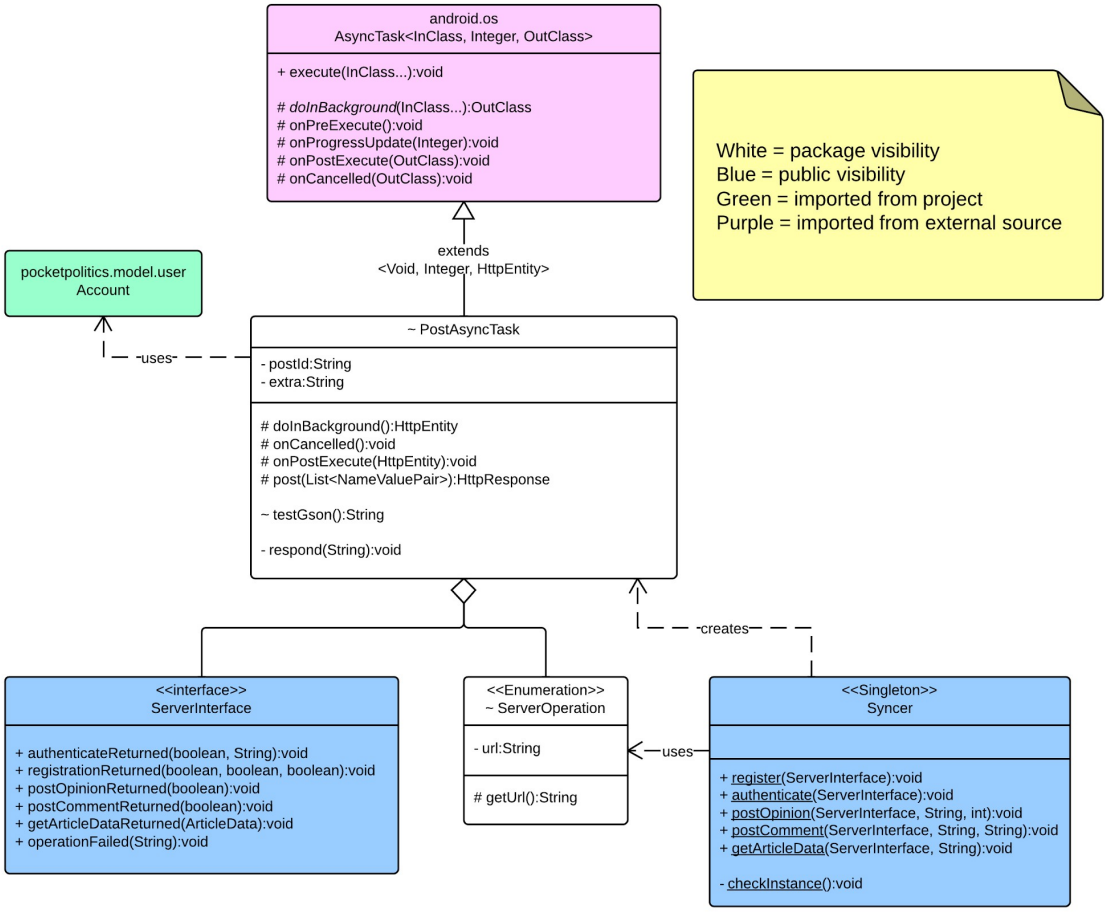
4.1.4 Package pocketpolitics.control



4.1.5 Package pocketpolitics.net.riksdag

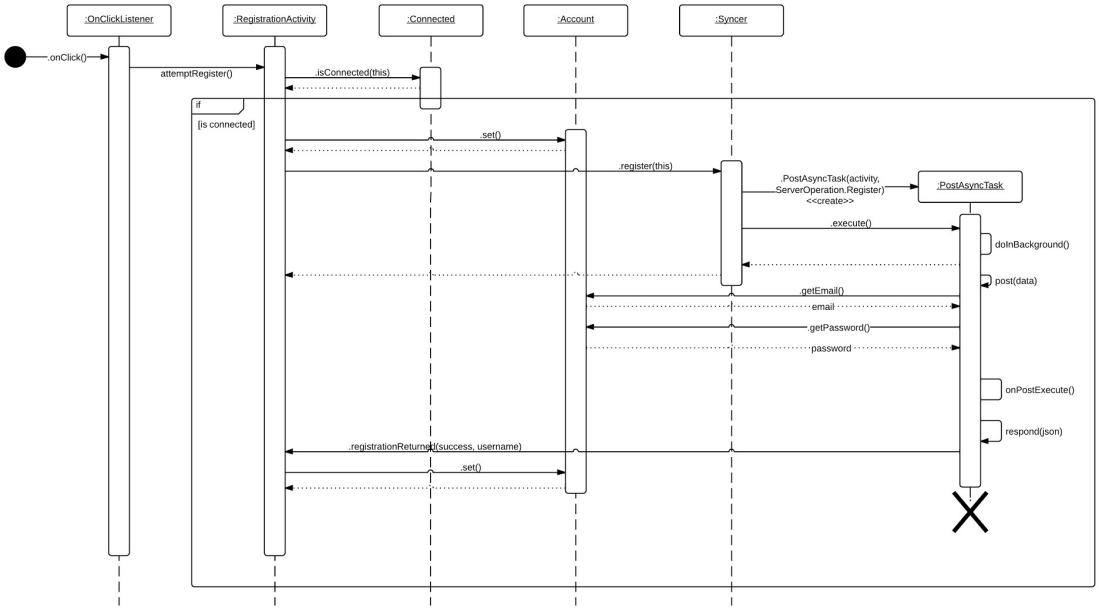


4.1.6 Package pocketpolitics.net.server



4.2 UML sequence diagrams

4.2.1 Registration



4.2.2 Opening an article