Hangman

Martin Mårtensson

November 2, 2020

This is a game where you can pick a category which holds a number of words. The game will shuffle the words and the player will try to guess a hidden word, letter by letter. If the player guesses a letter which does not exist in the word, then a man, will slowly, step by step, be hung by his neck, on the screen.

The app has some flexibility and gives the user a possibility to change, add, and remove words, and categories from a database on which the app is connected to [using the REST API.](https://github.com/DAT4/android-galgeleg-rest-api)

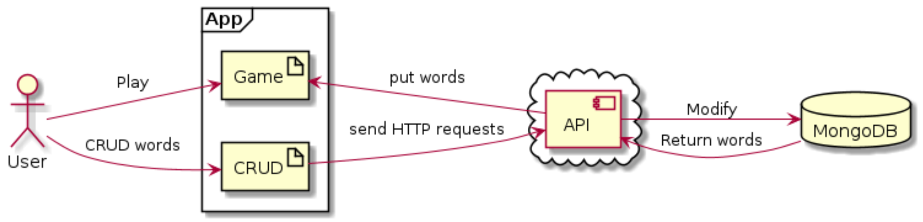

Figure 1: Diagram of the interaction between different components in the system, the app is using.

1 Patterns

In the development of the app one key requirement has been to follow software patterns, for me to lean about that.

1.1 MVC

I have chosen to use MVC pattern (Model View Controller), which came naturally to me when I began developing the app, because I worked with this pattern before, and it helped me a lot creating the structure of the game. The MVC pattern is heavily used in the Game Part of the app.

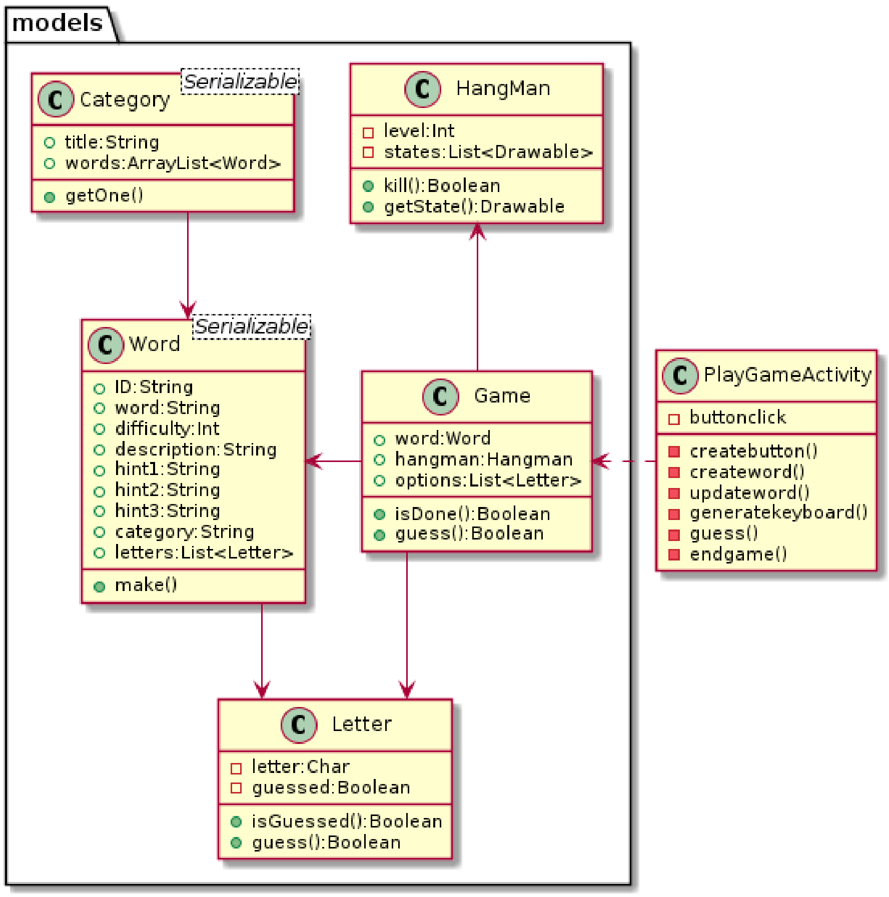

Figure 2: UML diagram of how the MVC pattern is implemented in the app

1.2 Observer Pattern

Since I am getting data from the internet, I thought that it will be nice if there was a way to cache the received data on the phone, and share it between Activities, and I was considering using something like a Singleton, or static class. I ended up creating a Kotlin-Object as a **Subject/Observable** and making the Activities who wants to know about the data, **Subscripers/Observers** in their ‘onCreate()‘ method, then the observer pattern takes care of updating the views on each active Activity each time the data changes.

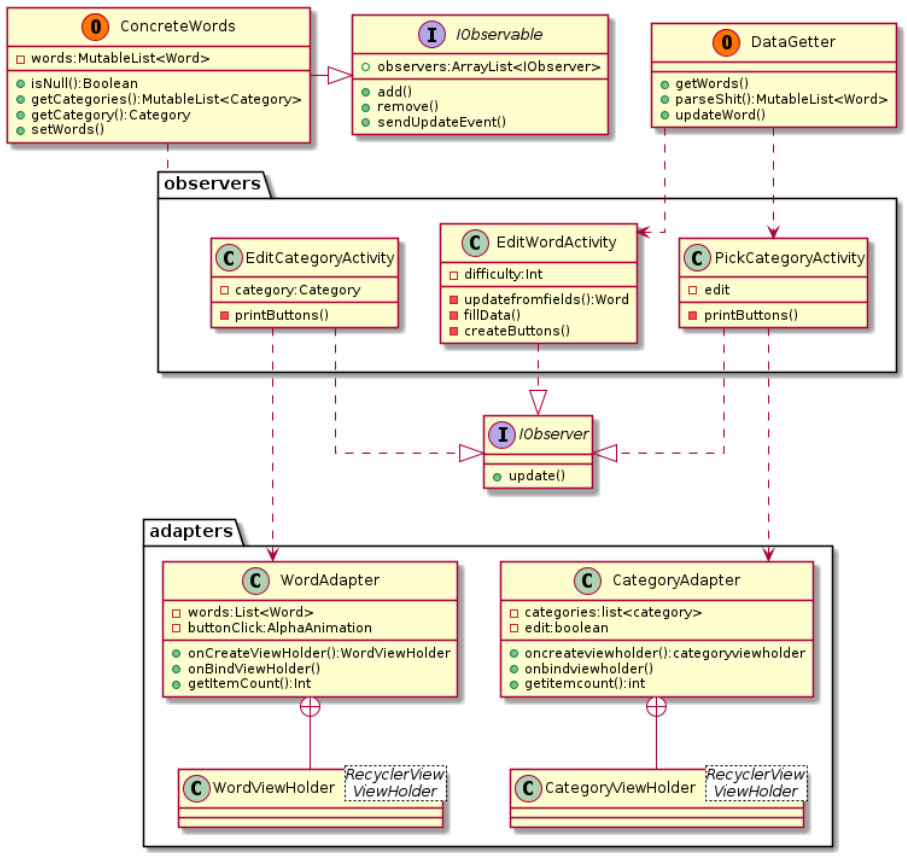

Figure 3: UML diagram of how the Observer pattern is implemented in the app

1.3 Other patterns

Android development is using a lot of different patterns, like for example the way an Adapter Pattern is used to connect a list of objects to a recycler view, and then do some magic in the background so that the app developer does not have to do it.

2 To do

-
- Let user pick difficulty
- High score activity show high scores pr category / word
- More patterns
- Make app prettier
  - Buttons/Keyboard
  - Learn animations
- Add statistics words/terms
  - Dynamic image from description(equations)
- Refactor redundant code by use of patterns.
  - Patterns
    - Composition
    - Factory
  - Code
    - DataGetter
    - Adapters

3 Notes:

I implemented everythin related to HighScores in the last minute so I didn't have time to document it this time.