Word Game documentation

The development of this test application was made using the MVVM architecture, consisting of TranslationPair as the model, MainActivity as the view and the GameViewModel as the viewmodel.

The TranslationPair contains the two words, on in each language, and an indicator if they are a correct translation match, this model is used to provide the game with the words that appear on the screen as well as the data source for the game. This data is parsed from the Json file provided whenever the game view is created and stored in memory.

The MainActivity hosts the game, and is the starting point of the app. It is responsible to create the viewmodel and respond to its changes, manipulating the views and sending the appropriate events to the viewmodel.

The GameViewModel is data is stored, it is responsible to managing the informations displayed on the screen as well as manage the status of the game based on the inputs of the view.

Upon the app loading, the words would be fetched from the Json file, parsed to the model and stored on the viewmodel. As the user presses “PLAY” button the game will start and the spanish words will start falling with an english word on the bottom of the screen, the player should guess if the translation was correct and press the respective button, a feedback will be presented if the user got it right or not and the appropriate score will be granted, and the next word will appear. The game will run for 30 rounds. If the user does not press any of the buttons, when the word reaches the bottom, the round is over and it will be counted as a miss. At the end of the game, the user will be presented with a game over message and the play button again. Pressing the button, resets the scores and a new game begins.

The time

This project took about 7 hours to be developed, most of the time was spent on the view, changing the layout and playing with the animations, as there was no planning for how the app was going to look and behave a lot of time was used to figure out what looked and felt better. As a contrast, the viewmodel cost quite less time, as the way the words would be fetched and the scores would be counted, not much needed to change, just a small update on when the game would end.

Although, a good portion of time was used tring to fix a problem with gson library and kotlin, where the boolean field “isCorrect” was always set to false when parsing the Json file, despite its default value being set to true, after a while investigating and trying to fix it, the solution was to switch to moshi library, where this problem does not occur.

Due to the lack of time some things were left out, such as proper screen rotation handling and fancier animations/layout, although time was spent attempting it; And other things weren’t even attempted, like testing. Given more time, some aspects that could be improved are the layout and animations, but what could be added are for sure the tests cases, and maybe a game menu and ranking/score system.