2015

Guess the Movie

MOBILE PROGRAMMING WITH ANDROID

MARKOWITSCH GEORG

ANDRIANOV ANDREI

Contents

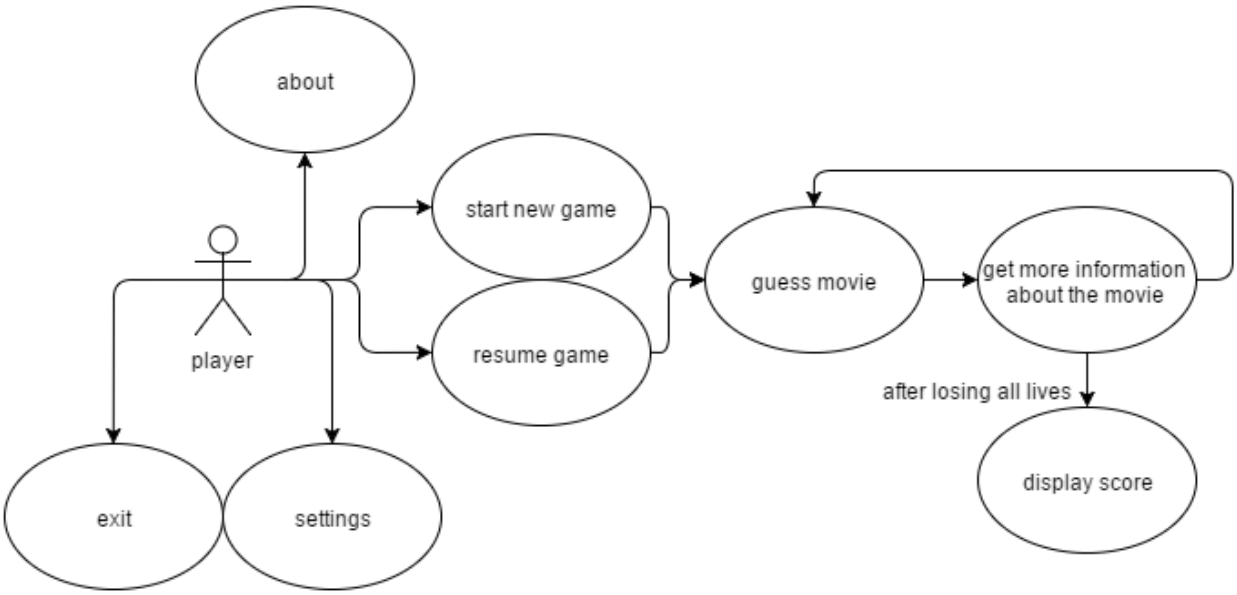
Introduction

Guess the Movie is an Android Game where you get a screenshot of a unique scene of one specific movie out of a random pool of movies and guess the name based on the information on the picture and randomized letters of the movie title. The pool from which the movies are randomly picked can be specified by the user in the Settings of the application. All the movie data and pictures are loaded via the API provided by themoviedb.org.

Objective

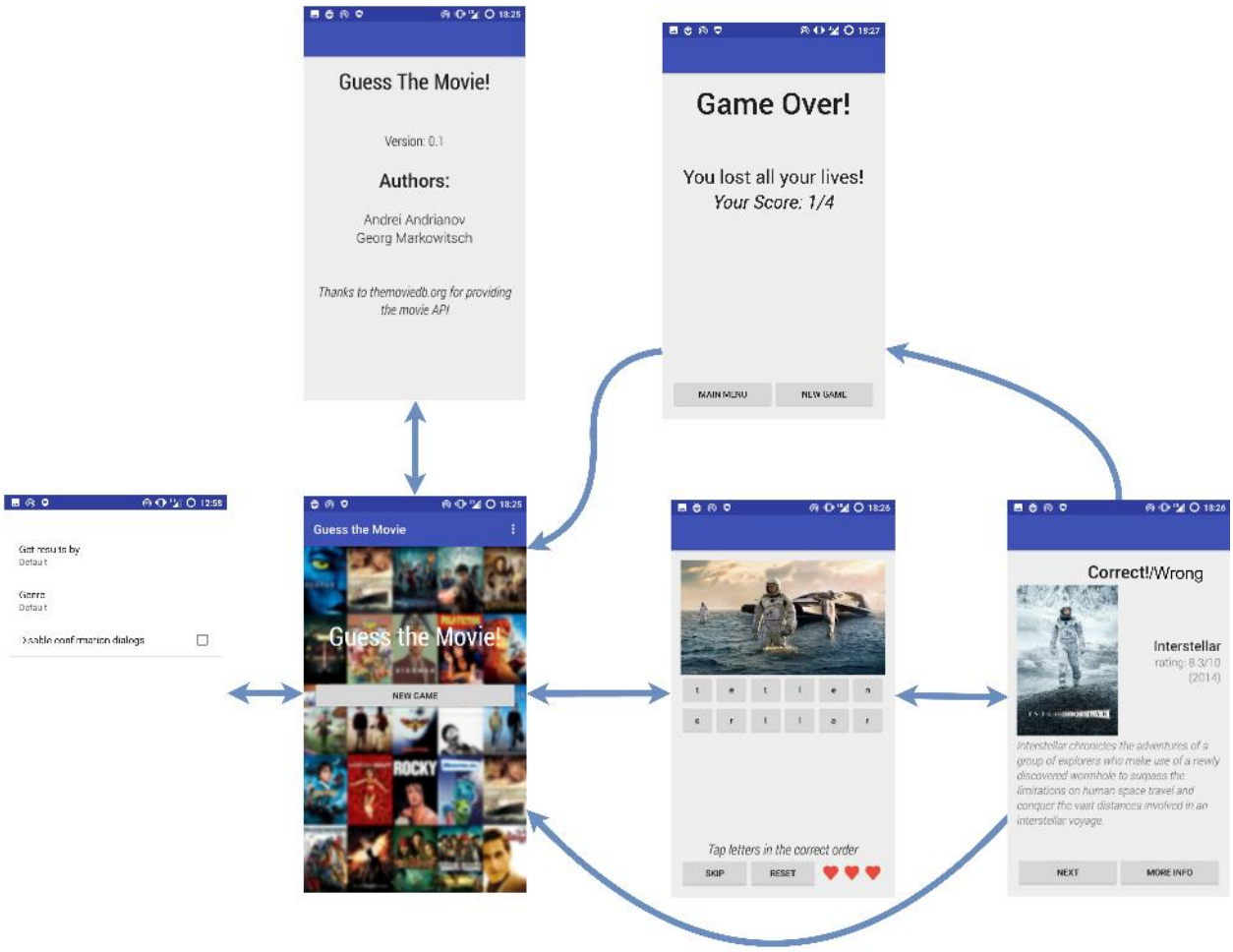
The main objective in this game is to get a fun activity while on the bus or waiting for someone. In addition to that, if you stumble over a movie you did not know, you can also learn more information about it.

Use Case Diagram

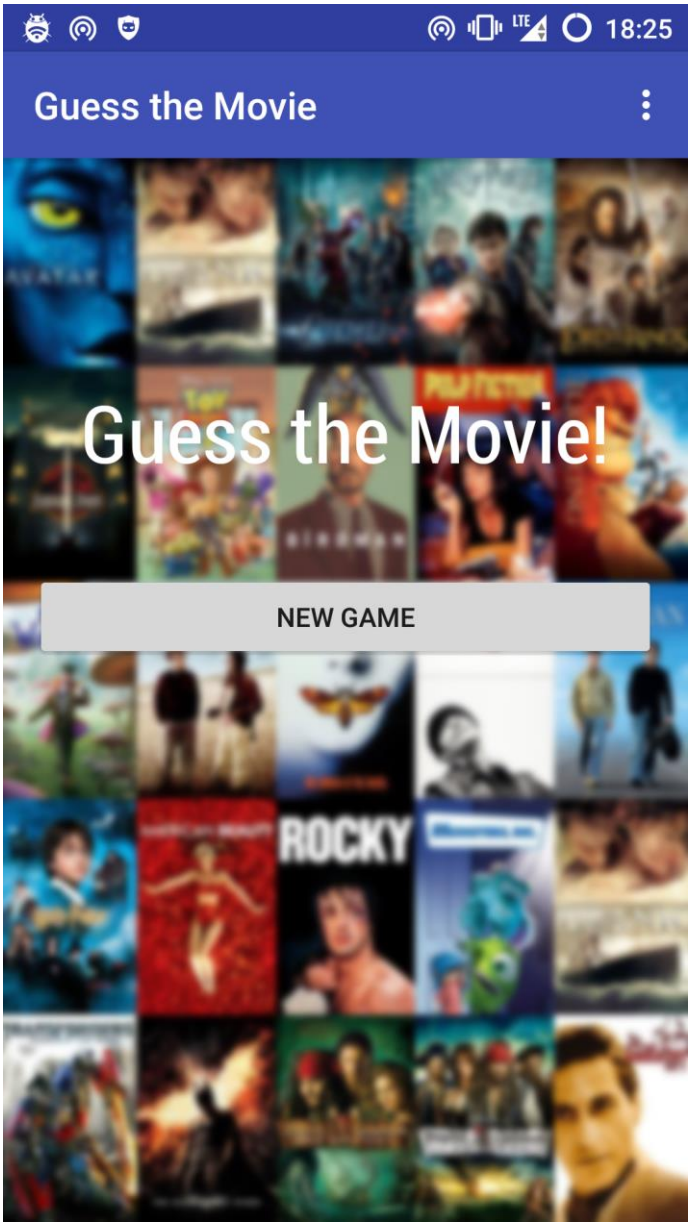


Activities

Overview / Screen flow

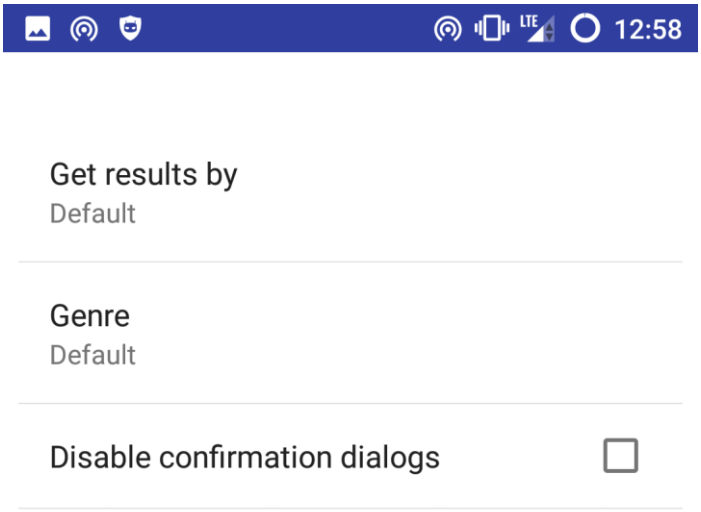


Main Menu



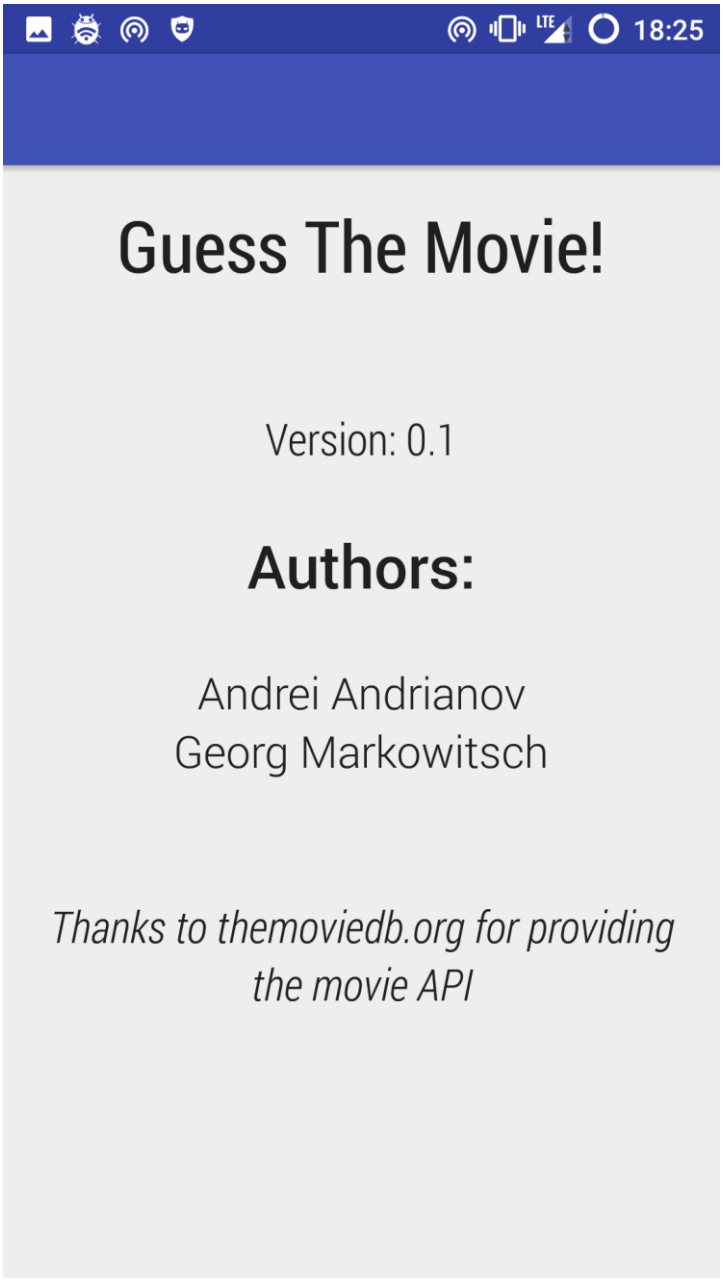
First thing after the application is started. Here the user can start a game, configure the settings or learn more about the Application.

Settings



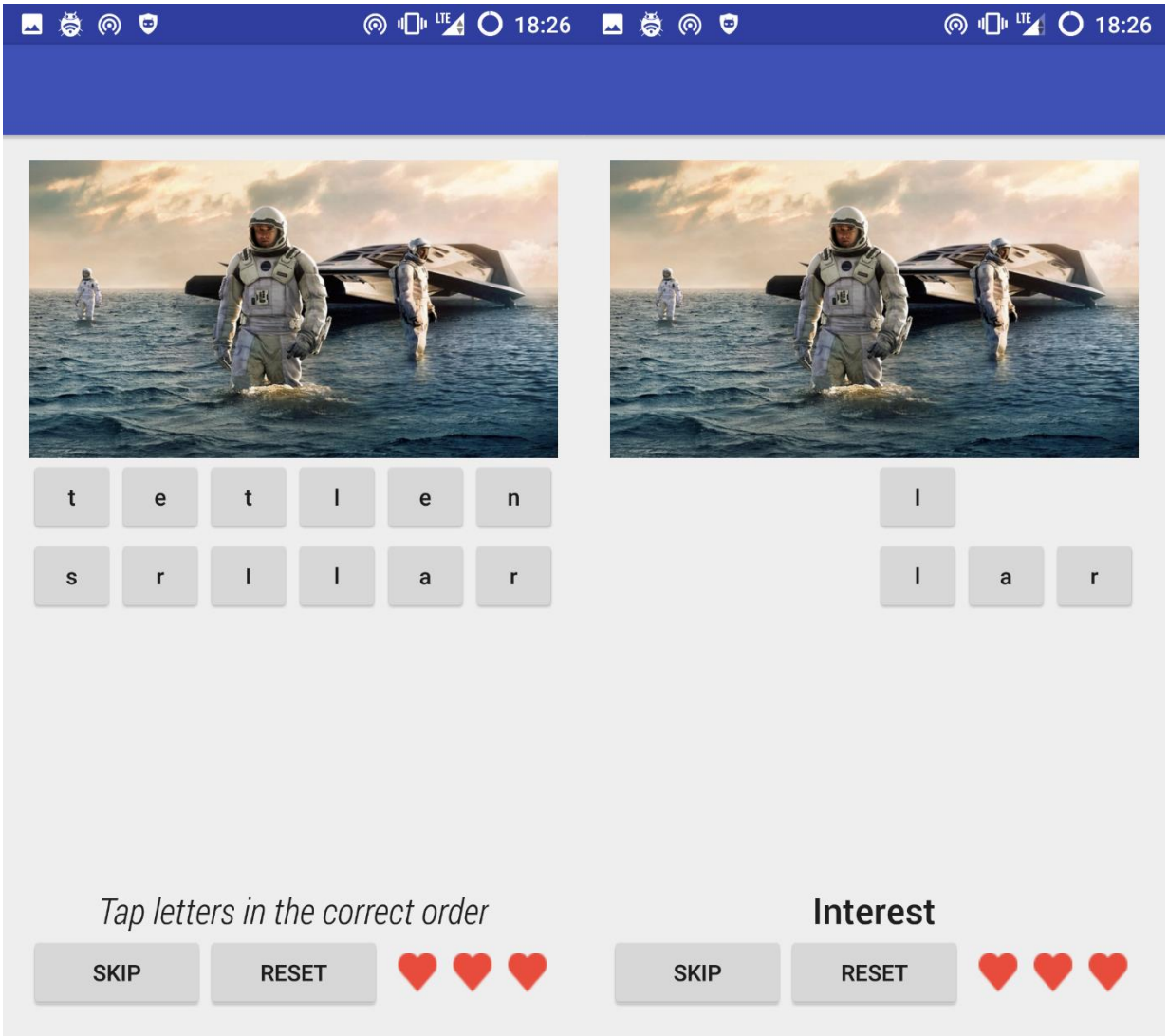
Different options to configure the pool of movies. Currently there are the options to get movies in order of popularity or rating. This affects only how the movie data is received from the server, during the game, the user is presented with a random movie from the received data. This is a precaution to make the make more interesting because the order of the movies will be random. The user can also specify to only get movies fitting in a specific genre and the last option is to disable the confirmation dialog which pops up if the user wants to skip a movie.

About



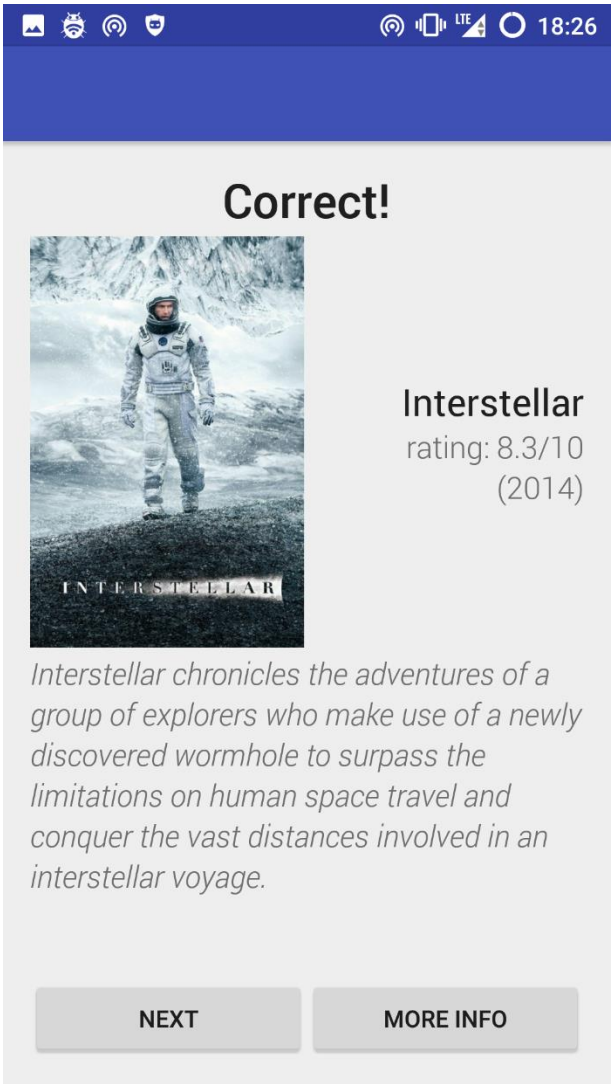
Information about the authors of the game and the version.

Guessing



After starting the game the user will be presented with a picture of a movie picked randomly from the movie pool. Below the picture the name of the movie is randomized and has to be brought in order by tapping the letters in the correct order. Upon tapping a letter, the button will disappear and the letter will be added to the TextView below. If the user made a mistake, there is the option to reset the buttons and the TextView using the reset button. If the user cannot guess the movie, there is also the option to skip it losing one live. Lives are visible heart images next to the reset button. If all the letter-buttons are used, the game will the guessed title with the original title and if does not match, the user will lose one live. Either way, the next activity is about displaying additional movie information.

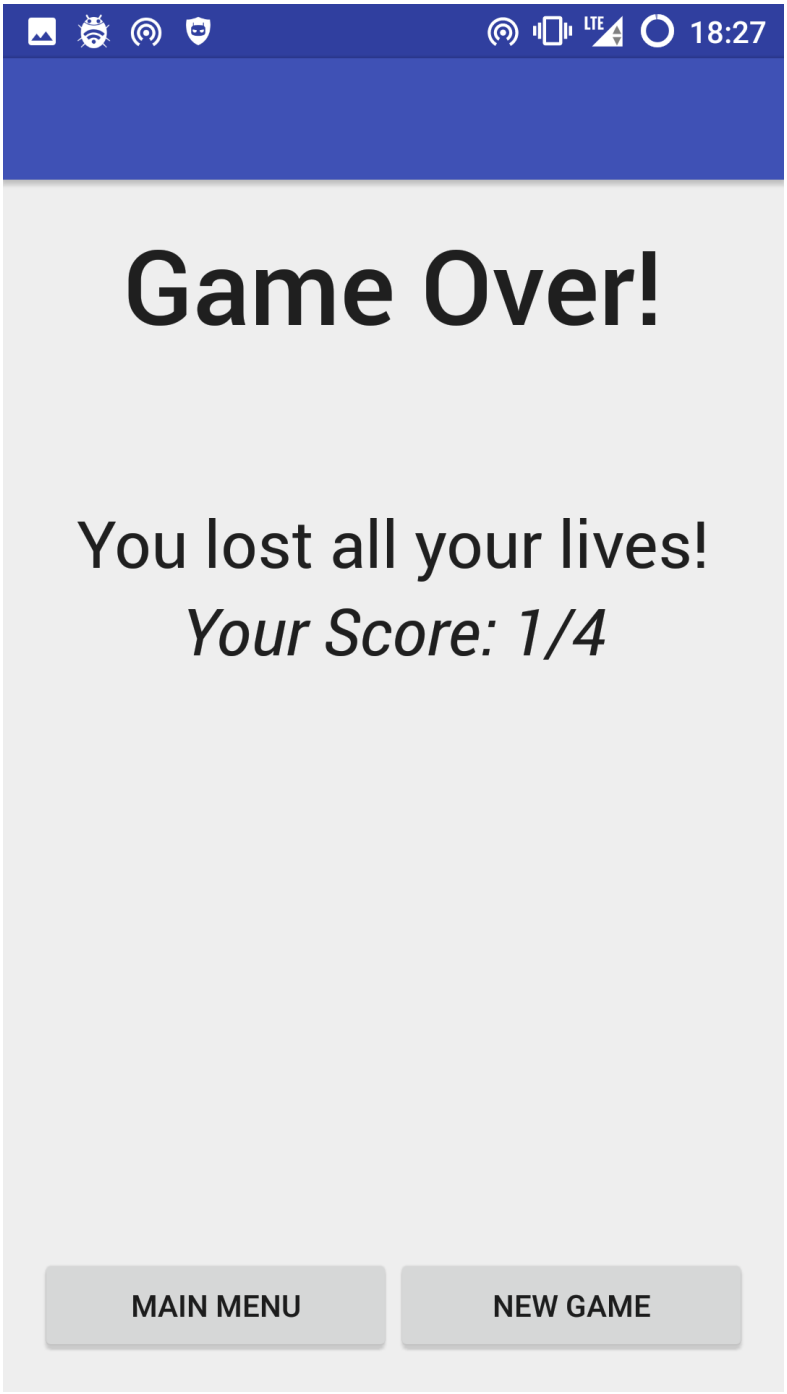
Movie Information



Depending on if the movie was guessed right or not, the user will be given the correct feedback and is presented with some basic information of the movie including the title, the current rating on themoviedb.org, the release year, the original movie poster and some short overview of what it is about. The user has now the option to open the movie on themoviedb.org by pressing the more info button to get even more information about this particular movie like for example actors and trailers.

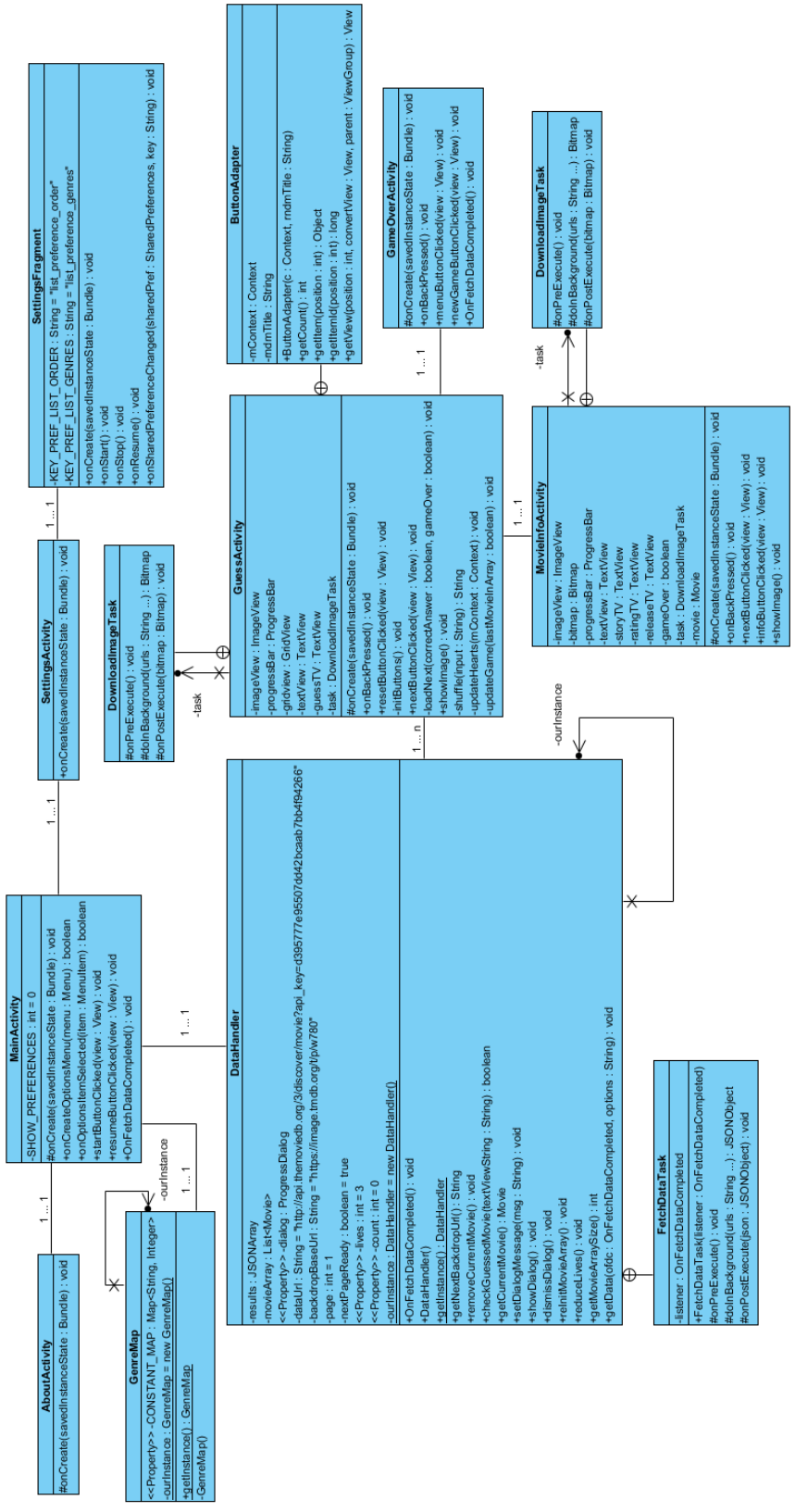
By clicking on the next button, the game will continue if the use still has sufficient lives left. If not, the game over activity will be displayed showing the score the user has achieved.

Game Over



After the user lost all three lives, the game over activity will display the achieved score and give the option to directly start a new game, or go back to the main menu to modify some settings.

Class Diagram



Backend

All data is loaded from the API of themoviedb.org and is available for free as long themoviedb is mentioned as source and the application does not collect revenue. To access the API an API-key is needed which can easily be obtained by registering an account and applying for it. The data will be delivered in JSON format.

Currently there is a limitation of 40 requests in 10 seconds per ip-adress which is perfect for this application, because we will only need one API-key for all installations on various devices.

How the API is used

In this application we mainly use the discover movie function of the provided API with some additional options. Using this we can get multiple results ordered and filtered how we want them and only with one request.

The basic URL of discover movie:

```
http://api.themoviedb.org/3/discover/movie?api_key=<API_KEY>
```

This responds with data in JSON format containing the 20 currently most popular movies on themoviedb.org.

Here an example of the start of the JSON data plus the first movie:

```
{ "page": 1,
  "results":[
  {  "adult": false,
    "backdrop_path": "/kvXLZqY0Ngl1XSw7EaMQO0C1CCj.jpg",
    "genre_ids":[28, 12, 878],
    "id": 102899,
    "original_language": "en",
    "original_title": "Ant-Man",
    "overview": "Armed with the astonishing ability to shrink in scale but
increase in strength, con-man Scott Lang must embrace his inner-hero and help
his mentor, Dr. Hank Pym, protect the secret behind his spectacular Ant-Man
suit from a new generation of towering threats. Against seemingly
insurmountable obstacles, Pym and Lang must plan and pull off a heist that
will save the world.",
    "release_date": "2015-08-14",
    "poster_path": "/D6e8RJf2qUstnfkTslTXNTUAlT.jpg",
    "popularity": 70.003821,
    "title": "Ant-Man",
    "video": false,
    "vote_average": 6.9,
    "vote_count": 1744
  },
```

As this only loads 20 movies, we are using the page option to load the next page, which is another set of 20 movies, if all the already loaded movies have been used.

`http://api.themoviedb.org/3/discover/movie?api_key=<API_KEY>&`**`page=<#>`**

The default sort order is by popularity.

`http://api.themoviedb.org/3/discover/movie?api_key=<API_KEY>&`**`sort_by=popularity.desc`**

When using the sort order by average voting we also use an option to only get movies with at least 1000 votes to avoid a lot of unknown movies.

`http://api.themoviedb.org/3/discover/movie?api_key=<API_KEY>&`**`sort_by=vote_average.desc`**`&`**`vote_count.gte=1000`**

To get only movies with a specific genre, a genre specific number has to be sent. For example to get only action movies, the following request URL is used:

`http://api.themoviedb.org/3/discover/movie?api_key=<API_KEY>&`**`with_genres=28`**

Workload and Planning

WEEK	TASK	HOURS
43	finding project topic	5
44	planning documentation	5
45	software design	3
45	tested how the API key works, which options we need	5
46	started the project in android studio, prepared main and guess activity	5
46	learned how to do ajax calls with android	3
46	created a button adapter for the guess activity to dynamically create buttons	4
46	added a scroll view to the button adapter for extra-long titles	2
46	first simple endless gameplay for testing purpose	3
47	introduction of lives to disable endless gameplay	3
47	added a game over activity to display amount of movies guessed correct	3
47	added a movie info activity to display movie info and if the guess was correct/wrong after guessing	5
47	a lot of fiddling with the movie info activity layout xml (scroll view brought problems)	4
47	added settings menu with order and genre options	4
47	added about activity	1
47	general UI cleanup and formatting	2
48	fixed a bug leading to a out-of-memory error	5
48	sent apk to friends/family for bug testing	0.5
48	collecting feedback	1
48	fixed a bug with preferences not being available before opening settings fragment	3
48	fixed a bug causing multiple main activities	2
48	final documentation	6
49	presentation	2.5

Sources

Themoviedb.org: http://themoviedb.org

Themoviedb.org API: http://themoviedb.org/documentation/api

Themoviedb.org API-ary documentation: http://docs.themoviedb.apiary.io