Queen Mary University of London

Group project Report

Module: Functional Programming

Team members:

- Johannes Hartman
- Liam Kelly
- Juan Manuel Campos Villarreal

Date: 2017-12-14

1 Solution Design

This report outlines the design and development of a Haskell program capable of searching and retrieving a list of Cinemas in the UK which are currently playing a specific movie.

The main functionality of the application is as follows:

- Retrieve all the recent movies from the “TheMovieDB” API and store them into a SQLite database.
- Retrieve the list of actors that perform in all the movies by calling the “TheMovieDB” API and store them into the Database
- Look up for a specific actor (specified by user) in the database and retrieve the list of movies the actor is playing in.
- Retrieve a list of cinemas in a specific area (specified by user) that are playing the chosen movie.

1.1 User Manual

Program compilation:

To compile the program, run the following command in the terminal:

```
> stack build
```

This will generate the program executables

Program execution:

To execute the program, run the following command in the terminal:

```
> stack exec groupproject-exe
```

This will load the components of the project and start the program execution.

Main functionality:

Upon its first execution, the main module will download all the required data from the API’s, create the database and store the data in it (this may take a while).

After the program has finished its basic configuration, the user will be able to start using the application.

After the program starts, the user will be required to type the name of the actor their interested in (see example below) :

```
Please enter the actor name you want to search for:
> Ben Affleck
```

The program will then retrieve the list of movies released in the past two months (if any) that the secified actor performs in, and the user will be required to select a movie from the given list (see example below):

> *The given actor plays in the following movies*
>
> *(1) Justice League*
>
> *(2) Laddie*
>
> *Please select a movie from the List:*
>
> *>1*

After the user has selected the Movie, the user is required to type the location to search for the cinemas in that area:

> *Please enter your location:*
>
> *>Stratford*

If the user provided a valid location, the system will return the list of cinemas in the user location that are playing the selected movie.

1.2 Program Modules

The whole Haddock documentation is hostet online at:
https://incrediblehannes.github.io/QMUL-FunctionalProgrammingGroupProject/index.html

as well as included to the source folder.

1.2.1 DatabaseModule.hs

This module is responsible for all the database functionality of the application: Database creation, main queries and db connection.

1.2.2 DataStructures.hs

This module contains all the custom data structures that are used in the application. The created datatypes will hold the data retrieved and parsed from the different API’s as well as the data extracted from the database.

1.2.3 HTTPRequestModule.hs

This module is responsible for making the HTTP calls to the “TheMovieDB” API.

Two different calls to the API are being made in this module:

- The first call is responsible for retrieving the list of Movies released from a specific date.
- The second API call is responsible for retrieving all the actors playing in the given list of movies from the "TheMovieDB" API.

The data retrieved from the API comes in the JSON format, and this module is also responsible for returning the parsed data to their respective data structure.

1.2.4 HTTPRequestModule2.hs

This module is responsible for making the HTTP calls to the "CineList" API.

Two different calls to the API are being made in this module:

- The first one will retrieve a JSON with the list of cinemas by searching for a specific UK location.
- The second call retrieves a JSON with the list of movies that are currently playing in each one of the cinemas for the given location.

The data retrieved from the API comes in the JSON format, and this module is also responsible for returning the parsed data to their respective data structure.

1.2.5 IOActionModule.hs

This module is responsible for providing all the necessary functions for handling the user interaction with the system. It contains methods which print the required data on screen and obtain the user inputs from the console.

1.2.6 JSONParserModule.hs

This module takes care of the JSON parsing for each HTTP module. It uses the Aeson library and defines the parsing methods that are going to be used by each HTTP module to parse the different responses to their specific data structure.

Extra Features:

- We have created a Unit Testing Module. This module contains methods to test the complete functionality of the application. It has functions to test the database setup, database information, data cleaning, inconsistent data, and complex querying; It also contains test cases for the JSON parsing module including (Movies, Cinemas, Actors, and Movie Listings). Responses from the HTTP Modules are also tested.
- We are making use of 2 API's in the application. There are 4 different API's calls in total, and they're not used just for downloading the data and storing it in the DB. We actually use this data to construct a new API request for each one of the response elements on the first request to get extra useful information.
- The DB module contains complex logic that enables us to clean the database to avoid using too much memory, as well as only download the movies data when they're not in the db to avoid too much traffic.

API's used:

TheMovieDB:
https://developers.themoviedb.org/3/getting-started/authenticatio

CineList :
http://www.cinelist.co.uk/

- http://api.cinelist.co.uk/search/cinemas/location/
- http://api.cinelist.co.uk/get/times/cinema/