Kyle Russell - 13831056
Distributed and Mobile Systems
A3 Documentation

CONTENTS

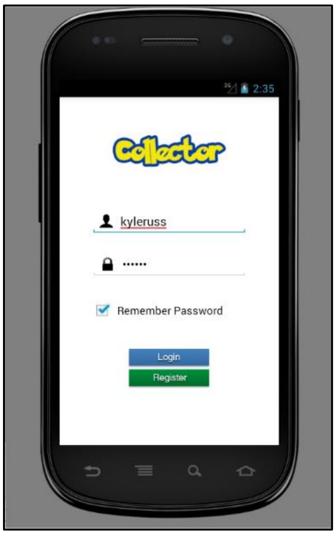
1.0 About

The ‘Collector’ app is an android application that allows users to collect card items, manage them and trade them with their friends. The context of the traded items used in this project was the ‘Pokemon’ trading card game where users collect and trade Pokemon cards. The aim of this project was to provide a fun and useful way to collect and manage items. More specifically, users initially create an account with the Collector service followed by signing in where they will be given access to their account friends, cards, decks, settings and more. Users start off with 10 random cards to begin their collection, each day a user signs in they are granted with 2 new random cards and users can trade any of their cards with others which allows them to grow their collection. Users can also create decks and add their favourite cards to them for later viewing and sharing with their friends.

2.0 User Guide

2.1 Signing in

In order to access your account you will need to first sign in to the application. To sign in, launch the application and you will be first presented with the sign in screen as shown in figure. 1. The first field asks that you enter your unique, 3-16 character username. The second field asks for your 6-18 character account password. Lastly, the ‘Remember password’ checkbox if checked, saves your credentials to the phone such that for future app launches, your username and password will already be entered. When you are ready to sign in click the ‘Login’ button and your login request will be processed. If your credentials are correct then you will automatically transferred to the home page where you can manage your account otherwise you will be notified with an error message. If you do not have an account with Collector then see section 2.2 for creating an account.



kyleruss

••••••

☑ Remember Password

Figure 1: Sign in screen

2.2 Registration

Users can create a new account with Collector by launching the application and instead of attempting to login, click the ‘Register’ button as shown in figure 1 which will redirect you to the registration screen as shown in figure 2. Before continuing with the registration, you will need to enter a unique 3-16 character username, 6-18 character password, re-enter the password and finally a valid email address. Once you have completed the form and are ready to continue, click the ‘Register’ button at the bottom of the screen to proceed with the registration. On successful registration you will be redirected back to the login screen where you can use the username/password that was entered in the registration form, to sign in. If registration is unsuccessful, you will be notified with an error message. Please check you have entered the correct information as per the field requirements before proceeding.

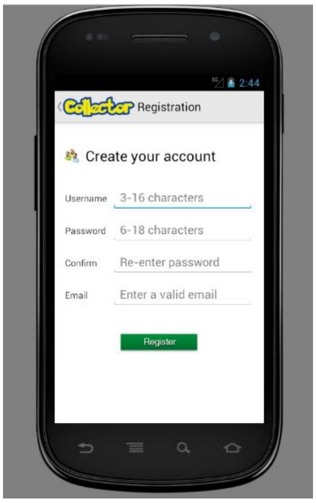


Figure 2: Registration screen

2.3 Friend management

The Collector app allows users to maintain friends such that they can trade cards easily to each other. To view your friends, you must first be logged in. Select the ‘Friends’ option in the home navigation and you will be displayed a list of your current friends and controls for managing them. To add a new friend click the ‘Add friend’ button and enter the unique username of the friend you wish to add, if this user exists then they will be sent a friend request to be your friend. To remove a friend click the ‘Remove friend’ button and enter the users unique username, if this user exists and is on your friend list then they will be removed.

2.4 Settings

To manage your Collector account settings, first ensure you are logged in then select the ‘Settings’ option from the home navigation. You will be presented with a form that has your account details filled out, you can change any of these details as you see fit and when you are done click the ‘Save settings’ button. For password changes, the new password will be in effect the next time you attempt to sign in.

2.5 Deck management

The Collector app provides as useful way to manage your cards by storing them into custom created decks. Cards can be placed in multiple decks and they simply offer a way for a collector to organize and couple cards in interesting ways for later viewing/sharing. To view your decks, select the ‘Decks’ option from the home navigation then select any deck from the list to view its cards. To create a new deck click the ‘Add deck’ button and enter an appropriate name and description for this deck. To remove a deck, click the ‘Remove deck’ button. Lastly, to add cards to a deck, select the deck and click ‘Add cards’, you will be prompted to select cards you want to add to this deck which will be saved once completed.

3.0 Technical notes

3.1 Server Installation

Setting up the database
The application assumes the use of the MySQL RDBMS. First start the MySQL service and connect to your MySQL server:

```
sudo service mysql start
mysql –u yourusername –p
```

The application uses the ‘collectordb’ database so next create this database in your MySQL server by executing the following query:

```
CREATE DATABASE collectordb;
```

Next we will create the tables and populate them. To do this you will need to execute the ‘migration.sql’ script inside the install directory:

```
mysql –u yourusername –p –h localhost collectordb < migration.sql
```

Setting up the server
This project assumes the use of the GlassFish 4.1 application server. The application uses non-default jdbc-connection pools and jdbc-resources and as such you will need to import the project GlassFish domain configurations. To do this, copy the ‘domain.xml’ file in the install directory into:

```
<GlassFish install Directory>/glassfish/domains/<your domain>/config/
```

Next, you will need to change the jdbc-resource username and password parameters. This can be done through the admin console by going to
http://localhost:33648 and selecting JDBC -> JDBC Resources -> jdbc/mysql from the common tasks navigation. There you will need to change username and password parameter values to your MySQL server username and password.

Setting up the server
Lately, you will need to open the project in NetBeans and deploy it to the application server. To do this, open NetBeans and open the ‘collector-server’ project from the menus. Next start the GlassFish application server and then deploy the application by right clicking the ‘collector-server’ project and selecting deploy.

3.2 App Installation

Importing the project
Before the app can be run, you must first import the app project into android studio. The project directory is ‘Collector-Mobile’ and to import the project in android studio select from the menu File -> Open and select the Collector-Mobile/app directory.

Running the App
Once the app project has been imported into android studio you can run the app. Select the project and from the ‘Run’ menu select ‘Run app’. Select an appropriate emulator or device to install the app from, please note the minimum SDK version is 4.0.3 (IcecreamSandwhich). Before attempting to run the app, please ensure the server is running including MySQL. Lastly, to test the app follow the user guide instructions in section 2.

3.3 Collecting data

Another component in this project is a web scraper that was used to collect card data particularly, images, names etc. To quickly achieve this PHP was used to scrape card data from a Pokemon card database: http://www.pokemon.com/us/pokemon-tcg/pokemon-cards/. Card images and names were extracted from searches results and the card images were downloaded to a local directory. Additionally, the paths for the cards were stored in the database ‘cards’ table and once card information was successfully extracted these paths and other card data were added to the database. The data collected consisted of 717 images from 11 different categories/card types.

3.4 Class Structure/UML

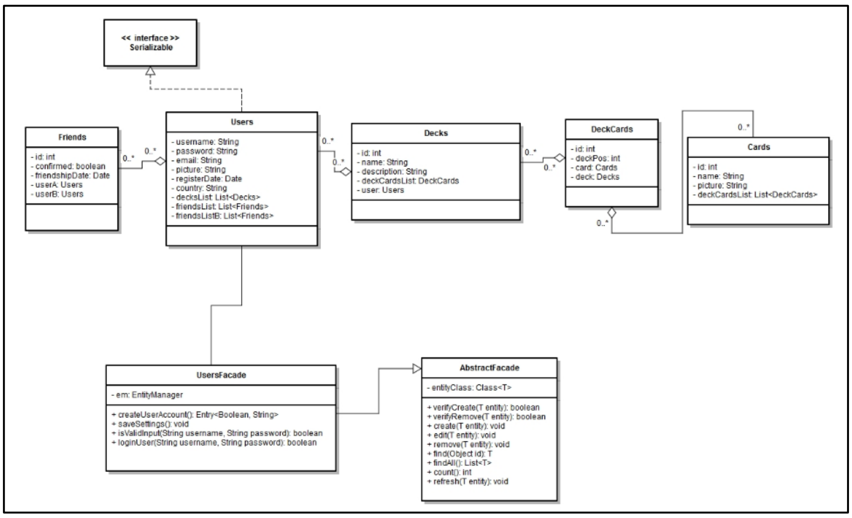


Figure 3: Server persistence entity structure

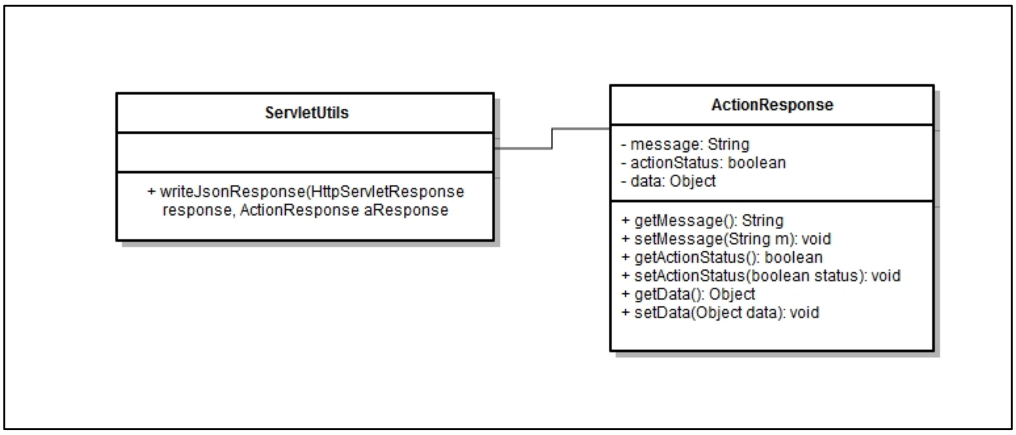

ServletUtils
+ writeJsonResponse(HttpServletResponse response, ActionResponse aResponse
ActionResponse
- message: String
- actionStatus: boolean
- data: Object
+ getMessage(): String
+ setMessage(String m): void
+ getActionStatus(): boolean
+ setActionStatus(boolean status): void
+ getData(): Object
+ setData(Object data): void

Figure 4: Server JSON communication structure

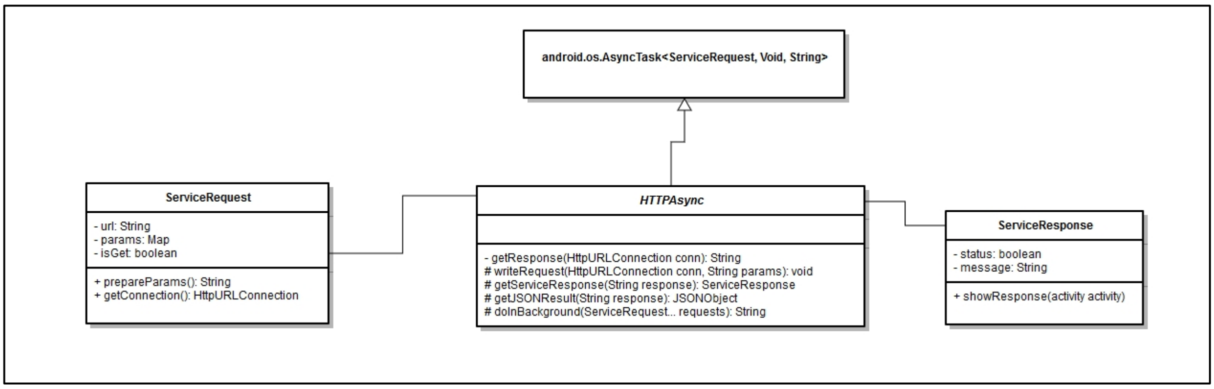

android.os.AsyncTask<ServiceRequest, Void, String>
ServiceRequest
- url: String
- params: Map
- isGet: boolean
+ prepareParams(): String
+ getConnection(): HttpURLConnection
HTTPAsync
- getResponse(HttpURLConnection conn): String
# writeRequest(HttpURLConnection conn, String params): void
# getServiceResponse(String response): ServiceResponse
# getJSONResult(String response): JSONObject
# doInBackground(ServiceRequest... requests): String
ServiceResponse
- status: boolean
- message: String
+ showResponse(activity activity)

Figure 5: Client/App HTTP communication structure