GitHub Username: J-Henrique (https://github.com/J-Henrique)

GoT Collection

Description

The Game of Thrones saga has a huge number of characters, cultures variety and others informations; and be considered an extremely rich fictional universe. However, it is not a simple task to remember basic information about characters, that sometimes can cause confusion throughout the story.
This app aims to bring specific characters informations, in order to simplify quick consultations in a organized way; thus providing a much richer fans experience.

For whom this app was made

This app was developed for Game of Thrones saga fans and readers.

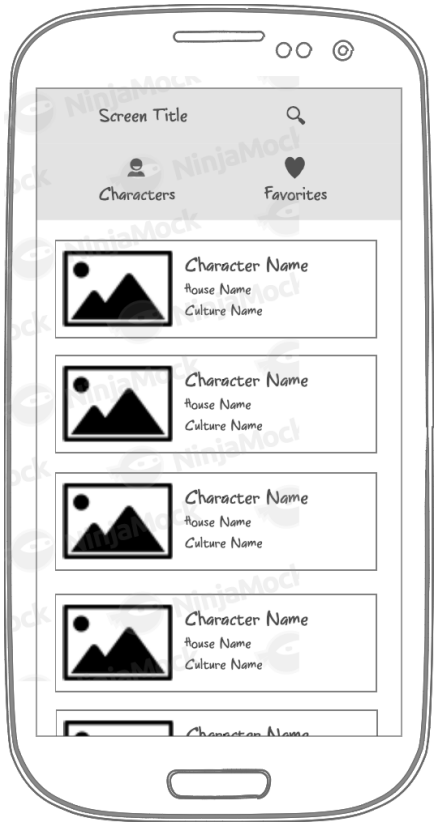
Features

This app brings as main features:

- Specific character searching
- Provide details, like books participation, which house it belongs or culture definition
- Build a favorite list of characters
- Display favorites characters on a Home Screen widget
- Display characters specific informations easily, by clicking on widget list item

UI Mock

Screen 1 - MainActivity

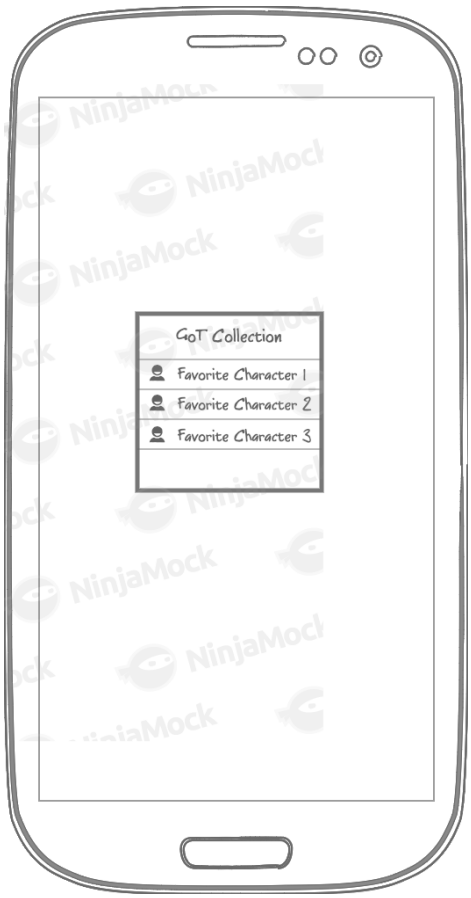


This is screen is responsible for displaying characters cards, queries result and characters that was previously marked as favorite.

Screen 2 - DetailsActivity

Characters details listing screen.

Screen 3 - Widget

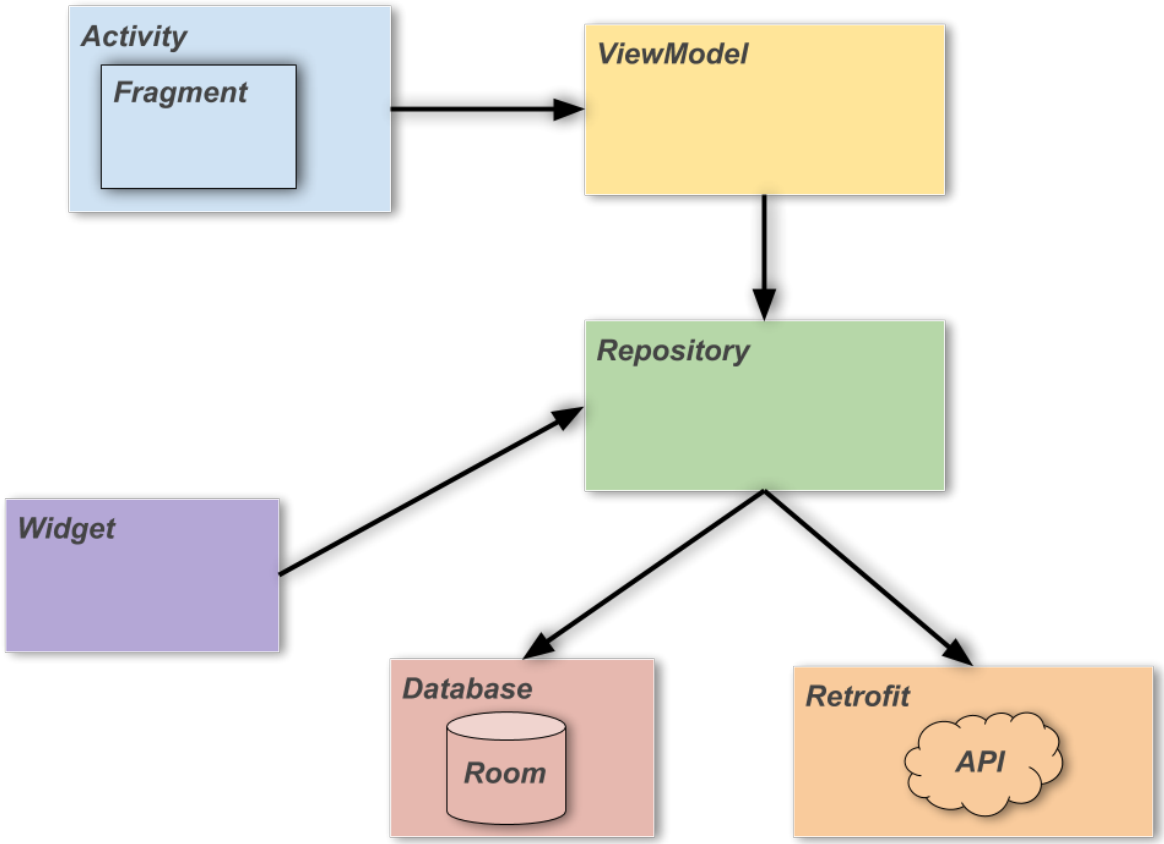


Home Screen widget, displaying the favorites characters.

Key considerations

Data persisting schema.

The characters details informations can be accessed by: API service consumption or local database queries.
It will persist only characters previously marked as favorite, through Room library and presented by MVVM pattern. That way, by clicking on “favorite” button the data will be persisted on disk; thus available for App or Widget search.
Classes relationships and database accessing are demonstrated on the above diagram:



UX basics.

The very simple user experience scenario can be defined as:

- App is opened by the user

- A character search is made and the result is displayed
- By clicking on resulting character card, a transition to details screen occurs
- Clicking on details screen Floating Button, this character is marked as favorite
- If Home button is pressed, considering that a widget was previously added on Home screen, the chosen character should appears along the others favorites ones
- By clicking in a given widget list item, again the details screen is displayed

Utilized libraries.

The following libraries will be used:

- **Picasso**, for web picture loading and caching
- **Retrofit**, API consumption and requests management
- **Room**, local database persisting schema
- **Parceler**, for serializing and deserializing shared objects between activities

Firebase services used.

It will be needed 2 Firebase services:

- **Firebase Google Analytics**, in order to log basic users informations and main searches it made
- **Firebase Google Admob**, to show ads on free version app

Next Steps: Building project main tasks

Task 1: Project Setup

Add necessary project dependencies (Retrofit, Picasso, Firebase, Material Design) and enable Data Binding.

Task 2: Model mapping

Add an object model to map characters informations returned from service (https://api.got.show/doc/). The following object model is currently being returned from API: https://github.com/Rostlab/JS16_ProjectA/blob/master/app/models/character.js

Task 3: Add a persisting data schema

Add persisting model structure:

- Database class
- Repository class, dedicated to database access

- DAO operations interface (Data listing and insert)
- Update created model with @Entity notation

Task 4: Repository class configurations

API endpoints mapping:

- Return all characters (https://api.got.show/api/characters/)
- Return a specific character based on a given parameter (https://api.got.show/api/characters/:name)

Database queries:

- Return all data currently persisted on characters table
- Insert data

PS: All this functions should be exposed for consumption through a ViewModel pattern class

Task 5: MainActivity layout implementation

Add the following components on MainActivity:

- TopBar with a search button
- TabLayout with two options (Characters and Favorites)
- RecyclerView for characters cards listing (create a card layout)

Task 6: ViewModel creation

Create a ViewModel class for MainActivity. This class should:

- Retrieve API and database data from Repository class, that will be utilized as RecyclerView dataset
- Expose insert database function, that will be invoked from Activity

Task 7: UI Data Binding

Activity functions implementations:

- Set data returned from API to characters RecyclerView list
- Set data returned from local database on characters RecyclerView list
- Enable specific character search

PS: API calls should be made asynchronously, in order to not block UI Main Thread

Task 8: Add DetailsActivity layout and ViewModel

Add a layout activity that will display selected character details:

- Enable data to be loaded by Data Binding
- Display selected character picture
- Display it details
- Add a "favorite" button, to save data on local database (it may call ViewModel function)

Task 9: MainActivity and DetailsActivity transition

Provide transitions between activities:

- By clicking on character card, it should navigate to details screen (the current object should be serialized before transition)
- Implement a basic transition animation between activities

Task 10: Firebase Services

- Configure Analytics service to save basic user data when app is opened
- Configure Admob to show an ad on characters details screen

Task 11: Free and Paid Flavors

Enable a free and a paid flavors app.
The free flavor should not display ads to the user (remove unnecessary dependencies).

Task 12: Accessibility

Provide app content descriptions for users with visual limitations.

Task 13: Favorite characters Widget

Widget implementation:

- The widget will contain a character list to show favorite characters cards
- The list may be populated by Repository class calls
- By clicking on widget list item, a details screen may be displayed
- Update details activity favorite button to send a broadcast when it is pressed, in order to update widget list items