Table of Contents

GitHub Username: [dnkaratzas]

StarWars Pedia

All the Star Wars data you've ever wanted

Description

StarWars Pedia allows users to search a huge amount of information on all sorts of subjects within the official Star Wars universe ranging from Planets, Spaceships, Vehicles, People, Films and Species.

Intended User

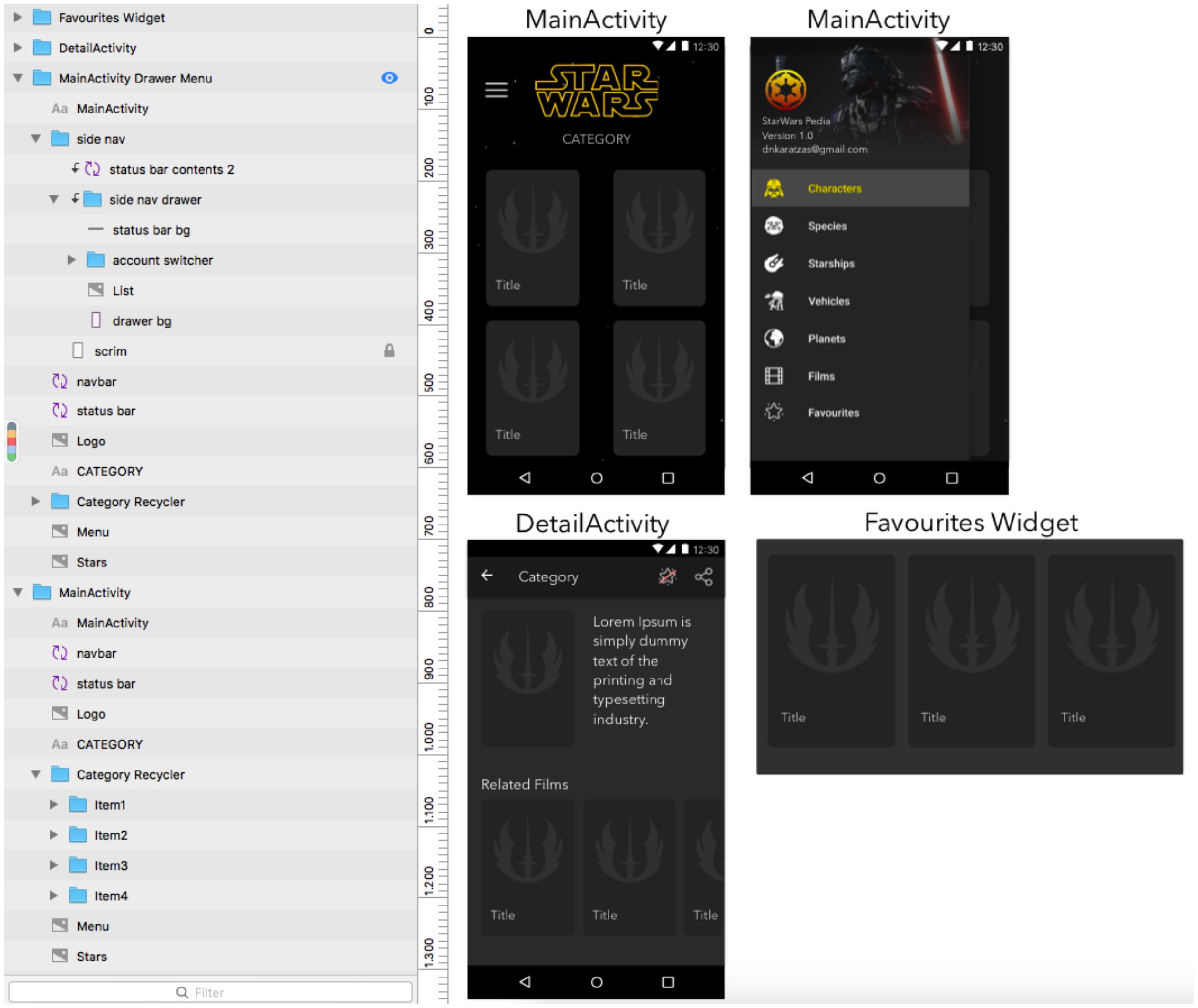
This is app intended to be used by StarWars fans or anybody who is willing to explore the Star Wars universe.

Features

- Explore all the basic Star Wars category items such as Planets, Spaceships, Vehicles, People, Films and Species
- Access in extra details for any of the category items
- Share the item details with others
- Ability to add any category item to favorites list
- Homescreen widget including all of your favorite list items

User Interface Mocks

All Screens



Key Considerations

How will your app handle data persistence?

StarWars data are retrieved using SWAPI (The Star Wars API). The app will save the favorite items data (title and image) internally using SQLite and a Content Provider.

Describe any edge or corner cases in the UX.

- In case of any failure, empty list content or lack of internet connection, the user is informed with an appropriate status message;
- If the user unfavorites an item, this item will be removed from the favorite list immediately.
- When the app will request something from the server the animated stars will animate faster!

Describe any libraries you'll be using and share your reasoning for including them.

- Firebase Storage to store and retrieve the StarWars data images since the API doesn't provide image resources.
- Glide for image loading and caching.
- Butter Knife for view binding.
- Timber for Android Logging
- LeakCanary to detect and solve memory leaks
- Apollo GraphQL client to work with GraphCMS API:
  - GraphCMS ported the popular Star Wars API (swapi.co - by Paul Hallett) to GraphCMS to demonstrate the flexibility of its generated GraphQL API: https://api.graphcms.com/simple/v1/swapi
  - Will use the power of GraphQL because the SWAPI endpoint have complex repeatable retrievals having as result performance issues for the app and the user experience. With GraphQL, data can be delivered in a single request as a package rather than through multiple calls.

Describe how you will implement Google Play Services or other external services.

- Firebase Storage to store image assets
- Google AdMod to monetize the app by displaying in-app ads

Next Steps: Required Tasks

Task 1: Project Setup

- Create the project and add it to git repository
- Set up the libraries
- Structure the project packages: view, data, model, libs, etc...
- Add the necessary ProGuard rules
- Generate a KeyStore and add it in the project's root directory
- Add in Gradle the signing configuration with the KeyStore's location, password and the key alias name
- Put the GraphQL queries and schema to the project

Task 2: ApiManager

- ApiManager must be a singleton
- Implement the calls that will use a Loader to load and move the data to the views
- The loader doesn't need to implement AsyncTaskLoader to query the graphQL server. Apollo-android client uses okhttp client for requesting the queries from the server, so the calls are already asynchronous.
- Create the required models

Task 3: Implement MainActivity

- Build UI
- Implement a navigation drawer to hold the available SWAPI categories
- Create and attach the CategoryFragment or the FavouritesFragment on category selection
- MainActivity needs to listen to Fragment callbacks
- Implement the procedure of fetching the item details from the server and launching the DetailActivity to display them:
    - On fragment item click callback
    - On the widget item selection

Task 4: Implement DetailActivity

- Build UI
- For the selected item, create dynamic RecyclerViews regarding the item lists related to that item
- Add an app bar which will hold the selected category title and the back, favorite and share buttons
- Add the ability to favorite or unfavorite an item
- Implement sharing functionality, making use of intent extras to share the item details
- On related item click, implement the procedure of fetching the item details from the server and relaunch the DetailActivity and display them

Task 5: Implement CategoryFragment

- Fetch the selected category items from the server and display them on a recycler view with a StaggeredGridLayoutManager
- On save and restore instance state, preserve and restore recycler's view data and current position

- Inform the parent Activity on item click or when the loading status change

Task 6: Implement FavouritesFragment

- Retrieve the favorite items from the database by using the content provider and display them on a recycler view with a StaggeredGridLayoutManager
- On save and restore instance state, preserve and restore recycler's view current position
- Inform the parent Activity on item click or when the loading status change

Task 7: Implement feature "add to favorites"

- Create the database contract
- Create the content provider

Task 8: Implement widget for displaying favorite items

- Create widget using GridView layout
- Make use of the content provider to fetch the favorite items from the database
- On the item click, the widget, must provide via an intent extra the selected item's id to the MainActivity