GitHub Username: ferrannp

Show-man

Description

Show-man is a super hero! And it has three main powers: Find any TV show, track your watched episodes and notify you about new ones!

Keep track of the episodes you are Following and how many you have left to be up to date. Also, you can see when next episodes will be aired. On the other hand, if you are not feeling like following a show right now, don't worry! You can set them to Pending and start following them another time. For the most nostalgics, you can also have a record of your already Seen shows.

Intended User

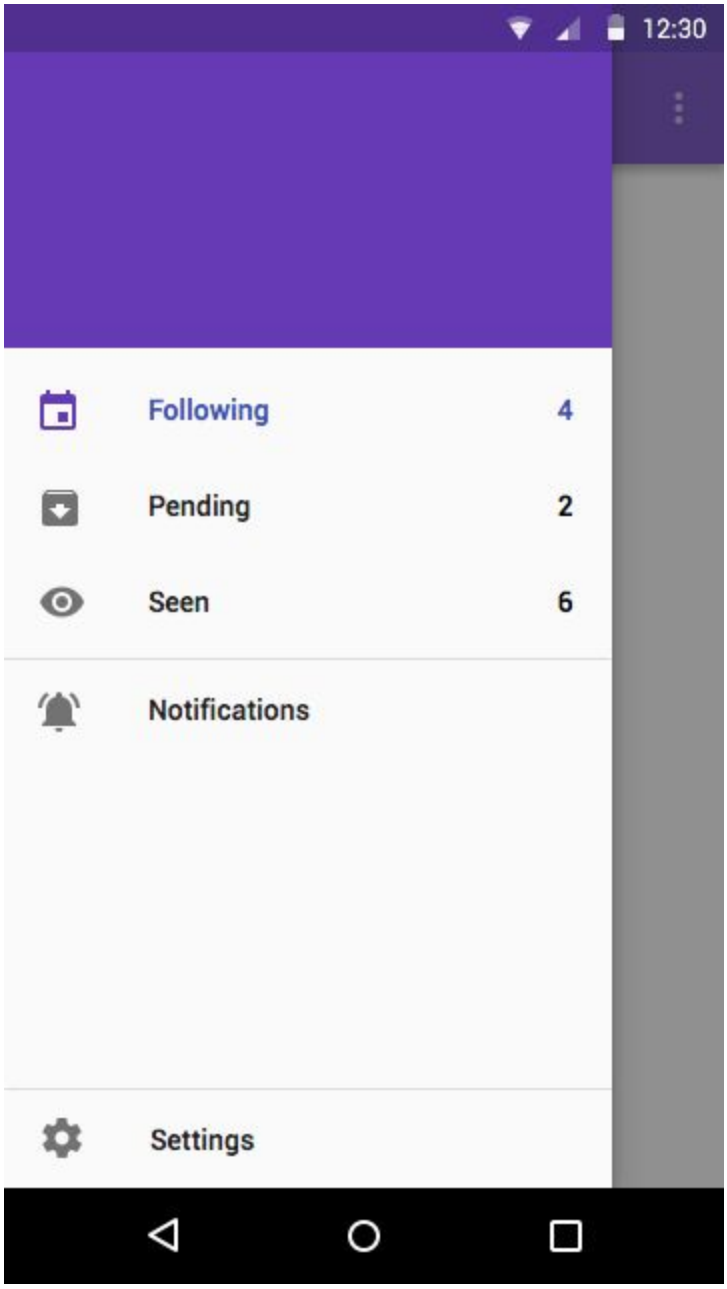
For any TV show fan in the world who wants to keep track of his/her favourite shows!

Features

- Search for TV shows and add them to your collection
- Organize your TV shows as Following, Pending or Seen
- In the Following section, it is possible to see the episodes that are already available and the ones that will be aired in the future
- Collection Widget for your Following shows
- Daily notifications when an episode will be aired and/or it is already available (configurable)
- Share when an episode will be aired or it is already available

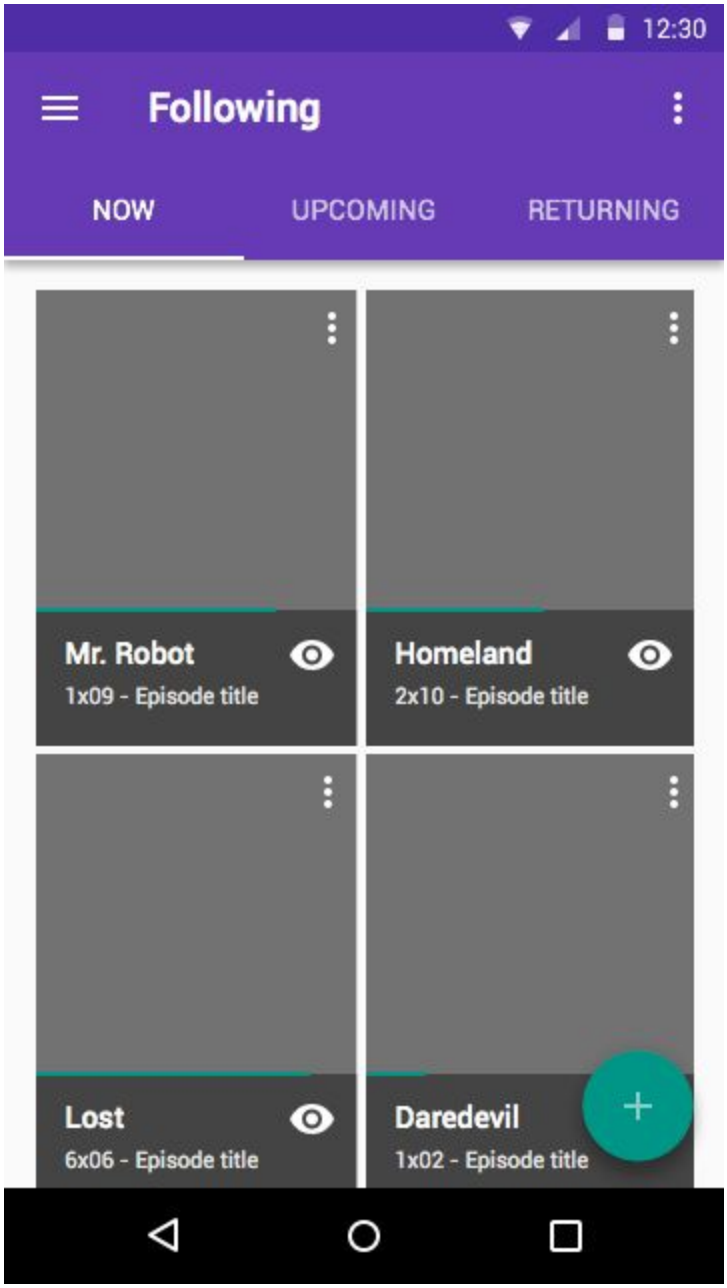
User Interface Mocks

Screen 1 (Navigation Menu)



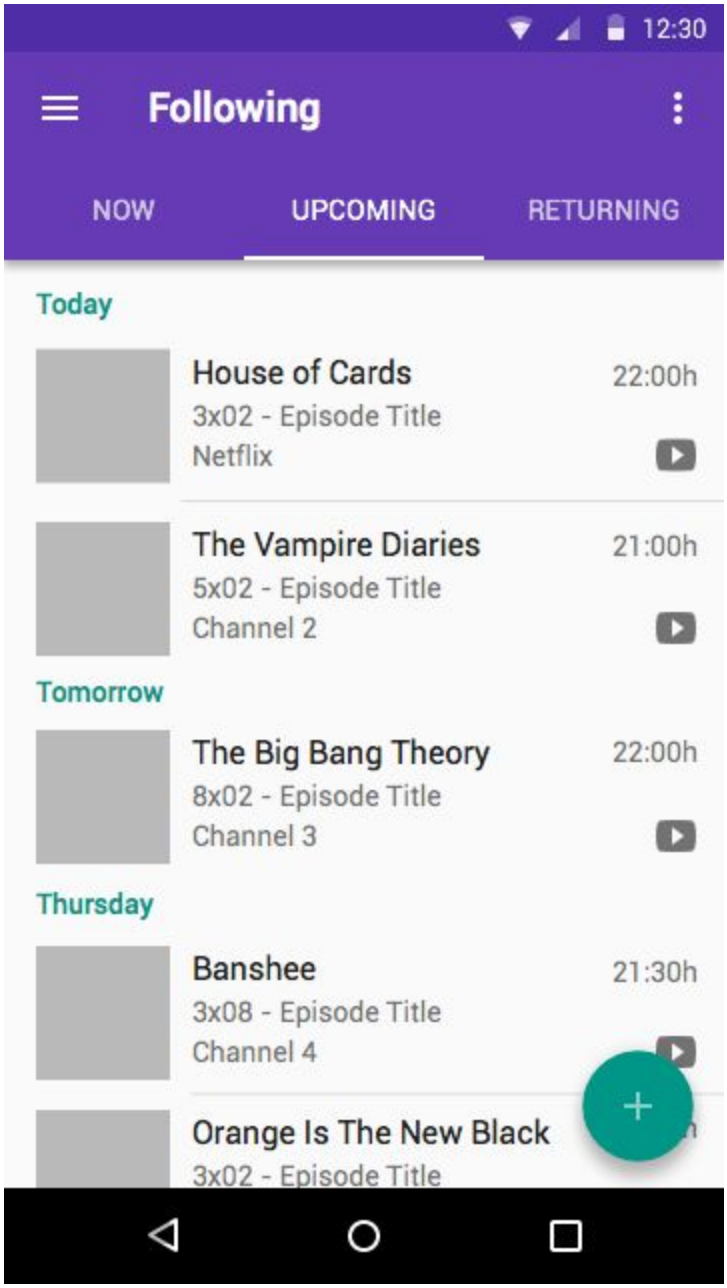
The main menu shows the three categories where a TV show can fit into. The fourth section is the Notifications feature and its configuration. Settings is going to be for other stuff like “Only download images via Wifi”.

Screen 2 (Following - Now)



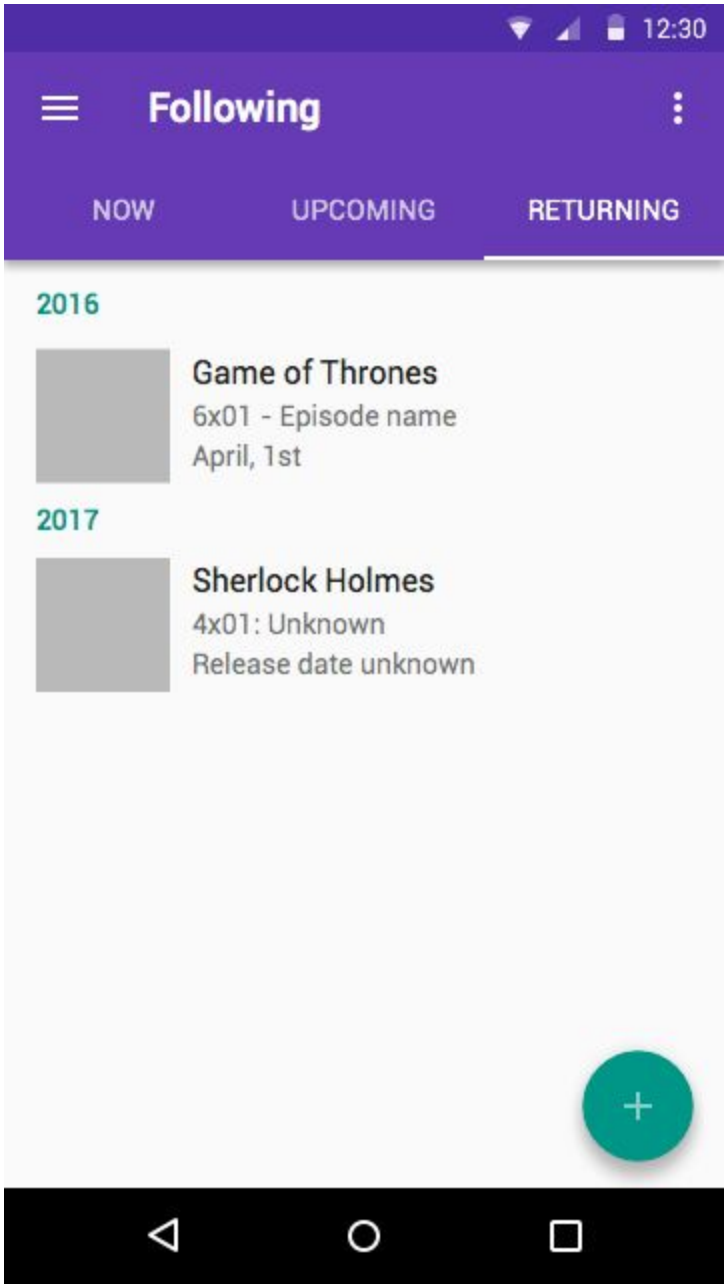
This is the main view when the user opens the app, the screen shows the episodes the user needs to watch next. The user can mark them as watched just by simply clicking into the “eye” icon. The green bar shows how many episodes are left to be up to date. The FAB icon allows adding more shows (see Screen 6).

Screen 3 (Following - Upcoming)



In the next tab of the Following section, the user can see the upcoming shows (episode, day, time and channel). The youtube icon is intended to open the promo of the episode sending an intent to the youtube app (if possible).

Screen 4 (Following - Returning)



Last tab has TV shows that the user is up to date and there are no new seasons on TV. Basically, waiting for new seasons.

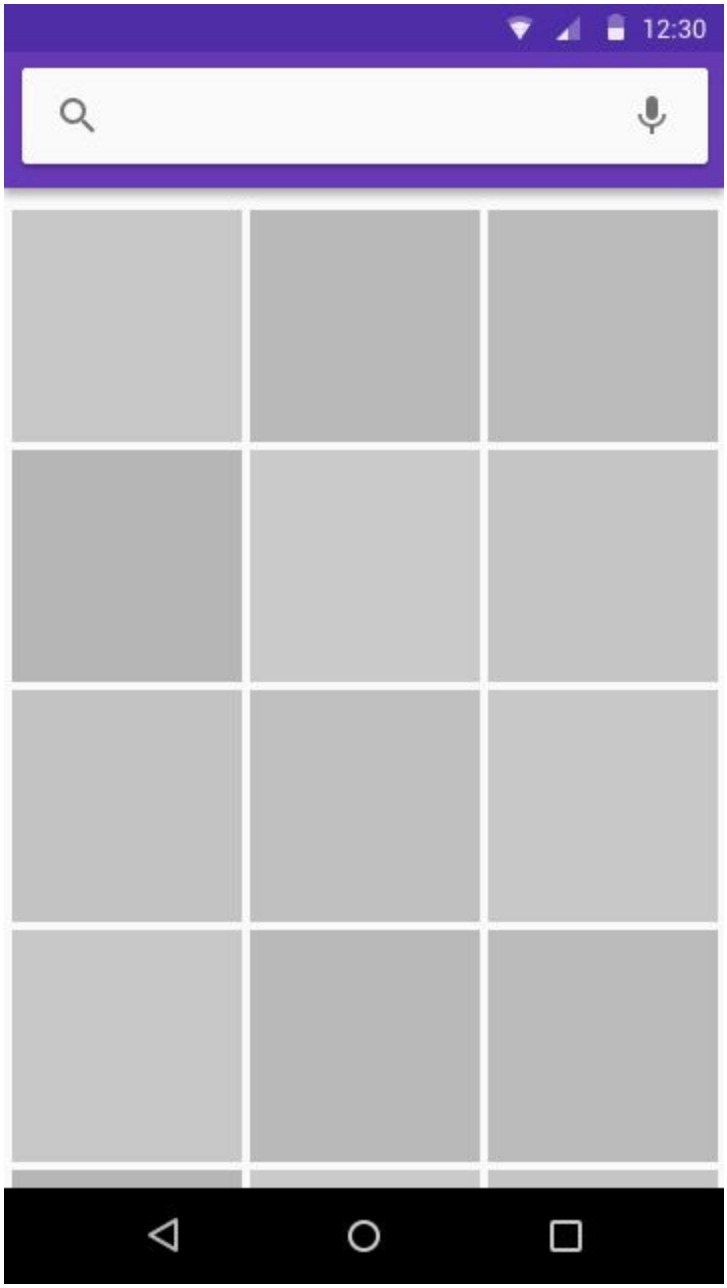
Screen 4 (Pending)

Pending episodes will be a list with the image of the TV show, the title and the last watched episode (similar to the Following - Upcoming screen, see Screen 3)

Screen 5 (Seen)

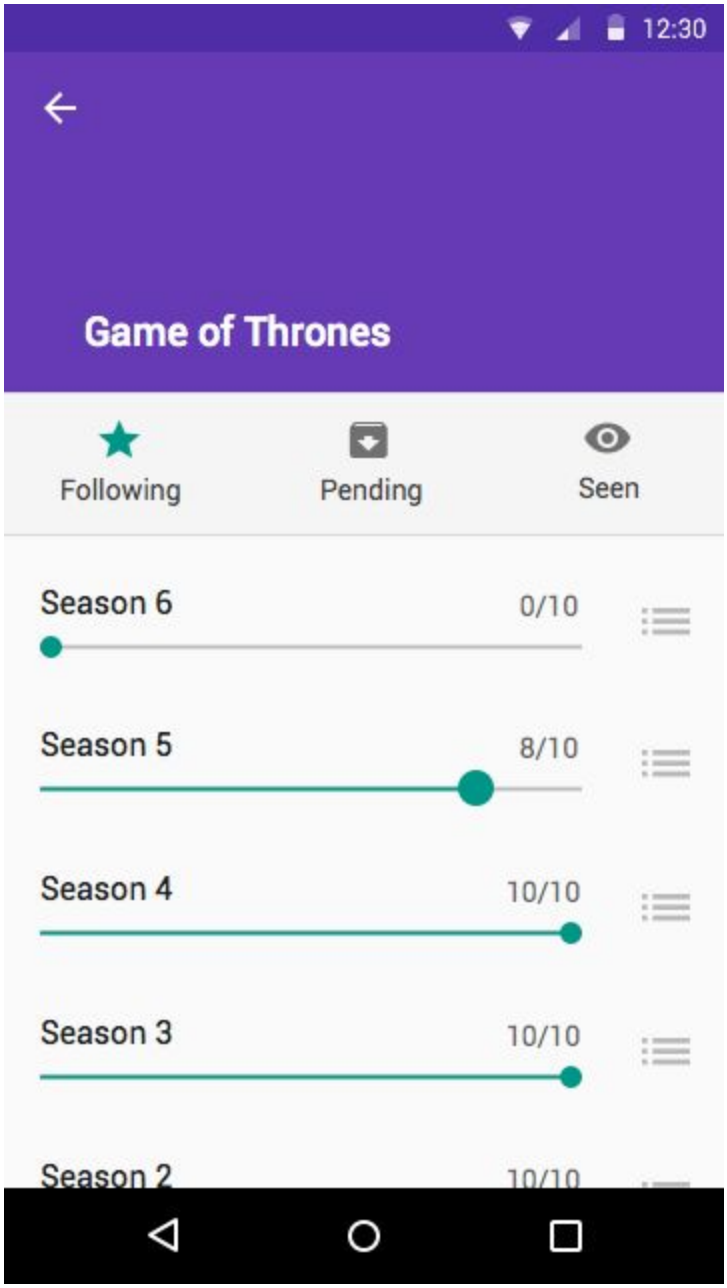
Seen episodes will be a list with the image of the TV show, the title and the ending date (similar to the Following - Upcoming screen, see Screen 3)

Screen 6 (Searching TV shows)



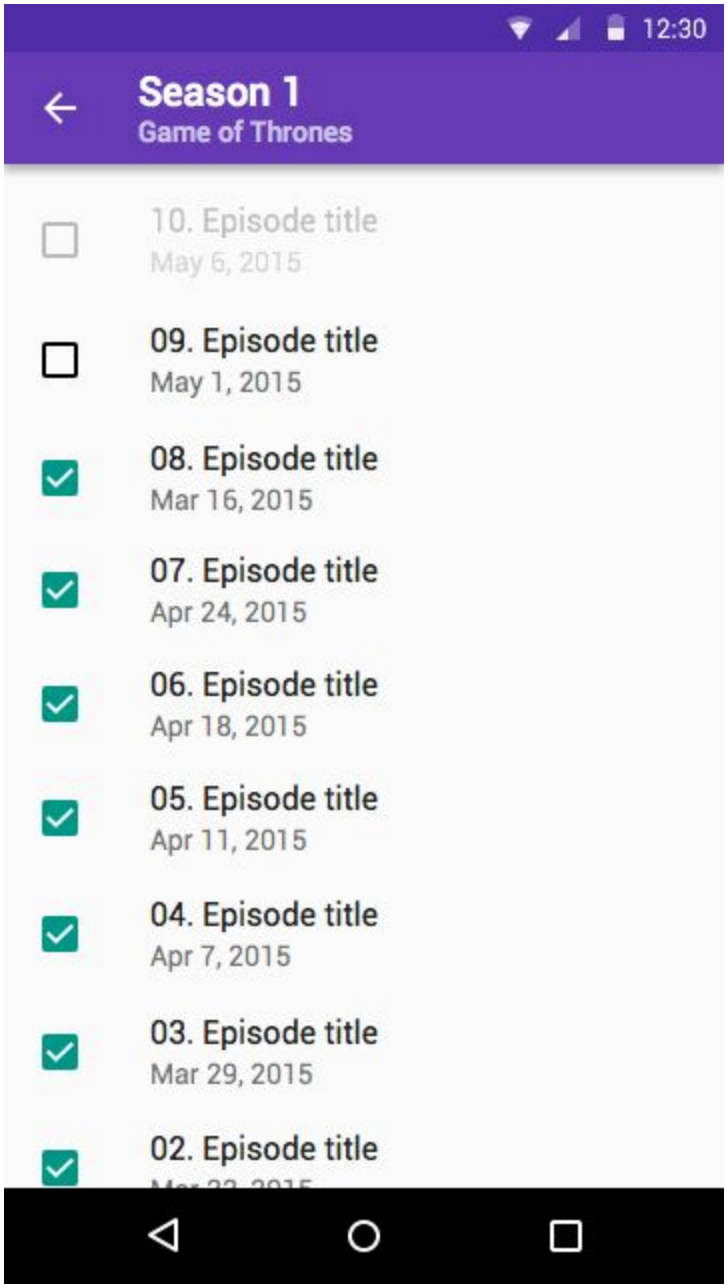
Clicking the FAB button (+) in any of these sections: Following, Pending or Seen, will let the user to add a new TV show to their collection. Initially, we show a grid with most popular and recent shows.

Screen 7 (TV Show Seasons)



Clicking into a TV show that is in the user collection will show the detail page (with a big stunning image that will collapse when scrolling). The page allows the user to change the state (Following, Pending an Seen) of the show in a very easy way. Also, the sliders can help the user to mark watched episodes quickly (of all seasons). If the user wants more details for marking watched episodes, he/she can click the show list icon (see Screen 8).

Screen 8 (TV Show Episodes)



Another way to mark watched episodes. Episodes that are not aired yet will be disabled.

Key Considerations

How will your app handle data persistence?

TV Shows data will come from an external API. The app will save some data internally using SQLite and a Content Provider. Settings will be saved using SharedPreferences.

Describe any corner cases in the UX.

- A show can only be set to Seen if all the seasons have been aired (in other words, the show is finished or cancelled)
- The user can add a show from any of these screens: Following, Pending or Seen
- The user should be able to move a show to Following, Pending or Seen at any time
- If there are no TV Shows in some section, we should show an empty screen

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

- Picasso (http://square.github.io/picasso/): I will need a good one to handle loading a lot of images (see mockups) and caching them
- Retrofit (http://square.github.io/retrofit): For getting data from an external API
- Butterknife (http://jakewharton.github.io/butterknife): Because I will be very happy if I don't have to use findViewById (including casting).
- Schematic (https://github.com/SimonVT/schematic): I liked the presentation of this on the Nanodegree. I will use it to help me on building the Content Provider.
- Leakcanary (https://github.com/square/leakcanary): I will use this one to see if there are any memory leaks

Next Steps: Required Tasks

Task 1: Project Setup

- Set up libraries
- Structure the project in packages: view, data, model, rest, etc. Separate pure Java classes in separate packages

Task 2: Gather data using trakt.tv API

We need a way to gather all the TV shows data from an external API.

- Write a helper class using Retrofit
- Use a Service (that can be bind to an activity) to do the job

Task 3: Implement UI for the Search Activity

- Build UI for the Search Activity & adding a new TV show
- Handle the logic of showing most popular and recent shows

Task 4: Implement UI for the main Activity and the Fragments

- Build UI for the MainActivity & Following Tabs
- Build UI for the Pending and Seen fragments
- Build UI for the TV Show detail page

Task 5: Data persistence

We need to store the following:

- Basic information regarding TV shows: api ids, title, seasons & list of episodes.
- To the above list of episodes, mark the ones the user has watched or has not
- Add a state to each TV show: Following, Pending or Seen

Tasks:

- Finish to decide the design of the database (what we store and what we don't). Keep in mind that only what we store will be available when we are offline

- Create the database and the Content Provider using Schematic
- Use Loaders to access the data

Task 6: Pull data

- Implement a SyncAdapter to update stored information if necessary (for example a new episode for the season) pulling data daily

Task 7: Collection Widget for Following Shows

- Create a widget that shows a list of following TV shows that have available episodes
- Make each item of the list to have an icon that allows the user to mark the episode as watch right into the widget

Task 8: Implement Google Analytics

- Track crashes and other stuff Analytics give us for free
- Track the amount of series the user has in Following, Pending and Seen
- Track if users uses Notifications (if they are implemented)
- Track if users shares something (if it is implemented)
- Track more stuff?

Task 9: Notifications (Optional)

Depends on how long it takes to finish all other tasks, Notifications might not be implemented for the Capstone version of this app. This would fit in the category of “Exceeds specifications”.

- Build UI for the Notifications section. Selected options will be saved using SharedPreferences
- User receives a notification when an episode will be aired (time can be configured)
- User receives a notifications when an episode has already been aired (time can be configured)

Task 10: ShareActionProvider (Optional)

Depends on how long it takes to finish all other tasks, ShareActionProvider might not be implemented for the Capstone version of this app. This would fit in the category of “Exceeds specifications”.

- Share available episode (tell a friend that is already there!)
- Share date of an episode that will be aired (tell a friend that it will be there soon!)