GitHub Username: michaeljordanr

SongStats

Description

SongStats register information about activities when synchronized with a Spotify account, information like the most listened playlists, tracks and artists.
All data will be retrieve from the Spotify Web API, for that the app will ask the information account for the user

Intended User

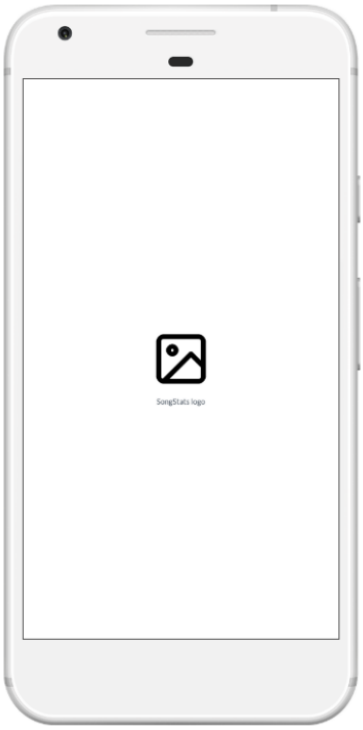
People who love listen to music on Spotify and want to know more about their history using the app.

Features

- Save information about playlists, tracks and artists played by the user
- Show information through graphs
- Filter by time period

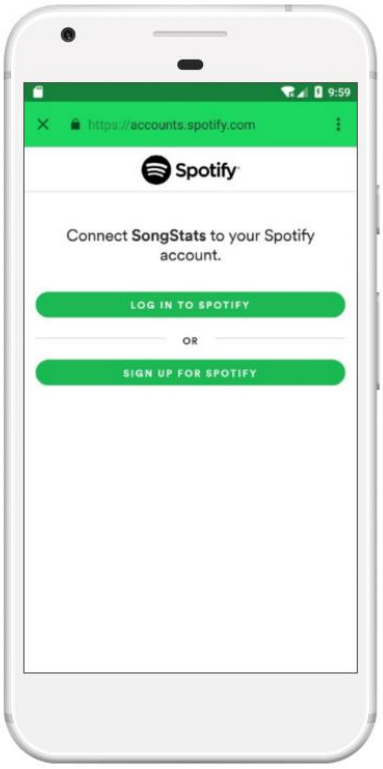
User Interface Mocks

Screen 1



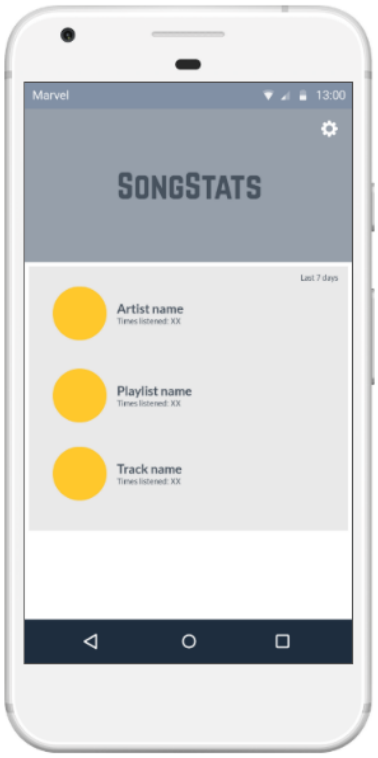
Splash screen with a 2 seconds delay. Image logo in the center of the screen

Screen 1.1



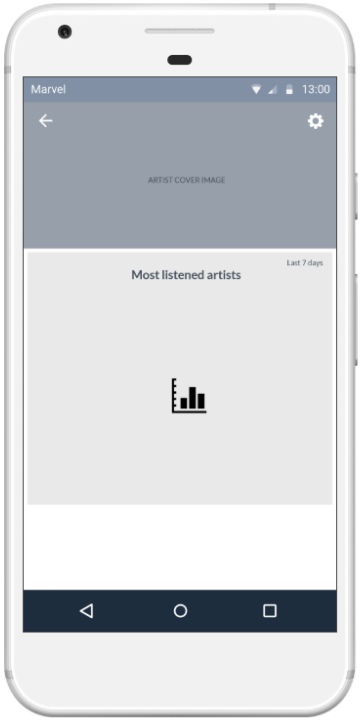
Case the user hasn't logged yet, the login page will pop up.

Screen 2



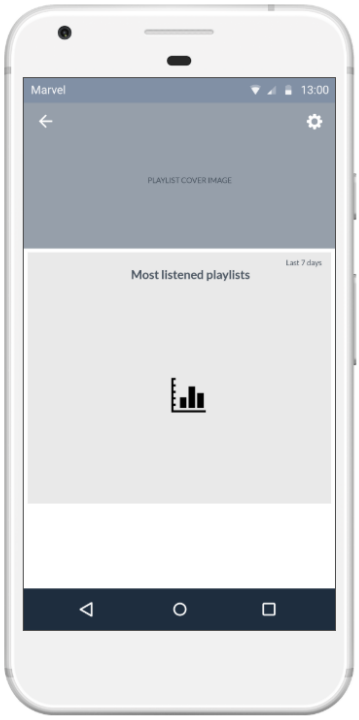
Dashboard screen with a button that redirect to settings screen on the top. On the card under the logo it shows some information about statistics of Artists, Playlists and Tracks. Each item redirects to the respective screen (Artists, Playlists and Tracks statistics).

Screen 3



Artist statistics screen shows a graph with information about the most listened artists on Spotify. It's possible filter by date hitting the button on the corner.

Screen 4

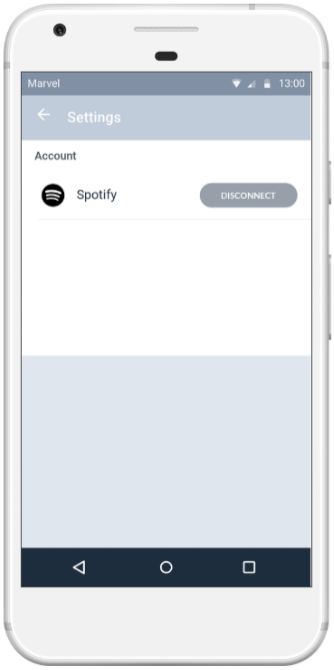


It’s the same functionality described on Screen 3 but with playlists information.

Screen 5

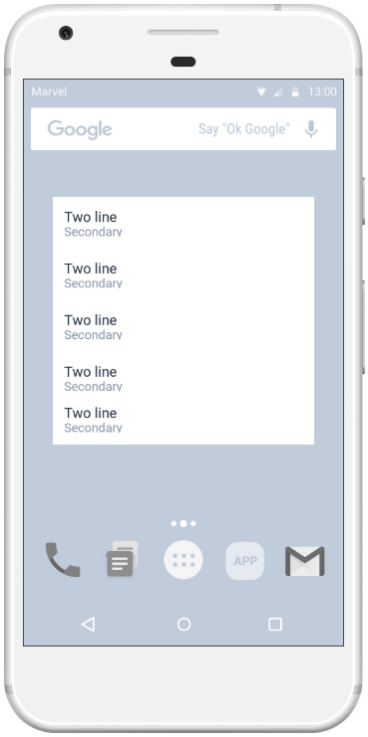
It’s the same functionality described on Screen 3 and 4 but with tracks information.

Screen 6



On Settings screen it’s possible to disconnect the Spotify account from the app hitting the button DISCONNECT

Screen 7



Widget that will show a list with the tracks most listened and the number for hits

Key Considerations

How will your app handle data persistence?

The data will be persisted using Room from Architecture Components Library
Use of LiveData with View Model for Room implementation

Describe any edge or corner cases in the UX.

Error handling for no internet and no data.
Label the views with `android:contentDescription` in order to make the app more accessible
Supporting for RTL

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

Picasso to handle the loading and caching of images.
Architecture components to handle persistence and avoid memory leaks in the configuration changes

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

All the songs data will be consumed from Spotify Web API, using Retrofit library.
Google Play services: Analytics and AdMobs

Resources

All the UI texts in strings.xml
Use qualifiers to handle different types of screen (images and layout)
A centralized theme for use in the entire app

Next Steps: Required Tasks

Task 1: Project Setup

- Create the project on Android Studio 3.1.3 and Gradle 3.1.3
- Configure libraries and SDK
    - compileSdkVersion = 28
    - minSdkVersion = 21
    - targetSdkVersion = 28
    - implementation 'com.android.support:appcompat-v7:27.0.2'
    - implementation 'com.android.support:design:27.0.2'
    - implementation 'com.android.support.constraint:constraint-layout:1.1.2'
    - implementation 'com.squareup.okhttp3:okhttp:3.9.1'
    - implementation 'com.spotify.android:auth:1.1.0'
    - implementation 'android.arch.lifecycle:runtime:1.1.1'
    - implementation 'android.arch.lifecycle:extensions:1.1.1'
    - implementation 'android.arch.lifecycle:compiler:1.1.1'
    - annotationProcessor 'android.arch.lifecycle:compiler:1.1.1'
    - implementation 'android.arch.persistence.room:runtime:1.1.1'
    - annotationProcessor 'android.arch.persistence.room:compiler:1.1.1'
    - testImplementation 'junit:junit:4.12'
    - androidTestImplementation 'com.android.support.test:runner:1.0.2'
    - androidTestImplementation
    - implementation’com.android.support.test.espresso:espresso-core:3.0.2'
    - implementation 'com.google.android.gms:play-services-analytics:16.0.1'
    - implementation 'com.google.android.gms:play-services-ads:16.0.1'
- Configure config files for Spotify Web API
- Configure .gitignore
- Configure the project to use Java as the programming language

Task 2: Implement UI for Each Activity and Fragment

Implement the Activities and Fragments using Java as the programming language

- Build UI for SplashScreenActivity
- Build UI for DashBoardActivity
- Build UI for ArtistFragment
- Build UI for PlaylistFragment
- Build UI for TrackFragment
- Build UI for SettingsActivity

Task 3: Implement Spotify Login Web API

Implement the methods to get the user credentials and get the OAuth access token from the API

- Create a class with the endpoints and require parameters
- Implement some require tags in AndroidManifest.xml
- Create ViewModel and Activity/Fragment for the Authentication

Task 4: Implement URIs for Artists, Playlists and Tracks

Create all the files for handle the APIs return

- Implement de URIs
- Create model classes
- Create the repository classes and implement the code that get the responses from the APIs

Task 5: Implement Java Code for Activities and Fragments

Implement the activities and fragments

- Inflate the views
- Bind the data

Task 6: Implement Java Code for Settings Activity

Implement Settings Activity where the user will be able to unlink their Spotify account

- Inflate the view
- Clear the user data

Task 7: Implement Widget

Implement a widget that will show the most listened track list

- Build UI
- Build java classes (Intent Service, Async Tasks)
- Inflate the view with the data