CAPSTONE PROJECT : Shift•t - Stage ONE

GitHub Username: carlosdesiderio

Description

Shift•t enriches the perception of our direct environment: our neighbourhood.

Shift•t shows tweets and twitter trends based on the user current location. By showing this information, Shift•t let the user know what people is talking about in the user's near vicinity.

Shift•t provides another layer of information to the perception of our neighbourhood.

Intended User

The neighbour
The intended user for the application is any one that would like to know more about what is going on in their area.

On top of what people can gather from other channels, Shift*t will allow the user to see what is happening and what the interests are in their area, based on other people interactions in Twitter.

Features

List the main features of your app. For example:

- Determines user location
- Shows Twitter trends in the user's area.
- Shows neighbourhoods around the user's location with Twitter activity
- Shows Tweets in a selected neighbourhood
- Shows Tweets related to a selected trend in the user's area

User Experience

As the application opens, a map is shown with the current device location. Users can select amongst two provided actions at the floating button: either getting the local trends or getting nearby areas where some Twitter activity has taken place.

Neighbourhood Journey

When the user selects to '*locate*' tweets in the area, a new screen is displayed with a map with a number of polygons drawn on it. The polygons define the different nearby areas where Twitter activity has taken place.

When one of this areas are selected, the app transitions to a new screen with a list of the tweets in that area.

Selecting one of the tweets in the list will take the user to '*detail view*' for the selected tweet.

Trends Journey

When the '*trends*' button is selected at the home screen, a list of the available trends for the define area is shown.

Selecting any of the trends list items will take the user to a different screen where a list of tweets will be shown. Selecting one of the tweets in the list will take the user to '*detail view*' for the selected tweet.

The transition will only take place if there are any tweets in the user's area related with the trend selected. A snackbar will be shown instead when no tweets are available for these parameters.

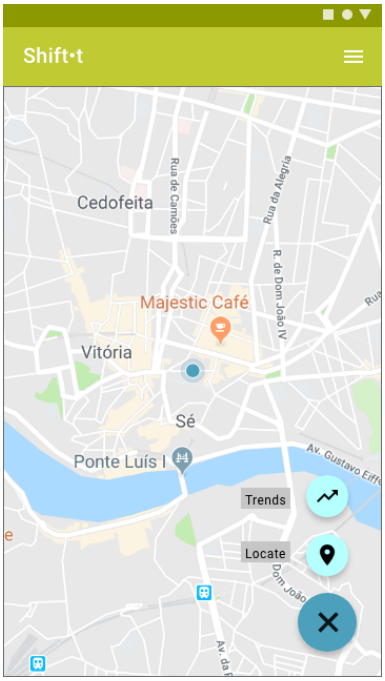
Settings Journey

The applications provides a option menu on the tab bar where the settings could be accessed. Users will be able to change the radius of the tweet search and the search radius unit (kilometers | miles)

User Interface

Phone UI flow

Level 1: Home

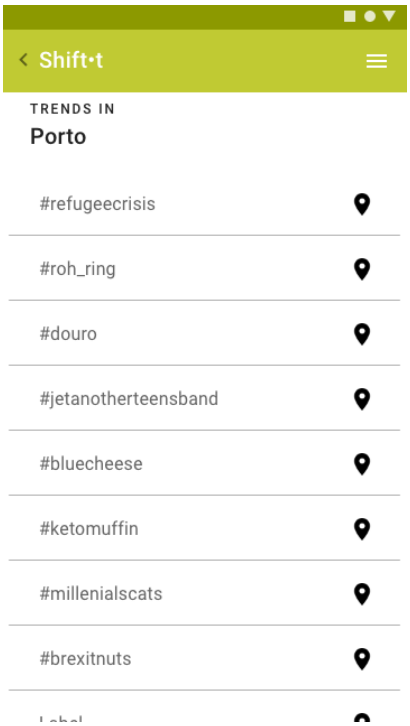


Home screen. Two actions are available:

- Selecting '*trends*' button will show the "local trends screen.
- Selecting '*locate*' will show the '*neighbourghood*' screen

Level 2: Local Trends / Neighbourhoods

Selecting items on either of these two screens will take the user to a tweet list.



Screen 2: local trends

Screen 3 : Neighbourhoods

Level 3 : Tweet List

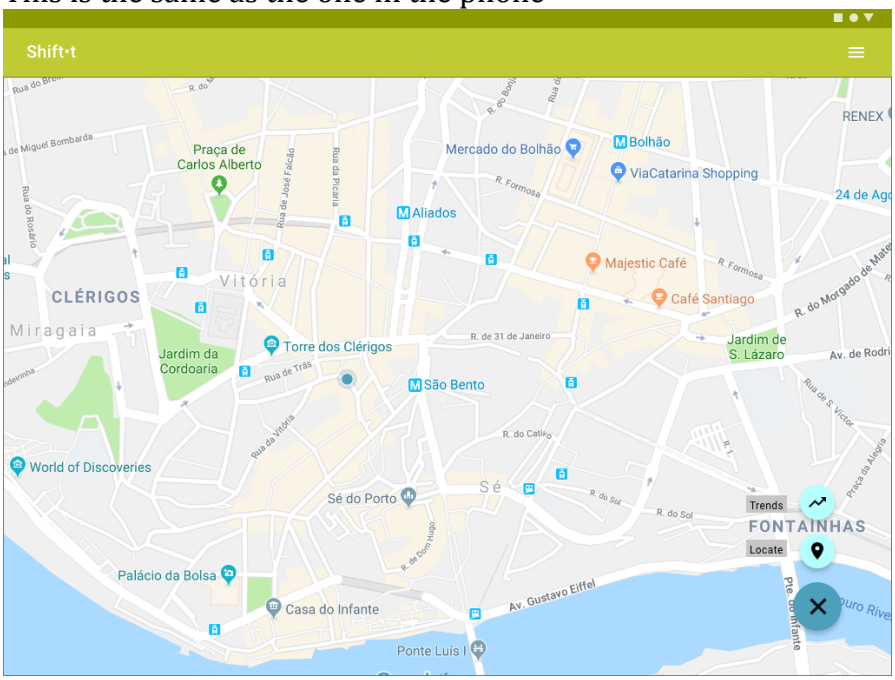
Screen showing a list of tweets based on location and trend (latter only when coming from the trend screen).

Tablet UI Flow

The application has a different interaction pattern on the Tablet devices.

Level 1 : Home

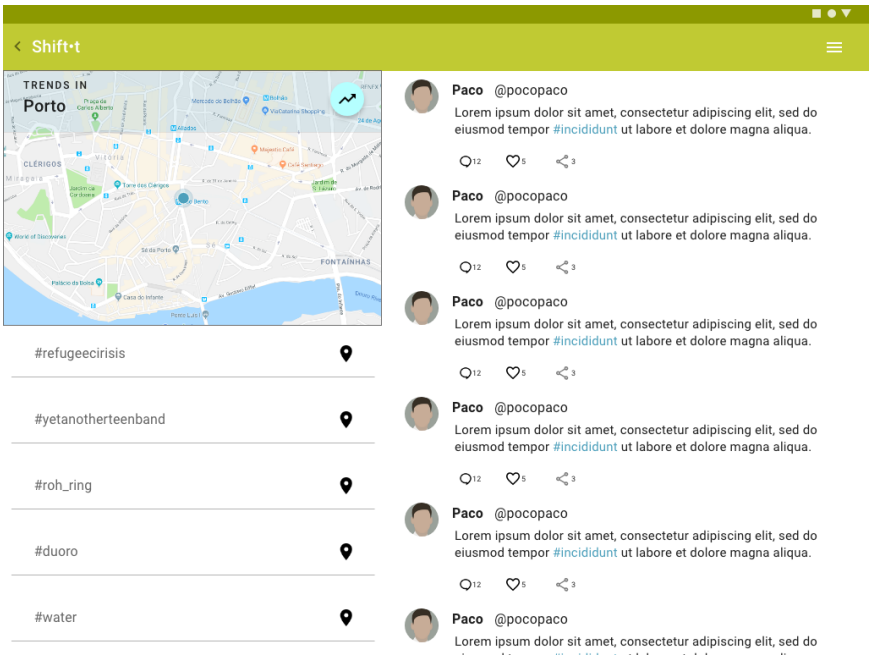
This is the same as the one in the phone



Level 2: Trends

Three-pane layout to show Twitter trends in the area

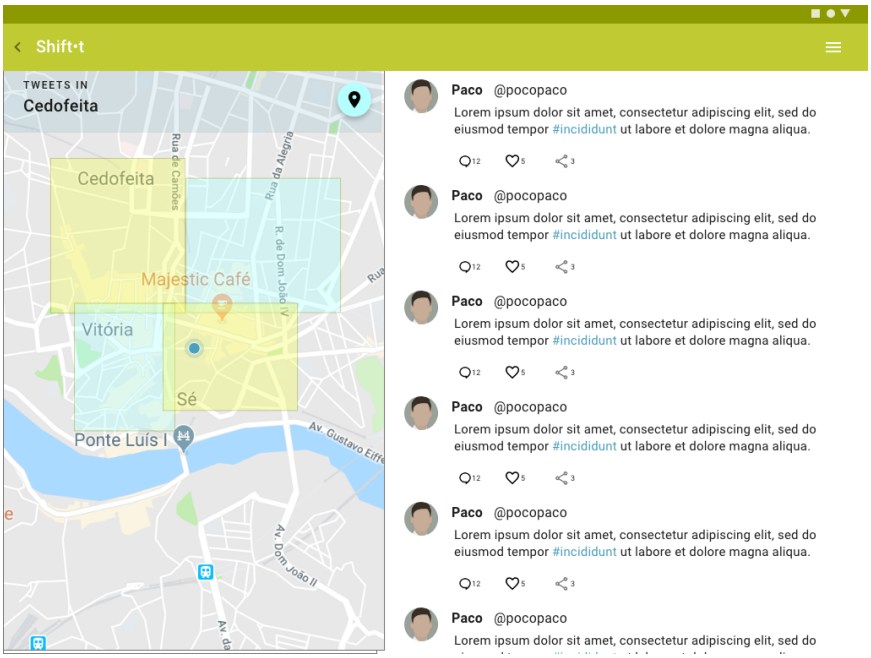
- Map pane: map showing the user location
- Trends list: list of the top trends in the area.
- Tweet list: list of tweet based on the selected trend and location. This list will initially be empty showing a “no tweets” message.



Level 2 : Neighbourhoods

Two-pane layout showing:

- A map with the neighbouring areas with Twitter activity.
- Tweet list: list of tweet in the area related to the trend selected from the trends list. This list will initially be empty showing a “no tweets” message.



Key Considerations

Data Strategy

At start-up, the application gathers the device location using the Google Service Location. As well as location, search area radius and the units in which the radius dimension is defined are available at start-up. These two parameter will be initialized with default values but could be changed at a later state by the user.

Two distinctive requests are made to the Twitter Search API using the values mentioned, these are:

- Location
- Search area radius
- Search area radius measurement units.

Requests are made using the Retrofit library and these are:
/search
This is used to request a list of tweets based on location. The parameters provided are latitude, longitude, radius, and radius units

/closest
This is used to request a list of trends based on location. The parameters provided are latitude, longitude.

Other consideration: data granularity
The responses of these two requests differ on their level of data granularity. Where the '*closest*' request will provide data related to the closest area defined by WOEID (Yahoo's What On Earth id), the tweets on the search request response refers to 'neighbourhood' location data. Areas defined by the WOEID tend to be much large that the one defined by the neighbourhood data attached to tweets.

The event generate a user case where the user selects a trend present in the area defined by the WOEID but not present in the search request response which refers to the user's closest neighbourghoods. In this instance, no tweets related to that trend will be present in the search area defined by the user location. The ui will prompt the user with "no tweet available" message.

Data Persistence

The application is implemented using the Model–View–Viewmodel (MVVM) pattern which is achieved with the use of Android Architecture Components. Data persistence then will be implemented using Room Persistence Library.

As well as the normal benefits of using local data persistence, the implementation aims to minimize data request in order to comply with constraints imposed by the *Standard* Twitter Search API.

The data object to persist will be the "Tweet" as defined in the Twitter documentation.

The user will be able to set the radius for the Twitter Search as well as the radius length units, either kilometer or miles. This will be considered as application settings and stored as Shared Preferences.

Libraries & Dependencies

The following libraries are used in the project:

Dagger: the application will implement dependency injection

Retrofit: a type-safe HTTP client for Android

Twitter Standard API
The application will show tweets and twitter trends related to the user location. Two libraries will be imported into the project from the Twitter Kit Android library:
twitter-core : it provides Twitter authentication capabilities.
twitter-ui : it provides timelines and tweets ui artefacts.

Android Architecture Components
ViewModel, LifeData and Room components of androidx.lifecycle package will be used in this project. The libraries will provide the framework to implement the Android MVVM pattern.

Google Play Services
There are two service used in the project:
play-service-map: it provides map ui displays where to show the user location and the areas where the tweets are located.

play-service-location: it will provide user location. This information will be then used to make requests to the Twitter Standard API.

Implementation and Tasks

The project will implement the following steps in order to create the Shift•t app:

Task 1: Project Setup

- Creation of the project and set up of version control
- *Dagger*
  Initial implementation of *android.dagger* so a minimal framework is in place where to attach other component during the project implementation.
- *Configure libraries*
  Define dependencies at the build.gradle file of the libraries describe in the "Libraries & Dependencies" section above.
- *Import design settings*
  Dimensions, colours and styles defined on the project prototype files should be imported into the project. These values will be added the corresponding asset files e.g.: colors.xml for the colours values
- *Generation of graphics*

Task 2: Application Architecture

Implement a basic structure where all activity are present with their architecture component e.g.: ViewModel, LiveData. At this point the app will show all its screen and their transition with no data

Task 3: Data request and storage

- Implement Twitter Kit request using Retorfit
- Implement Shared Preferences to stored search area radius, radius units and latest location.
- Implement repository to handle the data request and storage as describe in Android Architecture Components guidelines.
- Implement Room so to store data returned by the Twitter Kit requests.

Task 4: Implement UI

The application implements an adaptive UI where different layouts are used for phone and tablet screens. Fragments will be used to implement the different view elements of the UI so that they can be reuse on both layouts.

ConstraintLayout should be used as a root layout for any view defined in the project.

MainMapFragment

- Adds a Map to the UI using the MapFragment provided by the Android Map Library.
- Implement a FAB providing two actions

NeighbourhoodMapFragment

- Adds a Map to the UI using the MapFragment provided by the Android Map Library.
- Draws a set of polygons defined by the Twitter's response location data.

TrendsListFragment

- It implements a RecyclerView to show a list of trends. In phone devices, intents will be used to start the TweetListActivity. The intent will hold information on the trend selected by the user.

TweetListFragment

- It implements a RecyclerView to show a list of tweets. Implementation will rely on the adapter implementation provided by the twitter-ui library where a Timeline object containing a list of tweets has to be given as parameter.

Showing a single tweet detail view will be delegate to the Twitter application.

Task 5: Testing

- Implement basic testing focusing on Room and the testing of the data persistence

Task 6: Further implementation

The project also includes other task as further development for extra marks. These will only be undertaken if time allows.

The list could be also considered as providing ideas for further development to be undertaken after this exercise.

- Implement Admob so a banner ad at the bottom to the screen is present.
- Allow users to choose a different location apart from their current location
- Remove refreshing constraint and allow users to refresh data at anytime.