
GitHub Username: `https://github.com/Cherudek`

Turin Guide

Description

The goal of this multi-screen Android app is to create a tour guide which presents a list of top attractions, restaurants, museums and bars to a user who's visiting the city of Turin in Piedmont Italy. The app aim is to share local knowledge with who is spending a break in the city. The app wants to be a handy guide for finding location based experiences and to plan a day in the city.

Intended User

The user is the person that is visiting the city of Turin, for holidays or for work and wants to have a useful guide on the most interesting places to visit such as art galleries, museums, historical places. The app targets also the food lover who wants to try traditional dishes from the rich piedmontese cuisine, experience the classic evening appointment of the aperitivo where the locals meet sipping a good glass of wine or a craft beer enjoying some food. The app targets also the music lover that is after some classical music at the Auditorium, a rock concert or maybe likes to dance to the beats of famous international dj .

Features

SIGHTS: list of interesting sites based on user location

MUSEUMS: list of interesting museums and galleries based on user location

USER AUTHENTICATION: the user will be able to login with: email or Gmail
and save favourite places to his/her account.

CLOUD DATABASE (Firebase): all the informations of the app including photos descriptions of favourites will be saved on a cloud firebase server to allow easy synchronization.

LOCAL DATABASE (Firebase): all the informations of the app including photos descriptions of favourites will be saved also on the phone thanks to firebase persistence feature.

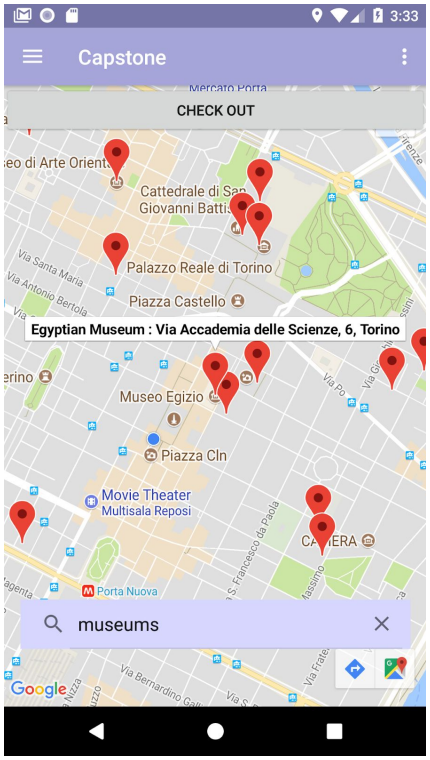
NOTIFICATIONS: the user will be able to receive notifications of his or her favourite places using geolocation as he strolls around the city.

MAP VIEW / PLACE PICKER: The user will be able to search and select interesting places from the map.

ADD TO FAVOURITES: save favourites locations to create a personalised tour on specific days.

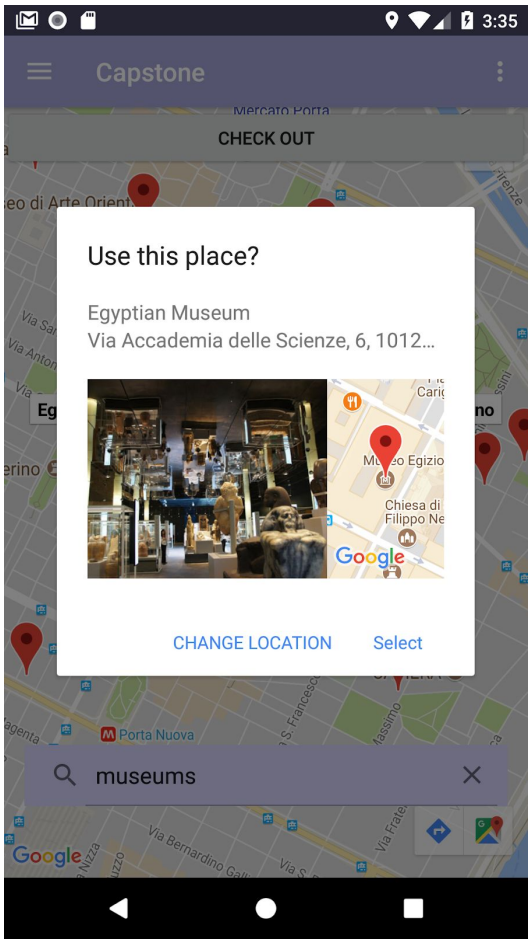
User Interface Mocks

Screen 1:



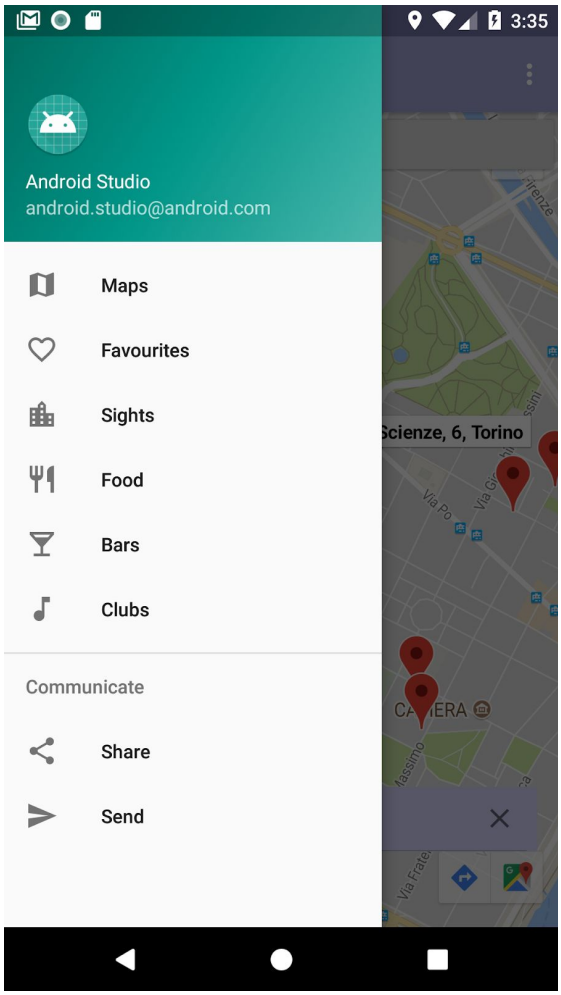
This is the landing page where the user after allowing location permissions, sees his position on the map and can search for places in his surrounding area (in a radius of 1500m). By clicking on one of the markers he or she can check infos about the place (name and address) and then with a furter click can access the detail page of the object selected with place reviews, photos, website address and if available opening times.

Screen 2: Place Picker View



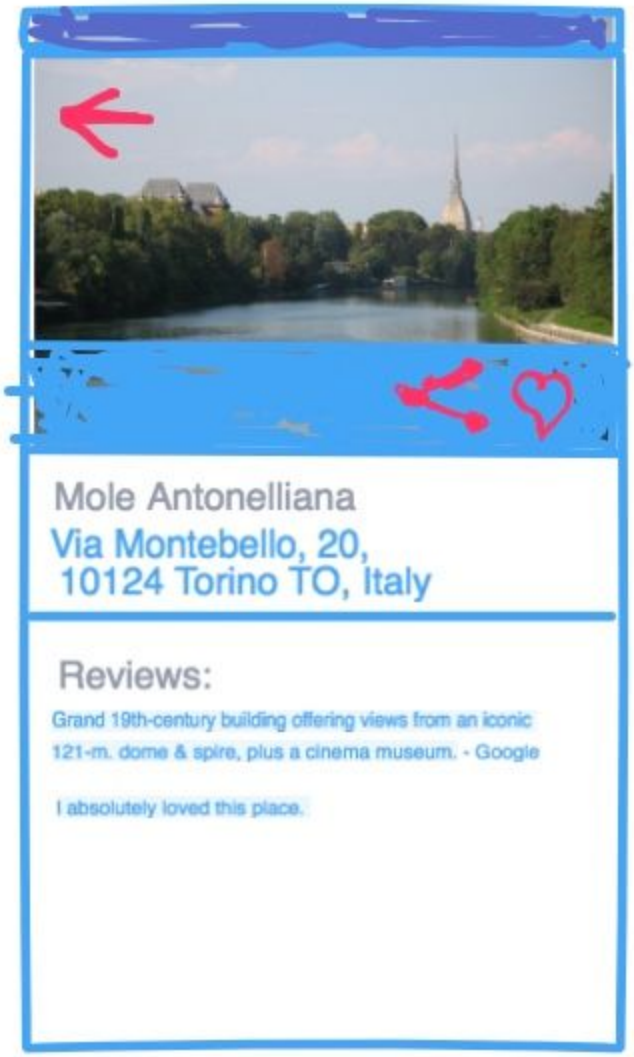
If the user clicks on the checkout place button, using Google Place Picker Api the user can move around the map and dropping the place picker on a specific location can get detailed informations by going to a detail page. On the detail page will be also possible to add a place to a favourite list that is synchronised with a Firebase realtime database. The detail page will also allow to share the location of the place with someone else.

Screen 3: Navigation Drawer



From the drawer view we can navigate through different UI elements and select particular type lists. Based on the user location, for example we can see on a list nearby restaurants, bars or sights. Each list will then allow the user to go to a detail page. The detail page will contain a map, photos and general info like address, websites and reviews…

Screen 4: Detail Fragment



The Detail View will show Name, address, website, reviews and the map location of the place. From the detail view the user will be able to save the location to his or her favourite list and also to share the location with someone else.

Screen 5: The Widget

The Widget will show the Favourite location saved by the user, clicking on one item it will open the Detail location of the place.

Key Considerations

How will your app handle data persistence?

Data persistence will be handled using Firebase Realtime database using local persistence enabled so the user can check data from the last known location, and his favourite list also when offline. OnSaveInstance will be used for saving small amount of data in case of phone rotation or the Activity getting destroyed. To persiste non trivial amount of data such objects including bitmaps data we will implement ViewModels and LiveData observable classes.

Describe any edge or corner cases in the UX.

If the user for example doesn't want to give permission to use the location. He or she can use an offline map of the city with a set of predetermined suggested places and browse offline data. It will also prompt a message highlighting the benefits of using location services for a better user experience and customised results.

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

- **Retrofit 2.0** to handle the Http connection and the JSON parsing from the Places api query, it reduces boilerplate for both handling the http connection and json parsing and also handles the call asynchronously on a separate thread
- **Picasso** to handle the image drawing on the detail layout, perfect for loading bitmap data on the UI and also great for caching of data and therefore to provide the use a better experience.
- **Firebase** api to handle cloud and local data persistence and Log in authorisation. This is the best choice as it doesn't require to write any backend code and allows to use Storage features, user login, database and also analytics and crash reports all in one place.
- **LiveData** is an observable class that has the main task of listening certain objects for data changes and to notify the via the ViewModel the UI of changes.
- **ViewModel** it will allow to persist state condition of object loaded on the UI freeing the Activity and the fragment from the responsibility of managing also data and allowing also a better testability of our classes and follow clear code best practice.

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

- **Google Location** api to detect user location and to provide customized data using the FINE LOCATION permission of the phone. This is the best option as it stores more than

100 million places all over the world and it gets updated by users with infos and guides from locals on a daily basis.
- **Places** api to provide location based search features and pick places around the user's location this api enjoys the benefits as the Location api mentioned above.

Next Steps: Required Tasks

Task 1: Project Setup

The Project will run on **Android Studio** 3.1.2 and the compiling will be done using **Gradle** 4.4 plug in. The target SDK is 27 and the minimum is SDK 19.

We will start by building a loading page with an animation while the app loads libraries such as AppCompat, Google location and Places api, Retrofit, Lifecycle libraries (ViewModel and LiveData), Firebase real time database, Firebase Auth to allow user login feature.

Task 2: Implement UI for Each Activity and Fragment

- Build UI for MainActivity
- Authentication Screen (the user authenticatication via Firebase)
- The authentication screen will explain to the user that he or she will be able to save locations in a favourite database and enjoy a better UX.
- Build UI for MapFragment loaded at start app in the MainActivity
- Build UI for the Drawer Menu to allow navigation between fragments.
- Build UI for Favourite location lists
- Build UI for Sights Near Me list
- Build UI for Food places Near Me list
- UI for Bars near me (list)
- UI for Clubs near me(list)
- UI for the detail view , with features for adding locations to favourites and share location
- If the user hasn't logged at the start will be prompted to log in before he or she is able to save to favourites.
- Build UI for offline maps of recommended places in case the user can't get any internet connection.

Task 3: Google Play Services Task

Downloading the Google Play Libs for Location Services and Places Api

- Build a Map Layout and a MapFragment
- Synchronise the Map Object
- Ask for the Permission to use the Location feature
- If the location feature is granted enable it on the map
- Set a search button for places nearby
- Set a pick up location button to use the Place Picker api
- The places retrieved will be shown on the map as markers
- If the user clicks on the marker a title and location detail message will pop out and it will also pop a link to get directions to the wanted place via Google Maps Intent.
- If the user clicks on the name it will launch an Intent to open the Detail Fragment.
- If the user clicks on the Place Picker Button, he or she can select a place on the map via picker or via a list of known locations nearby.
- If the User picks the location an Intent will open the detail fragment
- From the detail view the user can save to favourites and also share the location data.

Task 4: Firebase Task

Load the real time database and load the Authentication libraries :

- Fetch data to populate the favourite fragment
- Provide sync with firebase local data persistence in case the user is offline
- The database will also store the last known location

Task 5: Retrofit Task

Load the Retrofit 2.0 library to allow Http connectivity with the Google Places and location api and provide Json parsing:

- Via UI (Navigation Drawer or Map Fragment Search feature) we call the api for places nearby the user, Retrofit will handle asynchronously the call and return a data object and thanks to the Gjson api it will parse the data arrays.
- Once the data is fetched Markers will be placed on the map if we are in the Map Fragment
- If we are fetching a list of restaurant or bars we will visualize them in a Recyclerview in the relevant fragment (Food, Bars, Clubs, Sights)

Task 6: Notification Task

The user will get UI notifications when he or she enters near the area of one his or her favourites places.

- This feature will use the location data saved in the firebase database
- Using getCurrentPlace from our Places api will detect where we are.
- To avoid creating multiple calls to the server to check our position we will use a PlaceLikelihoodBuffer, if the likelihood returns a value higher than 60% we can say we are near the saved position and we call an pending intent to send a notification to the user.

Submission Instructions

- After you've completed all the sections, download this document as a PDF [File → Download as PDF]
    - Make sure the PDF is named "**Capstone_Stage1.pdf**"
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

- Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
- Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"