
Bologna transport

Description

This app helps to get around the city of Bologna Italy, using the bus.
Given a source and destination(both in bologna), the app will help you get to the destination by walking and changing buses. You can also store a request(source and destination), so you need not fill them every time you need.

Intended User

People in Bologna, Italy who want to get places in the city by using the bus.

Features

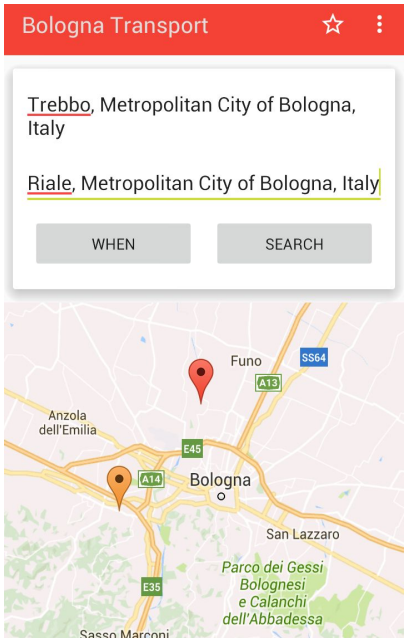
List the main features of your app. For example:

- We can fill in the source and destination either via autocomplete or putting place markers for source and destination.
- Options for itineraries(Ways to go about from source to destination) are shown, and we can choose to see the details of the selected option.
- If we choose to see a particular option, we can see the various legs of the option.
    - If a leg is walking, we can see the instructions regarding that.
    - If a leg is bus, we can see the intermediate stops.

User Interface Mocks

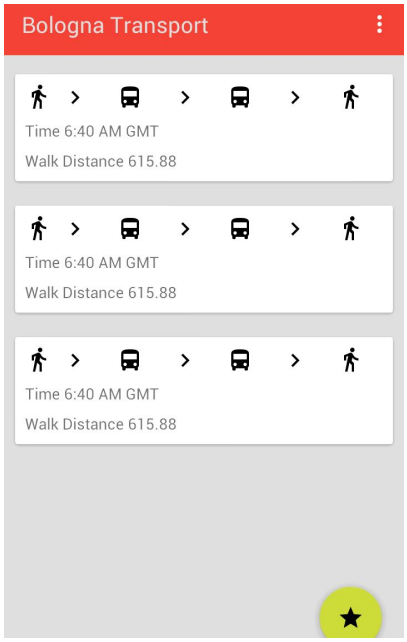
These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Photoshop or Balsamiq.

Main Screen



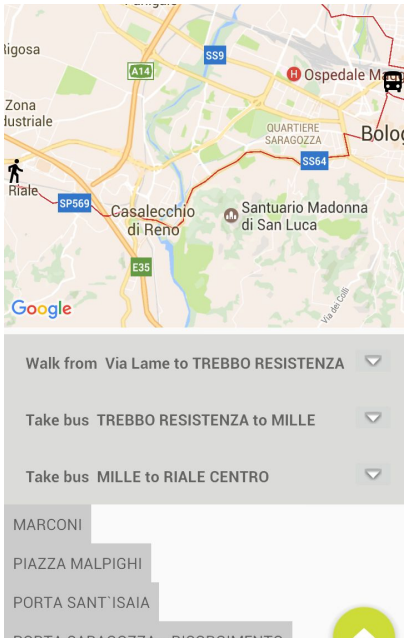
This screen is the main screen here we can fill in the source and destination to travel via autocompletetextview or moving the markers in the map below.

Options Screen



This is the options from which we can choose an option we want to see the detail about, There is also an option for saving the request, in case we like the options.

Results Screen

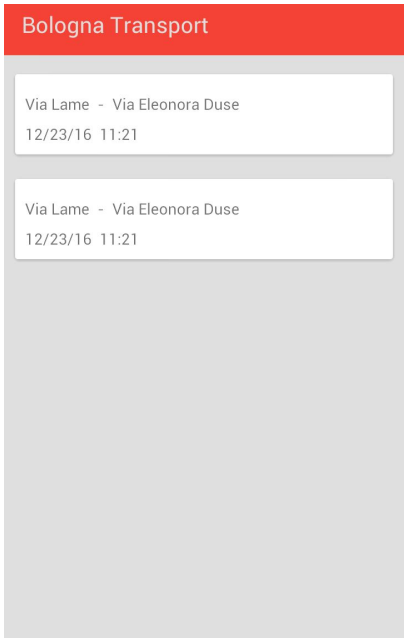


This is the options from which we can choose an option we want to see the detail about

-For bus gives the intermediate stops

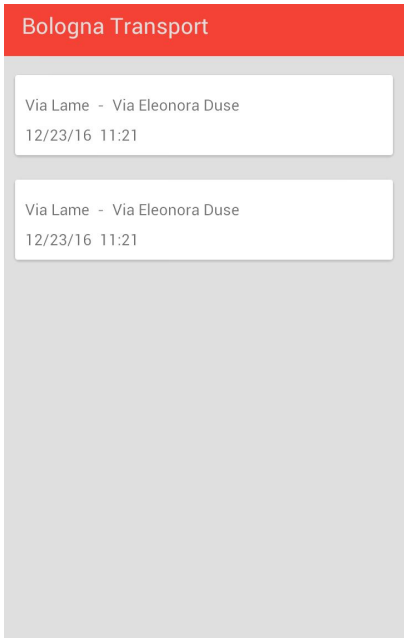
-For walk the instructions on taking right, left etc is given.

Saved Screen



This is the screen that show the saved requests in the db. After clicking on requests, it fills the from and to places as well as the time for the first screen, so that they need not be filled again.

Widget screen



This is the options from which we can choose an option we want to see the detail about,

Key Considerations

How will your app handle data persistence?

The app stored the request(source, destination, time) if we press the save button using content provider, we can use the saved requests for future use.

Describe any corner cases in the UX.

None.

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

Dagger - dependency injection
Butterknife - for binding views.
Retrofit and okhttp- for networking operations.
Gson - for json parsing the results.
Expandablerecyclerview - for the detailed result screen, for expandable recycler view.
Rxjava and rxandroid - for using as an eventbus and some communication between the layers,(since mvp is used for implementation)
Google maps,places, locations - For various uses and needs for showing the map, decoding the places.
Android support library recyclerview, cardview, design, appcompat- for various UI purposes.

Describe how you will implement Google Play Services.

Google maps- for showing the map in the first screen and result screen

Google places - For autocompletetextview for inputting source and destination.

Google location- For reverse geocoding the place selected by marker for source and destination.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and decompose them into tangible technical tasks that you can complete incrementally until you have a finished app.

Task 1: Project Setup

Start off with the https://github.com/ribot/android-boilerplate boilerplate code, rather than starting off empty.
You may want to list the subtasks. For example:

- Add the required libraries whichever is missing.
- Retain the necessary elements and eliminate others, and make a successful build.

Task 2: Implement UI for Each Activity and Fragment

- Build UI for MainActivity
- Build UI for Options Activity.
- Build UI for Results activity.
- Build UI for Options Activity.

Task 3: Build the adapters that are required for the lists.

- Adapter for the autocompletetextview
- Adapter for recyclerview in options screen.
- ExpandableRecyclerview adapter for result screen.
- Adapter for saved screen.

Task 4: Parsing and composing of data for network and db.

- Create the classes that are required for composing and parsing the json data over the network with retrofit.
- Create the db schema and implement the contentprovider which stores a request as well as tetrieves the saved request.
- The app will also use async task for reverse geo coding(and fetching of results over network) of points which were input using markers on the map in the main screen
- The saved screen will use loaders to load data from contentprovider to be displayed.

Task 5: Wire up communication between activities.

- The main activity receives a response from the network and passes it onto the options activity. It uses json to pass data between activities.
- The options activity passes it onto the result activity, so that it can show the result.
- On click of save we can save the current request on to the content provider.
- On click of saved button on main screen, saved activity is launched, which retrieves all the saved requests and displays it.
- On click of one of the saved requests, the main activity is launched with the particular request filled in.

Task 6: Widget and handling errors.

- Add the widget which displays the saved requests screen. On click of the widget the saved requests screen is launched.
- Handle the error cases when we receive a malformed or other input.
- Handle the error when we receive a malformed result from the server.
