How to Use this Template

1. Make a copy [File → Make a copy...]
2. Rename this file: "**Capstone_Stage1**"
3. Replace the text in green

Submission Instructions

1. After you've completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"

GitHub Username: flaviofrancisco

MyCondoBus

Description

Brazil has large cities and a not well developed public transport system. For that reason some condominiums (condos) hires private bus companies to serve a better and safer service in terms of common transportation in the cities.

Basically a person that lives in one of those condos pays an amount per month to use it. Usually is included in the condo fees. Each dweller has a document id that must be shown to the bus driver before be on board.

The goal of this app is offer to those condos a way to help the users to find the closest bus stop based on theirs location and destination.

Intended User

Users that are living in condos that offers a private bus service to theirs dwellers.

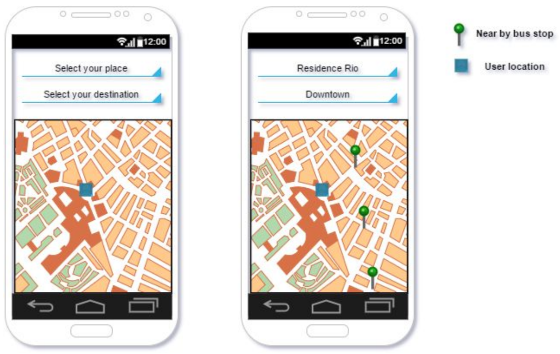
Features

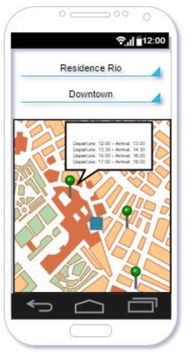
The main features for this app are:

- Show the closest bus stop of the condo bus.
- Show the departure times of the destination line and estimated times of arrival on the last stop.

User Interface Mocks

Screen 1

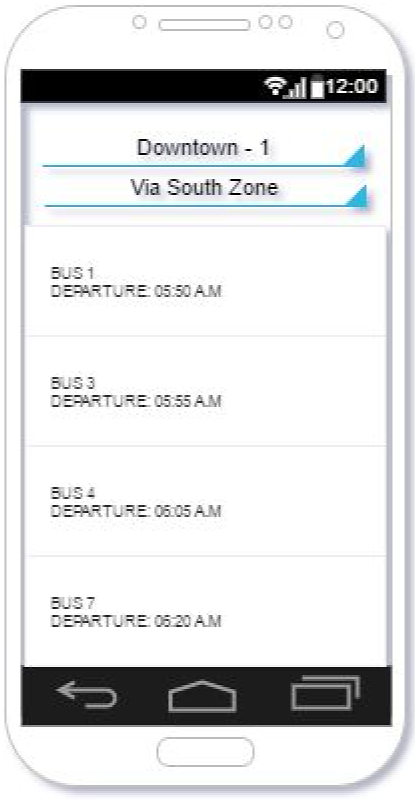




Departure: 12:00 - Arrival: 13:00
Departure: 13:30 - Arrival: 14:30
Departure: 15:00 - Arrival: 16:00
Departure: 17:00 - Arrival: 18:00

On this screen the user will be able to select the departure bus stop of reference and the arrival bus stop of reference. Once both information are informed than on the map will be shown the closest bus stops based on the user location and the departure and arrival estimated times. The departure and arrival times will fetched by an web service that will explained on the further sections.

Screen 2



BUS 1
DEPARTURE: 05:50 A.M

BUS 3
DEPARTURE: 05:55 A.M

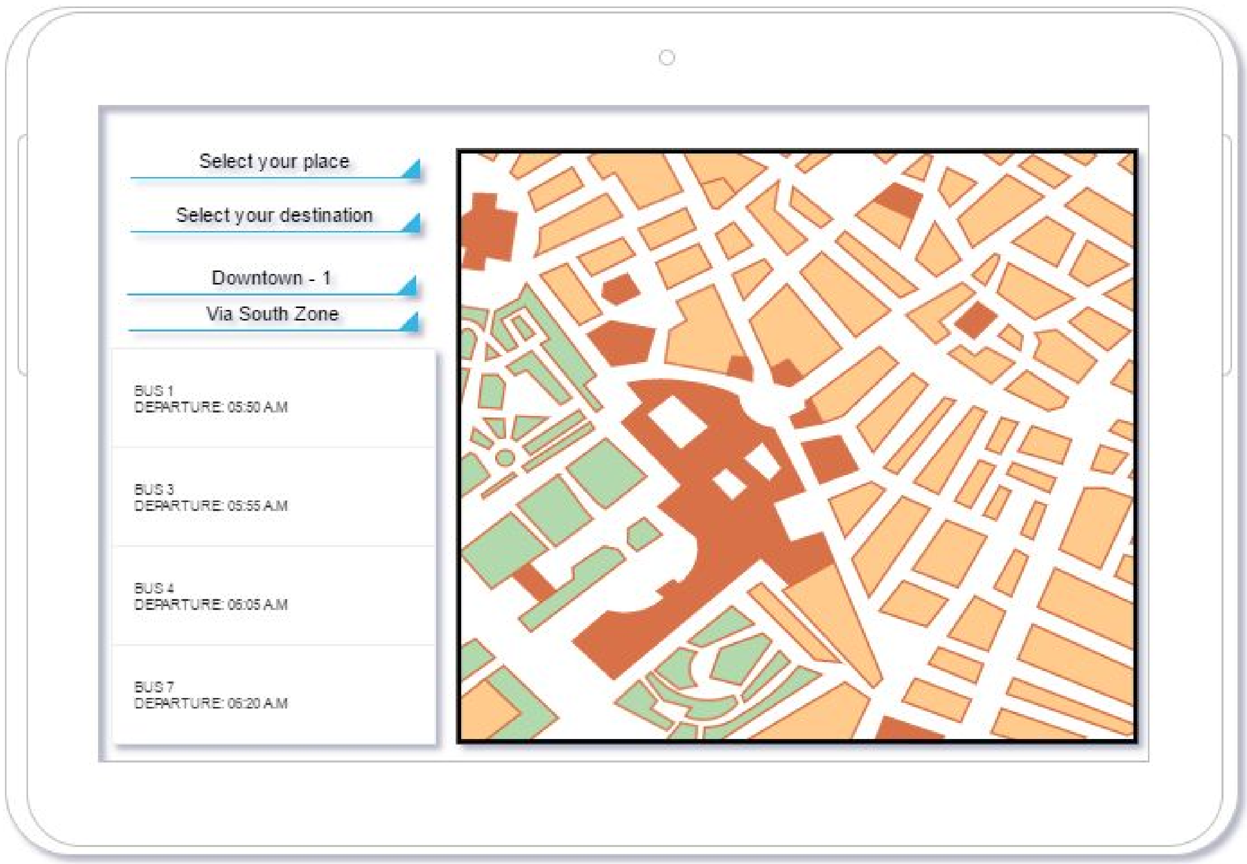
BUS 4
DEPARTURE: 06:05 A.M

BUS 7
DEPARTURE: 06:20 A.M

This screen provides the time schedule based on the destination and path of the bus showing the departure time of each one.

Screen 3

The tablet view of the Screen 1 and 2



Key Considerations

How will your app handle data persistence?

The app will have a content provider to save the info need locally.
The first load of data will come from an web service. The user could update the info manually using the menu synchronize with the online information.

Describe any corner cases in the UX.

A right corner menu will be used to navigate between the two screens.

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

The following libraries will be used:

- Location Services
- Google Maps Android
- External Web Service developed in .Net that will feed the ContentProvider.
- Fused Location Provider (to analyse the wi-fi; network and satellite location)

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

Create a Google ApiClient

Set up the credentials to the app access the google apis.

Use of the follwoing Apis:
Location Services

Edit build.gradle:
 Include: compile 'com.google.com.gms: play-services:7+'

Edit Android Manfiest:
Include the: android.permission.ACCESS_FINE_LOCATION

Task 2: Implement UI for Each Activity and Fragment

List the subtasks. For example:

- Create the layout of the Screen 1
- Create the layout of the Screen 2
- Build UI for MainActivity - (Screen 1)
- Build UI for BusScheduleActivity (Screen 2)
- Extend the activities with the ConnectionCallbacks

- Extend the activities with the OnConnectionFailedListener

Task 3: Your Next Task

Handling Error cases:

- OnConnectionFailed;
- OnConnectionSuspended

Handling connection statuses:

- OnConnected
- OnLocationChanged

Create the Content Provider with the schema tables.

Submission Instructions

1. After you've completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it “**Capstone Project**”
3. Add this document to your repo. Make sure it's named “**Capstone_Stage1.pdf**”