
GitHub Username: OlegSheliakin

PlacesNearMe

Description

An amazing app for finding the best restaurants, bars, cafes and attractions in any city in the world! This app makes easy to find great places near you. Just choose the category you are interested, enter a keyword to search for and enjoy visiting places. Also you can add the best ones to your favorite and see your history of visited places. In the description of the place you

can see reviews, photos, ratings and much more useful information that will help you make the best choice.

Intended User

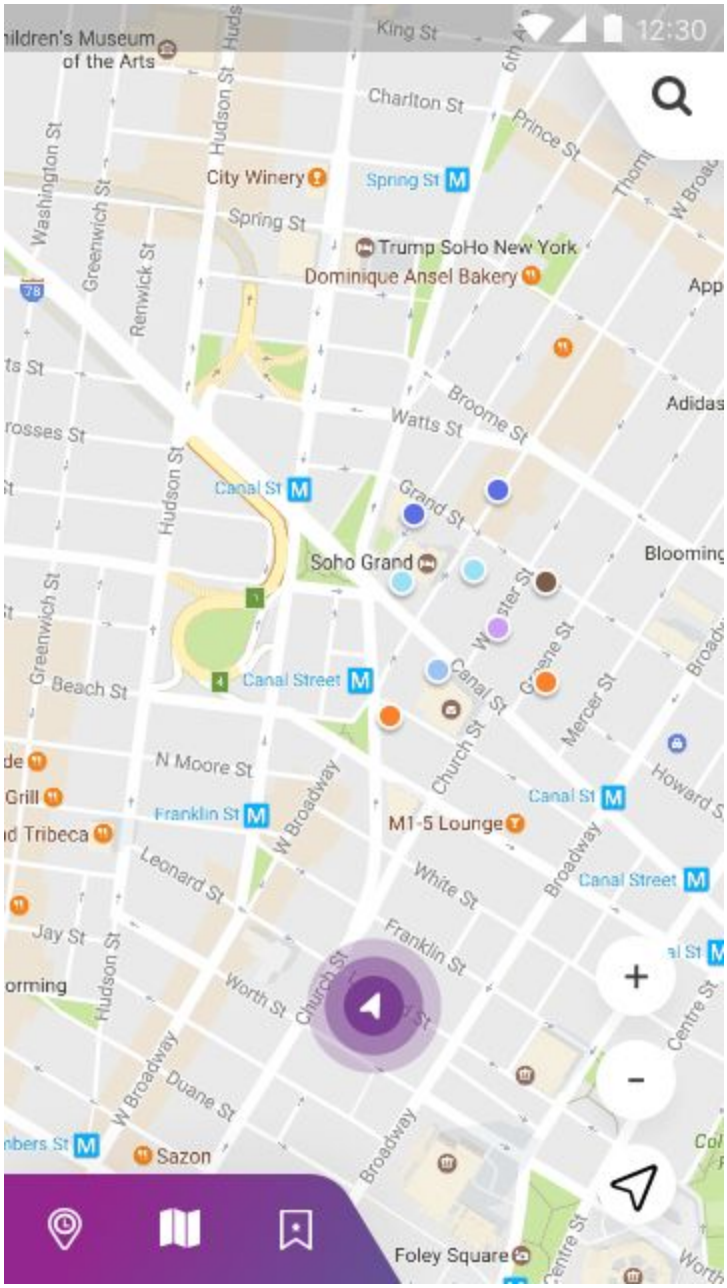
This application is for those who like to drink a cup of coffee in a quiet cozy place or to taste delicious food in a good restaurant. And also for those who like outdoor walks and seeing sights.

Features

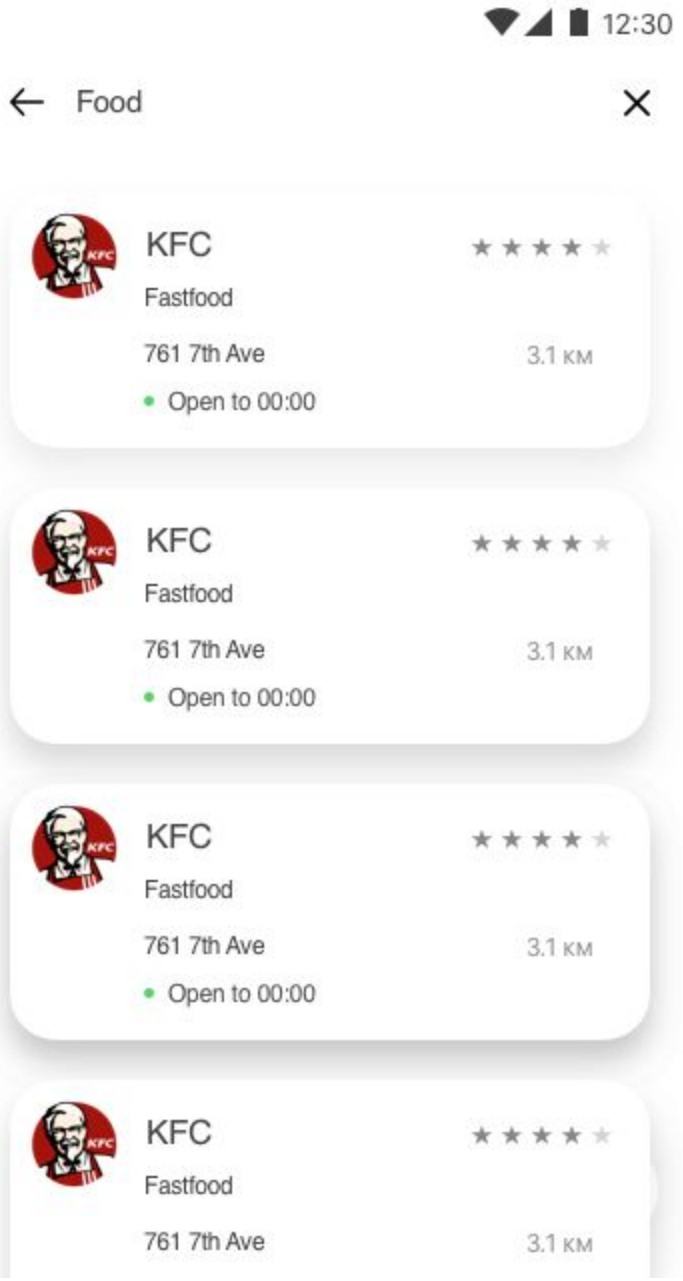
- Searching places on the map
- Getting list of recommended places
- Getting place’s details
- Getting a history of visited places
- Getting place’s tips
- Finding routes

User Interface Mocks

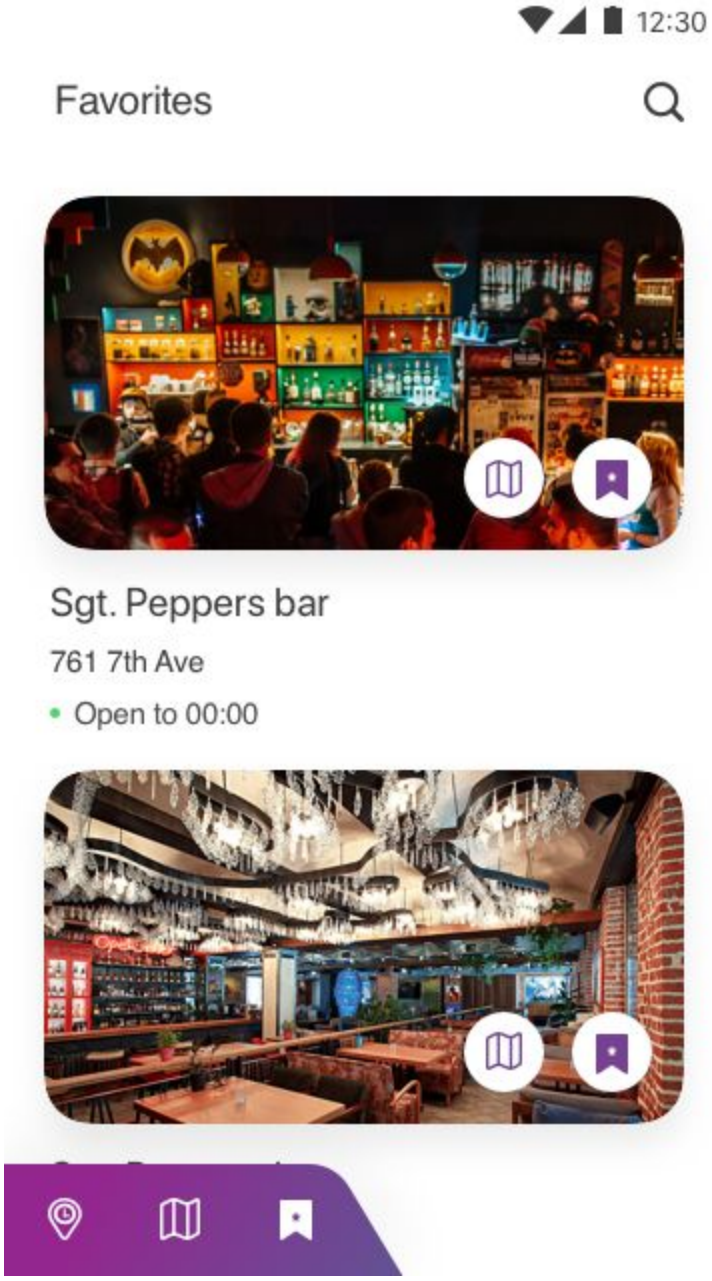
Map screen



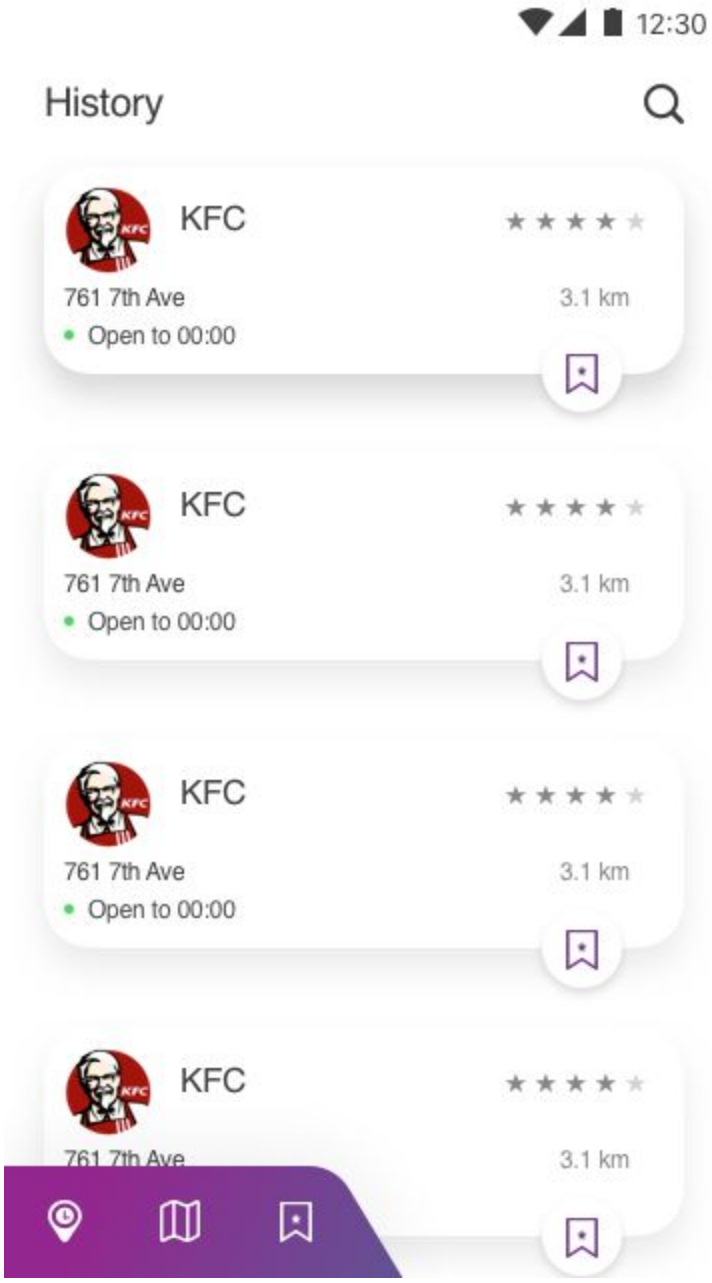
Search screen



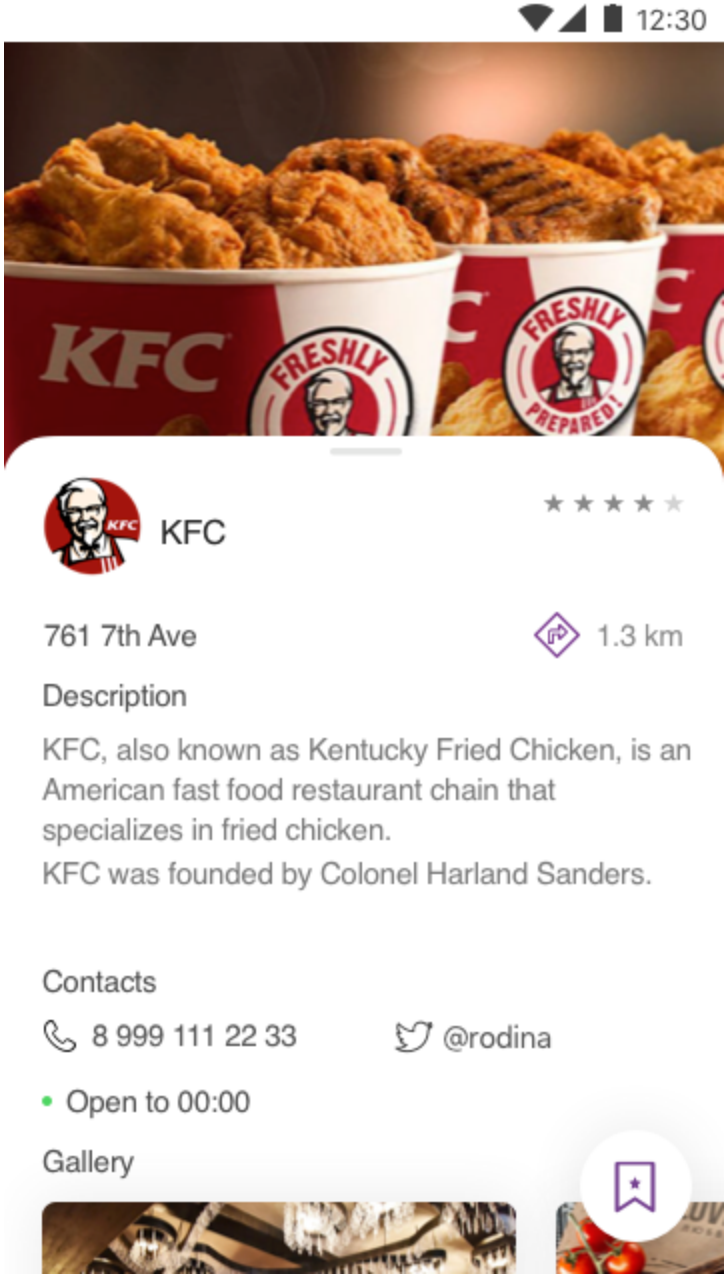
User’s favorite places screen



User’s places history screen



Place’s details screen



Key Considerations

How will your app handle data persistence?

Data persistence will be handled by Room database and SharedPreferences.

Describe any edge or corner cases in the UX.

- App does not redefine the expected function of a system icon (such as the Back button).
- App does not redefine or misuse Android UI patterns, such that icons or behaviors could be misleading or confusing to users.
- App supports standard system Back button navigation and does not make use of any custom, on-screen "Back button" prompts.
- Pressing the Home button at any point navigates to the Home screen of the device.
- App correctly preserves and restores user or app state, that is , student uses a bundle to save app state and restores it via onSaveInstanceState/onRestoreInstanceState. For example,
    - When a list item is selected, it remains selected on rotation.
    - When an activity is displayed, the same activity appears on rotation.
    - User text input is preserved on rotation.
    - Maintains list items positions on device rotation.
- When the app is resumed after the device wakes from sleep (locked) state, the app returns the user to the exact state in which it was last used.
- When the app is relaunched from Home or All Apps, the app restores the app state as closely as possible to the previous state.

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

The libraries will be used:

- *RxJava 2* - for asynchronous programming
- *Dagger 2* - for dependency injection
- *Retrofit* - for network requests
- *Gson* - for JSON parsing
- *Glide* - for image loading
- *Timber* - for nice logging
- *RxBindings* - to leverage using rx for views events
- *Espresso* - for UI testing
- *JUnit* - for unit testing
- *Mockito* - for mocking dependencies of units

- *PilgrimSDK* - for location detection
- *JetPack:Room Database* - for handling data persistence
- *JetPack:Paging* - for pagination
- *JetPack:ViewModel* - for surviving UI state under configuration changes
- *JetPack:Navigation* - for in-app navigation

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

The app will use:

- *Google Map* for adding map and tracking user's location
- *Firebase Crashlytics* for tracking any crashes in the app

Next Steps: Required Tasks

Task 1: Project Setup

- Creating project in Android Studio
- Configure libraries

Task 2: Implement UI for Each Activity and Fragment

- Build UI for MainActivity
- Build UI for FavoritesFragment
- Build UI for HistoryFragment
- Build UI for MapFragment
- Build UI for PlaceDetailsActivity
- Build UI for SearchFragment

Task 3: Implement Google Play Services

- Add Google Map SDK
- Add Firebase Crashlytics SDK

Task 4: Implement network API interface

- Create API interface for retrofit
- Create http interceptor for adding "apiKey" query param to every network requests

Task 5: Implement data persistence

- Create Entities
- Create Dao objects

- Create Database

Task 6: Implement DI

- Create application component
- Create components for each Activity and Fragments
- Create network module
- Create repository module
- Create RxModule
- Create Application module

Task 7: Implement business logic for each use cases

- Create use case for searching places
- Create use case for user sign in
- Create use case for saving places
- Create use case for getting full information about place
- Create use case for getting user's history of visited places

Task 8: Write Unit tests

Cover all required classes with unit tests.

Task 9: Write UI tests

Write UI tests for each necessary Activities, Fragments etc.