**GitHub Username**: AlessioPz

# Real Quest

## Description

This app is a useful tool to organize a treasure hunt game in real life. You can play the role of Master that organize the quest, or the role of Finder.
The Master set his current position as end point, then save the quest with some little information (description, start time). The Finder receive the list of all quests, with them very approximate location.
When the game starts, every Finder receives only the distance between the end point and his current position, no directions. The player can know which way just moving.

## Intended User

The intended users are guys who like the game.

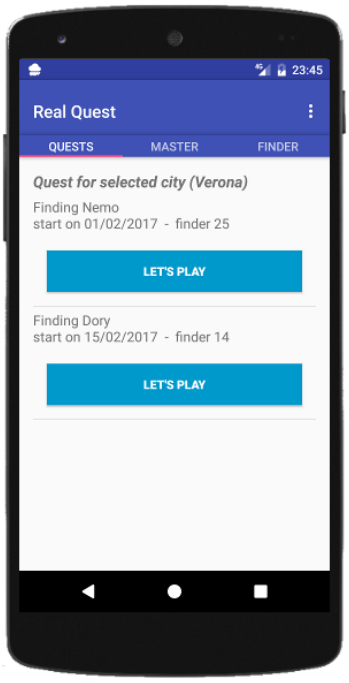
# Features

Main features list:

- User sign-up/sign-in
- Save information of the quests
- Users synchronization
- Quest management (geo-localization, start/stop, scores)

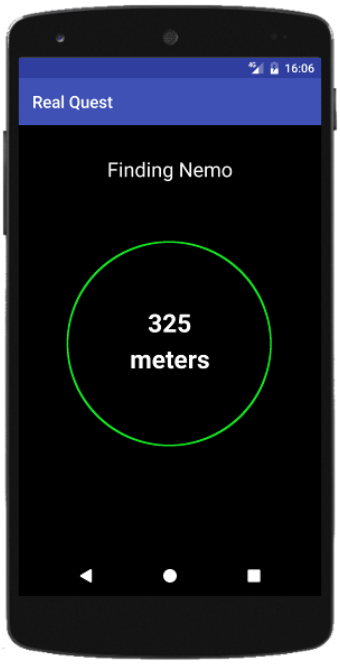
# User Interface Mocks

## Screen 1



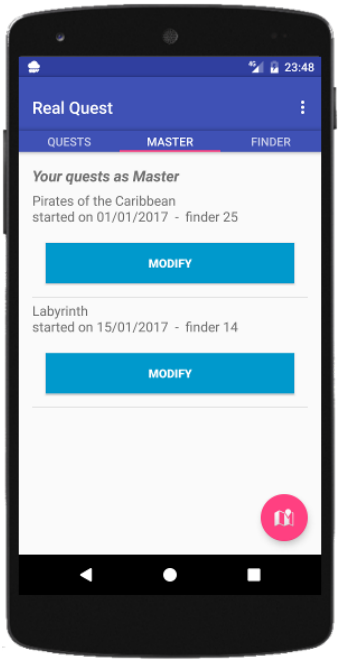

Quest list – list of quests for the selected city. Current location is filtered by default, user change selected city.

## Screen 2



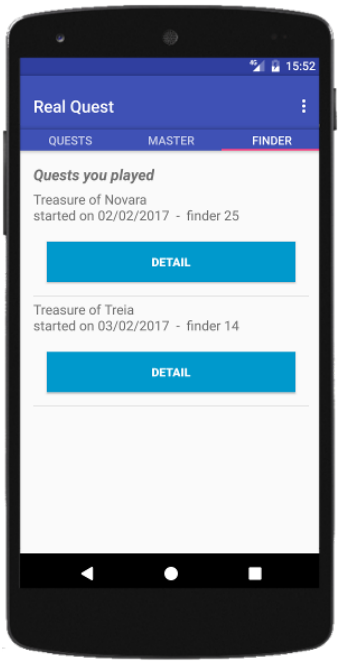

Play Quest – when the payer select “Let’s play” from previous screen, the screen is showed if the quest is already started, otherwise a pop-up asks if the user want to register t select quest.

## Screen 3



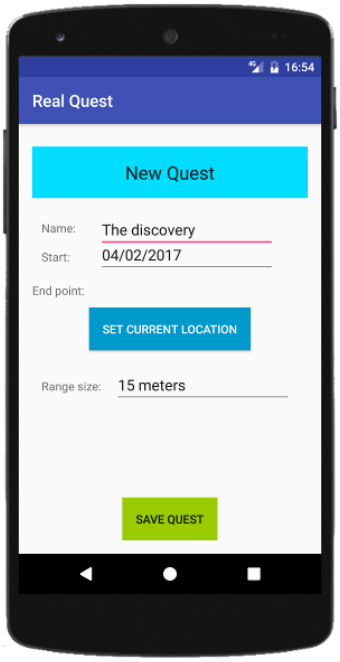

Master List – list of the quests created by this user.

## Screen 3



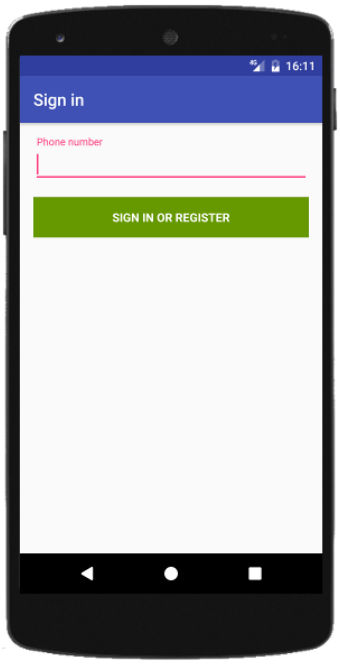

Finder List – list of the quests played by this user.

## Screen 4




New Quest – form to create a new quest as Master. This page is showed when the user tap the red button in the Master list (screen 3).

## Screen 5




Sign In – form to create sign in or register new user.

## Screen 5


Widget – list of quests for the current location.

# Key Considerations

### How will your app handle data persistence?

This app stores some information into a local SQLite database, managed by its Content Provider. These information are synchronized with a Cloud database.

### Describe any corner cases in the UX.

The app invite the user to create a quest if there is not for the city selected, or to change the filter.

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

ButterKnife for the views bind.
ACRA to catch exceptions, retrieve context data and send them to my own backend.
Google Objectify to store data into Cloud datastore.
Google AppEngine to deploy application endpoints.

### Describe how you will implement Google Play Services.

GMS: for memorize current location and calculate distance between current location and the end point.
firebase-ads: optional ad banner, hide if the user is subscribed.

## 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

Subtasks:

- Create screens structure: main activity with three pages, login activity, play activity
- Create local SQLite database and Content Provider
- Set permission for localization, internet, network state, billing
- Create backend endpoint module
- Implement Cloud synchronization: AsyncAdapter, Endpoint client
- Implement widget: text with city of local position, quest list filtered by city

## Task 2: Implement UI for Each Activity and Fragment

Subtasks:

- UI for MainActivity: Toolbar, TabLayout, ViewPager, FloatingActionButton
- UI for pages: TextView for the page subject, ListView
- UI for list item: TextViews for content descriptions and Button for the page's action
- UI for playing: TextView for Quest name and TextView for meters, draw coloured circle via Canvas (green – approaching, red – moving away)
- UI for new quest: input view for name, start date, range size; Button to set location, Button to save
- UI for login: input view for phone number and Button for action

## Task 3: Technical tasks

Subtasks:

- MainActivity pages managed with a FragmentPagerAdapter with three sections
- Every page implements a Fragment class and an ArrayAdapter
- App manage local data with a ContentProvider and a LoaderManager
- RealQuestWidgetProvider extends a AppWidgetProvider and receive data from a RemoteViewService
- PlayActivity implements a LocationListener to show the distance between local position and end point.
- The endpoint module is the backend manager. Store data using Objectify library.

## Task 4: Improve graphic layout

Chose theme colours. Search or create widgets, icons, background images.