Proximity - Design Document

1 - Introduction

1.1 - About the assignment

The application has been developed for the course “Design and Implementation of Mobile Applications” at Politecnico di Milano. The course aims to present the main techniques and technologies to design and implement applications for mobile devices. The assignment was quite open, and the team was given the freedom to choose what they would make. From the start there was a vision to create an application that would revolve around geography, as one of the team members already was studying geoinformatics at Politecnico. One of the top priorities from the beginning was to focus on a user-friendly design, and make this an app everyone could enjoy.

1.2 - General description and functionality

The application is called Proximity, and is an app mainly targeting people who want to travel and explore Italy. First the user is introduced to a welcome page including a quick informative tutorial of how to use the application. The user can then navigate to the explore page where a map will be shown. On this page the user’s current position will be shown, and the user can open a menu and choose a radius. The app will then locate Italian cities within that radius, and display them to the user both in the form of pins on a map, and a list view of the cities containing more information. For now the application only includes Italian cities, but it is clearly possible to extend this to other countries as well.
The idea for this application came from personal experience. When you use large map applications from companies like Google and Apple, you can easily get lost in too much information when trying to decide your next travel destination. The team wanted to make a simple application where the user easily can explore nearby Italian cities by using an interactive map.

2 - Requirements

2.1 - Functional requirements

The functional requirements are written as user stories and are shown below.

Priorities are listed as:

High - Necessary to use the application

Medium - Will improve the product significantly

Low - A nice addition to the application

ID	Requirement	Importance
FR1	As a user, I would like to be able to view and interact with a world map	High
FR2	As a user I would like to be able to view my own current position on the map	Medium
FR3	As a user I would like to be able to set a radius from my current position	High
FR4	As a user I would like to view cities in Italy within that radius on the map	High
FR5	As a user I would like to be able to get more information about the individual cities within the radius	Medium
FR6	As a user I would like to be able to view the cities in a list view as well	Medium
FR7	As a user I would like to be able to sort the cites based on population	Low
FR8	As a user I would like to be able to sort the cites based on distance	Low
FR9	As a user I would like to be able to manually search for cities	Low
FR10	As a user I would like to be able to go through a quick tutorial to learn how to use the application	Medium

2.2 - Non-functional requirements

Modifiability:

As already mentioned, the team wants it to be easy to expand the app to not only cover Italy, but other countries as well. A high degree of modifiability in order to add, remove and modify features is therefore required. In addition, if the data in the future is going to be stored in a database, the code must be modifiable.

Usability:

Another important attribute in the eyes of the team is usability. When creating an app for a user, it is essential to make sure it is easy for the user to interact and navigate through the app with ease. It affects user satisfaction and overall user experience

3 - Choice of technologies and data used

3.1 - Choice of framework

React Native:

For this project, the team chose to use the React Native framework, an open-source framework used to design mobile applications. Initially the plan was to use Flutter, as this was the framework that was introduced first in the course, and the overall structure seemed nice. The team managed to set up the environment, but when the development of the map-components began, there was a realization made. Flutter lacked some of the libraries the team required to do the geospatial calculations in the app. Since React Native has been around longer than Flutter, it has a larger community of developers and a wider range of third-party libraries. This included the libraries the team was seeking.

In addition the team already had some experience using React and JavaScript, and even though React and React Native are not identical, they use the same programming language and provide many of the same libraries. So after some thought, using React Native became the obvious choice.

Expo Go / Xcode:

To preview and simulate the application the team have used a combination of Expo Go and Xcode. Xcode lets the user set up a virtual Apple device on their computer which can be really helpful, but at the same time it consumes a lot of memory and processor power from the computer. Therefore the team mainly utilized Expo Go, a mobile application designed to allow users to run their code on a physical device.

3.2 - Target platform

iOS/iPadOS:

Even though React Native is a cross-platform framework, the team mainly focused their development on providing an application suited for iOS and iPadOS. The application should be able to run on an android device, but the default map is different so there might occur some issues related to the dependencies. The application has been tested physically on both iOS devices and iPadOS devices.

3.3 - Data storage

Frontend storage:

Because the application currently only considers Italian cities, the data amount is really not considerable at all. The .json file containing the cities and the information about the cities has a size of only 176kB, so the team decided that setting up a database would be an overkill. The solution was to store the city data in the frontend along with the rest of the code so that the script responsible for fetching the data could get access to it directly. This would both improve the performance of the application, and save the development team some time. It is to be noted though, that if the team were to expand the application to include larger datasets, they would create and use an external database service.

3.4 - Use of external services

Apple Maps:

Through the library ‘react-native-maps’, the application utilizes an Apple Maps API. The way this works is that when the user interacts with the map and moves around, the API fetches map tiles in real-time. In other words, all the tiles from the whole world map are not loaded simultaneously when the user opens the application, but rather just the parts currently visible in the map view. This is done to increase performance and decrease loading time, which again improves the user experience.

OpenDataSoft:

The data used for the cities is collected from the French company OpenDataSoft. They provide a tool for sharing the data of companies and public administrations. In our case, Italian cities. Their website allowed us to download the cities and information related to the cities in a .json format, which made the data compact and easy to use.

Link to website:
https://public.opendatasoft.com/explore/dataset/geonames-all-cities-with-a-population-1000/table/?disjunctive.cou_name_en&sort=name

Wikipedia:

Another external service used is Wikipedia. When the user is on the “More info”-page of a city, the user is provided with a link to the city’s wikipedia page. When tapping the link the user is taken out of the Proximity app and directed to the user's default browser where the wikipedia url is opened.

Wikipedia is also used for displaying an image for every city on their own “More info”-page. This is done by a wikipedia API where the top picture of the city’s wikipedia page is provided. By providing the top wikipedia picture the cities get very different types of photos. For example Milan displays a photo of the skyline, while Rome has an image of the Trevi fountain.

3.5 - Libraries used

react-native-maps:

The central element of the application is the map, and the library react-native-maps provides exactly this. It also provides markers, which was an essential part of our idea to display the locations. You have the opportunity to choose between Google Maps and Apple Maps, but the team preferred the aesthetics of the Apple markers. The library also has built in zooming and panning features, which made the development easier.

geolib:

Another critical aspect of the app is the calculation of distance between the user’s location and a city. There is a possibility of computing the distance between two geodetic coordinates using various mathematical formulas and matrices, but the library does these calculations with the user just having to insert the radius, latitude, and longitude of the two points.

expo-location:

This library extracts the user's real-time location, and is able to integrate with the map provided by react-native-maps.

react-navigation/native:

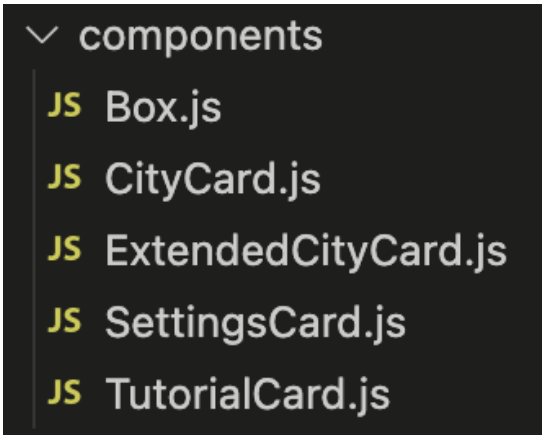
The menu/navigation system in the application is based on a very popular library “react-navigation”. It offers a clean, minimalistic navigation system which fits very well with our design.

4 - Architecture

The images below show how the files of the application are organized.

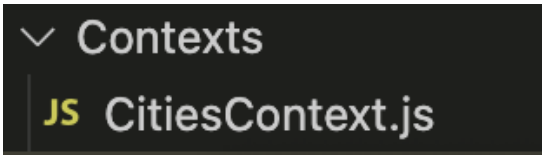
4.1 - Components

The components file contains components that are used in different parts of the application. These files make it easy to reuse code.



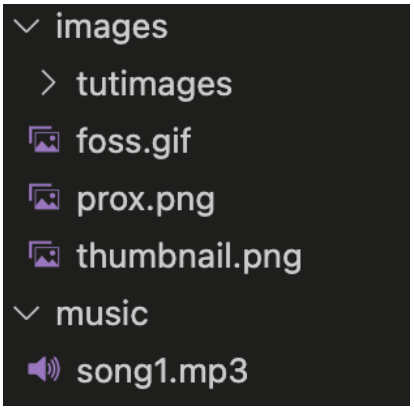
4.2 - Context

Context provides a way to pass data through the component tree without having to pass props down manually at every level. The team only used one context-file for the data exchange between the Explore page and the City List page.



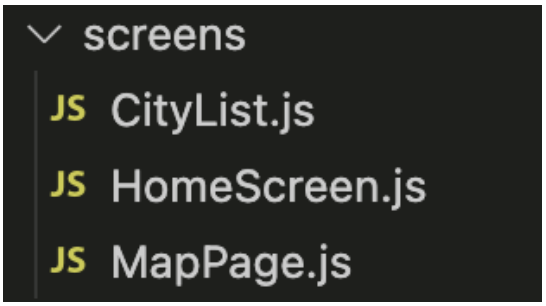
4.3 - Images and music

These files contain images and the music-track used for the application. The images that are used for the application are the tutorial images, the waterfall background and the logo.



4.4 - Screens

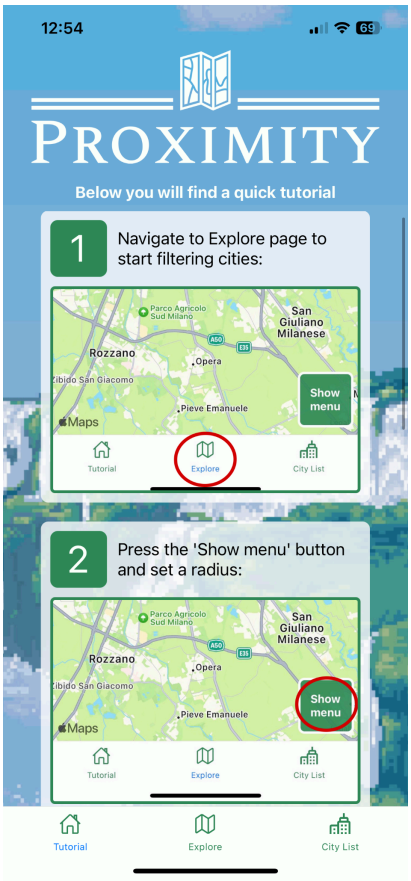
In the screens folder the team added all the screens.

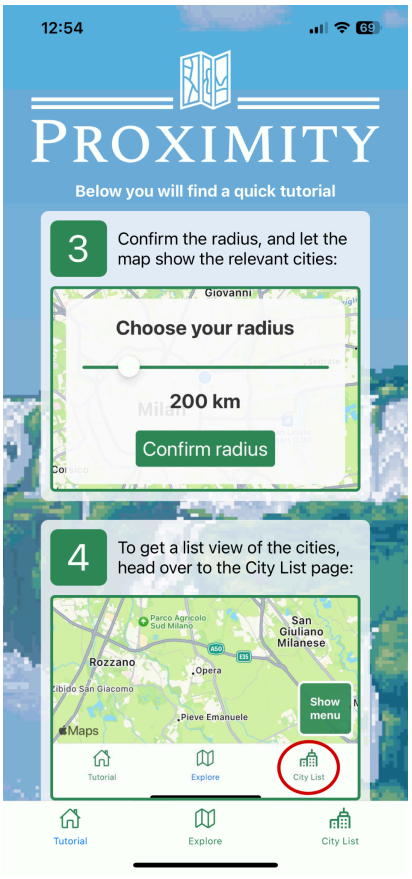


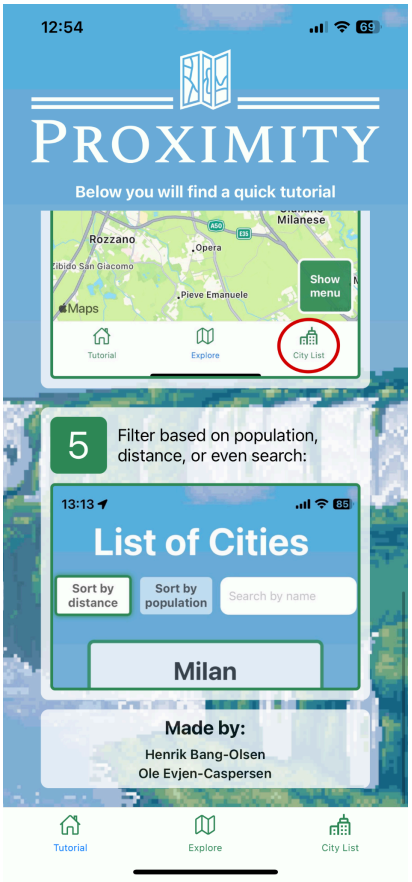
5 - User interfaces

5.1 - Tutorial page

The tutorial page is the first screen the user sees when starting the application. This screen welcomes the user and shows a quick 5-step tutorial on how to use the application. The tutorial consists of intuitive pictures with a small explanation to every step. From this screen the user can use the navigation bar at the bottom to navigate to any of the other screens. If the user wants to read the tutorial again at any other time, the user can use the navigation bar to navigate back to the tutorial screen.

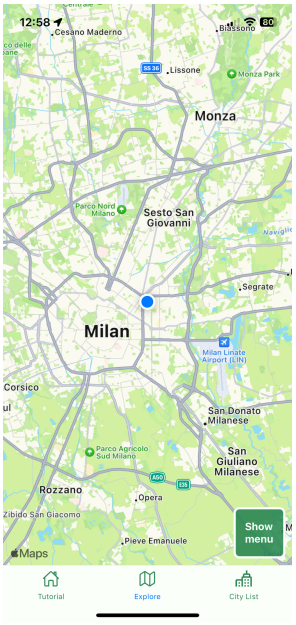


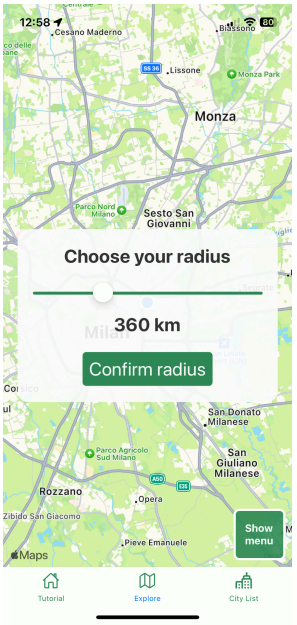


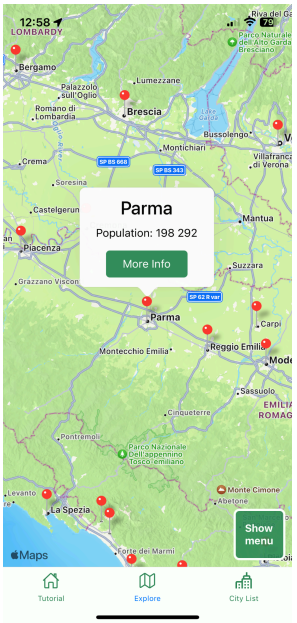


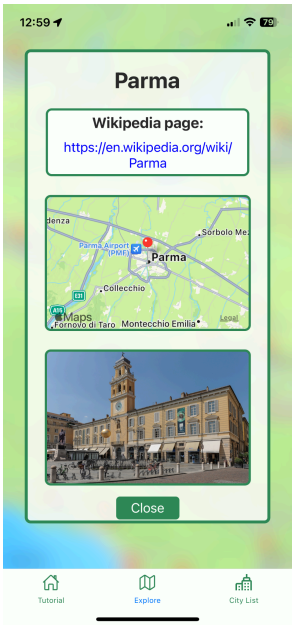
5.2 - Explore page

The explore page shows a map with the user's current location. When loading in the user's current location for the first time a loading screen is shown. When on the explore page the user can interact with the map by moving and zooming. In the right bottom corner the user can open a menu by tapping the "Show menu" button. When the menu opens, the user can set a radius by using the slide-bar and confirming the distance. The application will then show all the Italian cities within this radius in the form of pins shown on the map. When tapping on the pins a small pop-up panel will show. This panel shows the city population and also has a "More info" button . When tapping on "More info" the user will see a new pop-up that provides the city's wikipedia-page, location and a picture. The user can always change the distance if they want to see more or less cities.



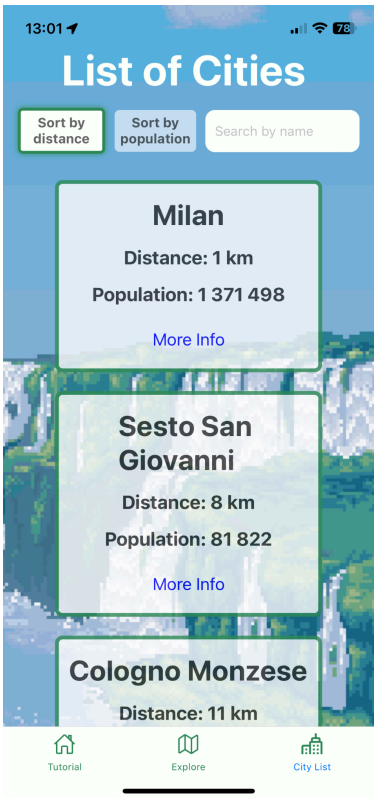


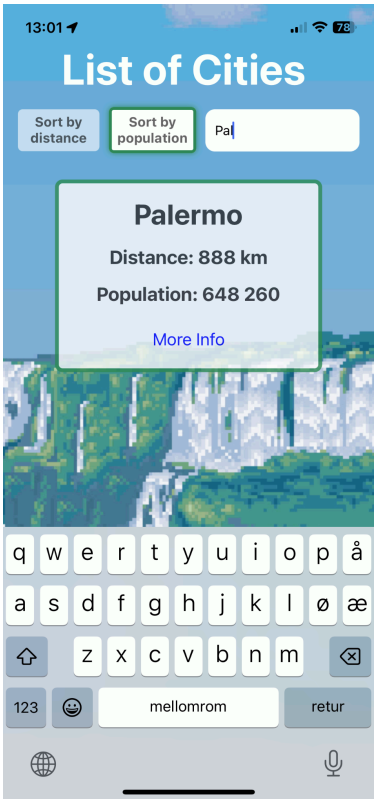




5.3 - City List page

On the City List screen the user can see all the cities shown on the Explore page in the form of a list. When updating the radius on the Explore page the City List also updates, showing the same cities. On this page the user gets a more clear, easy-to-read view of the cities. The user can choose to sort the list on distance, showing the nearest cities first, or on population, showing the most populous cities first. The user also has the option to search for cities, which can be nice when the user has set a large radius and wants to find a specific city. Like on the Explore page the user can tap on “More info” and additional information will be displayed.





6 - Test Campaign

The team made extensive use of user-testing to ensure code quality. The way this was carried out is that the team set up various fictitious scenarios, preferably with regard to the functional requirements, so that it could test whether the functionality had been implemented correctly. The team also got friends to test the application, by giving them various tasks, and asking them to provide constructive feedback afterwards. In the article "Putting the user in user-interface testing" (Nielsen, 1996, p. 89), the author writes about the importance of the test-users corresponding to the application's intended target group. In this way, the application is not only tested by people with a technical background, but also by people outside the development team who will actually use the application. Feedback from test-users helped the team to improve functionality that had already been implemented. These are improvements the team would rather not have discovered by themself. User testing is less time-consuming than writing automated tests, and also does not require the testers to have knowledge in writing such tests. This suited our group well as no one had much experience in writing tests, and thus we saved a lot of time that could be better spent on fixing bugs and further developing the application.

Another reason for why the team did not use any unit tests is because most of the functions and functionalities in this application is easy to test by using the application. For example, to ensure that the sorting functions for sorting the City List by distance and population can simply be checked by looking at the list in all possible scenarios.

7 - Issues encountered

Finding a database containing Italian cities, including both positional and population data for FREE turned out to be harder than expected. There was a moment where we had to consider adding the cities manually, but this would take a ridiculously long time, so we kept looking. Eventually we found the website OpenDataSoft, where the data could be downloaded for free

8 - Further development

Due to time constraints and the magnitude of this course the team had to set a limit to when the development should end. The team is happy with the end result and learned a lot about developing mobile applications which was the goal of this course. When it comes to further developing the application the team sees it possible to expand the application's functionalities and map coverage. Because of the application's architecture and non-functional requirement of modifiability, adding additional features should be simple. Extending the map with new cities from other countries should be simple work, but requires more data. The team would be interested in looking into an external data storing system like Firebase for saving storage space. The team would also be interested in using Firebase for user authentication, for adding new features where the user can save places they want to visit.