POLITECNICO
MILANO 1863

# Smart Tourist

**Design and Implementation of Mobile Applications**
**Design Document**

Fabio Codiglioni, Alessandro Nichelini

A.Y. 2019/2020

# Contents

# 1 Introduction

This is the *design document* (DD) of the ***Smart Tourist*** iOS application developed by *Fabio Codiglioni* and *Alessandro Nichelini* in the context of the *Design and Implementation of Mobile Application* course at Politecnico di Milano. The authors have been tutored by Bending Spoons for the usage of some technologies that are described later. The document explains the most important design choices we made and the motivations behind them, with specific focus on the Redux-like architecture adopted.

# 2 General Overview

Smart Tourist is a multi-device application for iOS and iPadOS.

## 2.1 Assignment

For the reader convenience, the (free) assignment is reported below:

**Assignement:** Smart Tourist

Help your users when they are exploring new cities!

By using the localisation systems on the smartphones and the google places API, notify the user when something interesting is nearby.

Connect to wikipedia, to retrieve interesting informations on the point of interest selected.

Search and filters among several interesting places of the city you are visiting, save them as favourites so you can check on them later.

Attach some pictures to those pins and eventually share them with your friends!

## 2.2 Features description

### 2.2.1 Exploring

The user is given with a navigation map filled with the city's point of interest. She has the chance to control which attraction to see on a map. She can choose between *Nearest places*, *Popular places* and *Favourites*. Each attraction is shown on the map and in a table view. They are clickable from both position to open the corresponding detail view.

In the map, two circles are displayed to help the user better understand distances. These circles correspond to the distance she can cover in 5 and 15 minutes. Circles' radius is automatically updated with information from user profile and thus they fit always with user's pace.

### 2.2.2 Dynamical exploring

SmartTourist works well in both crowded cities and small towns. The user will be always provided with a right amount of attractions. Where not so many attractions are available, the app will automatically show her less known attraction and particular point of interest.

### 2.2.3 Learning

The user can open the detail view of both the city and attractions. She will be provided with useful information about the point of interest such as pictures, Wikipedia description, useful links and a shortcut to open turn-by-turn navigation.

### 2.2.4 Notifications

The user is notified when she is nearby a "top location". The notification comes with a picture of the attraction and lets her either to open the the detail view of the attraction or to open turn-by-turn navigation.

### 2.2.5 Keep track of favourite attractions

The user can view the attraction of her current city or she can have a preview of other cities attractions. In both case she can keep track of them by adding them in the list of favourite attractions. The user is also provided the ability to open a worldwide map with all her favourite places.

### 2.2.6 Contribute (proof of concept)

SmartTourist is manly based on free data. When a detailed description is missing, the user is given the chance to contribute.

# 3 Architectural design

SmartTourist is a self-contained application. No external backend services are provided by us, instead it relies on a multitude of internal iOS/iPadOS services and third parties API.
Here it follows a schema that summarizes them all:

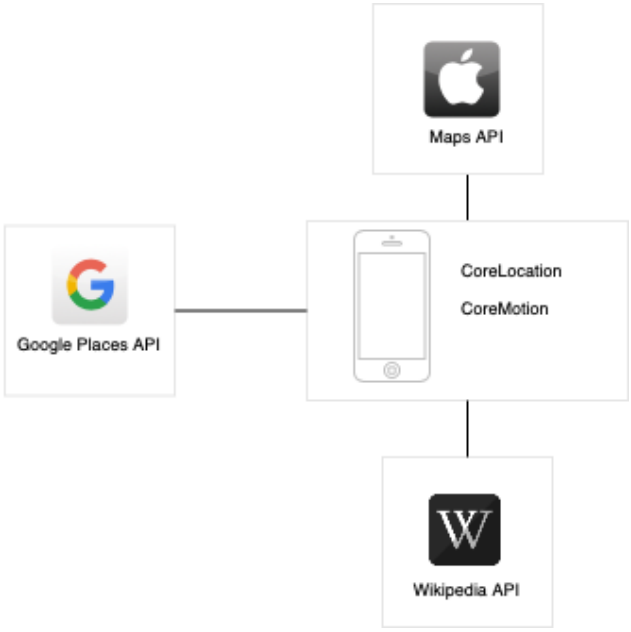


Figure 3.1: Architecture overview

Internal application design is strongly influenced by the choice of the Katana/Tempura couple that forced the adoption of a MVVC paradigm.

## 3.1 Katana

Katana is a modern Swift framework for writing iOS applications' business logic that are testable and easy to reason about. Katana is strongly inspired by Redux.

In few words, the app state is entirely described by a single serializable data structure, and the only way to change the state is to dispatch a StateUpdater. A StateUpdater is an intent to transform the state, and contains all the information to do so. Because all the changes are centralized and are happening in a strict order, there are no subtle race conditions to watch out for.

## 3.2 Tempura

Tempura is a holistic approach to iOS development, it borrows concepts from Redux (through Katana) and MVVM.

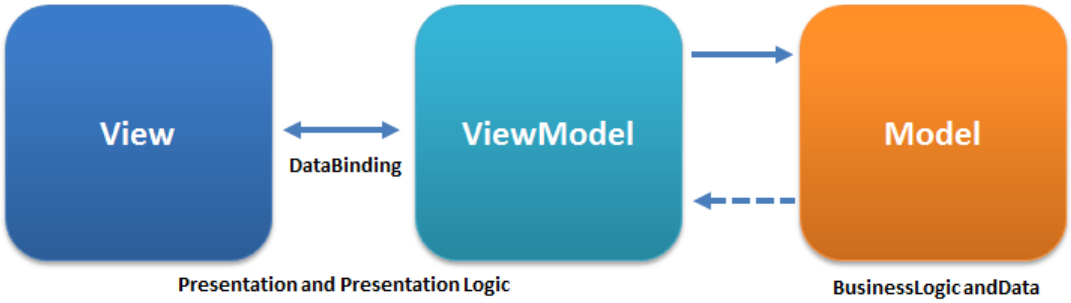


Figure 3.2: MVVM architecture overview

**Warning:** Katana and Tempura are from Bending Spoons and they tutored the authors to use it.

# 4 Services and libraries

## 4.1 Internal libraries

SmartTourist uses consistently internal libraries and services:

- ***MapKit***: for location gathering and maps.
- ***Core Motion***: for accessing accelerometer, gyroscope, pedometer, and environment-related events.

## 4.2 External services

SmartTourist relies almost only on *free data and services*.

> **Warning:** The choice to rely on free services has been carefully pondered and it cost the authors some troubles. SmartTourist has been designed to be a free application in an eventual public launch thus it would be impossible to support the cost of Google Places API without any premium subscription. Moreover, the author strongly believe in the Wikidata project and wanted to test it.

Here, it follows the description of the external services used.

- ***Wikipedia***: all displayed attractions are taken from the Wikidata knowledge base. The given API is quite basic: attraction entries are retrieved with a SPARQL query embedded in a http API call. Details and pictures are retrieved with standard HTTP API calls from different endpoints.

```
1
2         SELECT DISTINCT ?place ?placeLabel ?location ?image ?instance ?
              phoneNumber ?website ?wikipediaLink
3         WHERE {
4             SERVICE wikibase:label { bd:serviceParam wikibase:language "en,
                  it" }
5             SERVICE wikibase:around {
6                 ?place wdt:P625 ?location .
7                 bd:serviceParam wikibase:center "Point(\(location.longitude)
                      \(location.latitude))"^^geo:wktLiteral .
8                 bd:serviceParam wikibase:radius "\(radius)" .
9             }
10            ?place wdt:P31 ?instance  .
```

```
        ?place wdt:P18 ?image .
        OPTIONAL {?place wdt:P1329 ?phoneNumber}.
        OPTIONAL {?place wdt:P856 ?website} .
        OPTIONAL {?wikipediaLink schema:about ?place;
            schema:inLanguage "en";
            schema:isPartOf [ wikibase:wikiGroup "wikipedia" ]} .
        }
```

Listing 4.1: SPARQL query for attraction retrieving

```
            SELECT DISTINCT ?city ?cityLabel ?country ?countryLabel ?
                population ?area ?elevation ?link ?facebookPageId ?
                facebookPlacesId ?instagramUsername ?twitterUsername ?image ?
                coatOfArmsImage ?cityFlagImage ?countryCode ?wikipediaLink
                WHERE {
            BIND( <http://www.wikidata.org/entity/\(cityId)> as ?city ).
            OPTIONAL {?city wdt:P17 ?country}.
            OPTIONAL {?city wdt:P1082 ?population}.
            OPTIONAL {?city wdt:P2046 ?area}.
            OPTIONAL {?city wdt:P2044 ?elevation}.
            OPTIONAL {?city wdt:P856 ?link}.
            OPTIONAL {?city wdt:P2013 ?facebookPageId}.
            OPTIONAL {?city wdt:P1997 ?facebookPlacesId}.
            OPTIONAL {?city wdt:P2003 ?instagramUsername}.
            OPTIONAL {?city wdt:P2002 ?twitterUsername}.
            OPTIONAL {?city wdt:P18 ?image}.
            OPTIONAL {?city wdt:P94  ?coatOfArmsImage}.
            OPTIONAL {?city wdt:P41 ?cityFlagImage}.
            OPTIONAL {?country wdt:P297 ?countryCode}.
            OPTIONAL {?wikipediaLink schema:about ?city;
              schema:inLanguage "en";
              schema:isPartOf [ wikibase:wikiGroup "wikipedia" ]}.
            SERVICE wikibase:label { bd:serviceParam wikibase:language "en".
                }
        }
```

Listing 4.2: SPARQL query for city detail retrieving

- ***Google***: Places ratings are the only data retrieved from Google services. This was due to the lack of a reliable free alternative. For the simplicity of the task, we decided not to rely on the Google SDK for iOS and to manually make request to the Google API.

## 4.3 External libraries

The app uses lots of third parties libraries the can be roughly divided into three kind: architectural libraries, back-end libraries and front-end libraries.

| Kind | Library | Description |
|---|---|---|
| Architectural | Katana | Katana is a modern Swift framework for writing iOS applications' business logic that are testable and easy to reason about. Katana is inspired by Redux. |
| | Tempura | Tempura is a holistic approach to iOS development, it borrows concepts from Redux (through Katana) and MVVM. |
| Back-end | DeepDiff | DeepDiff tells the difference between 2 collections and the changes as edit steps. |
| | Fuse | Fuse is a super lightweight library which provides a simple way to do fuzzy searching. |
| | Alamofire | Alamofire is an HTTP networking library written in Swift. |
| | SigmaSwiftStatistics | It is a collection of functions that perform statistical calculations in Swift. It can be used in Swift apps for Apple devices and in open source Swift programs on other platforms. |
| | SwiftyXMLParser | Simple XML Parser implemented in Swift. |
| Front-end | PinLayout | Extremely fast views layouting without auto layout. No magic, pure code, full control and blazing fast. Concise syntax, intuitive, readable and chainable. PinLayout can layouts UIView, NSView and CALayer. |
| | FlexLayout | Angular Flex Layout provides a sophisticated layout API using Flexbox CSS + mediaQuery. |
| | Cosmos | This is a UI control for iOS and tvOS written in Swift. |
| | ImageSlideshow | Customizable Swift image slideshow with circular scrolling, timer and full screen viewer. |
| | MarqueeLabel | MarqueeLabel is a UILabel subclass adds a scrolling marquee effect when the text of the label outgrows the available width. |
| | FontAwesome | Use Font Awesome in your Swift projects. |
| | FlagKit | Beautiful flag icons for usage in apps and on the web. |

# 5 User interface

For the user interface design, we followed Apple’s UI guidelines, but we preferred also to use custom UI element to better fit Katana/Tempura paradigm’s requirements and to let the app have a endearing style.

## 5.1 Screenshot

Here follow some screenshot of the application in both Light and Dark mode.

### 5.1.1 Mapview

#### Mapview - nearest

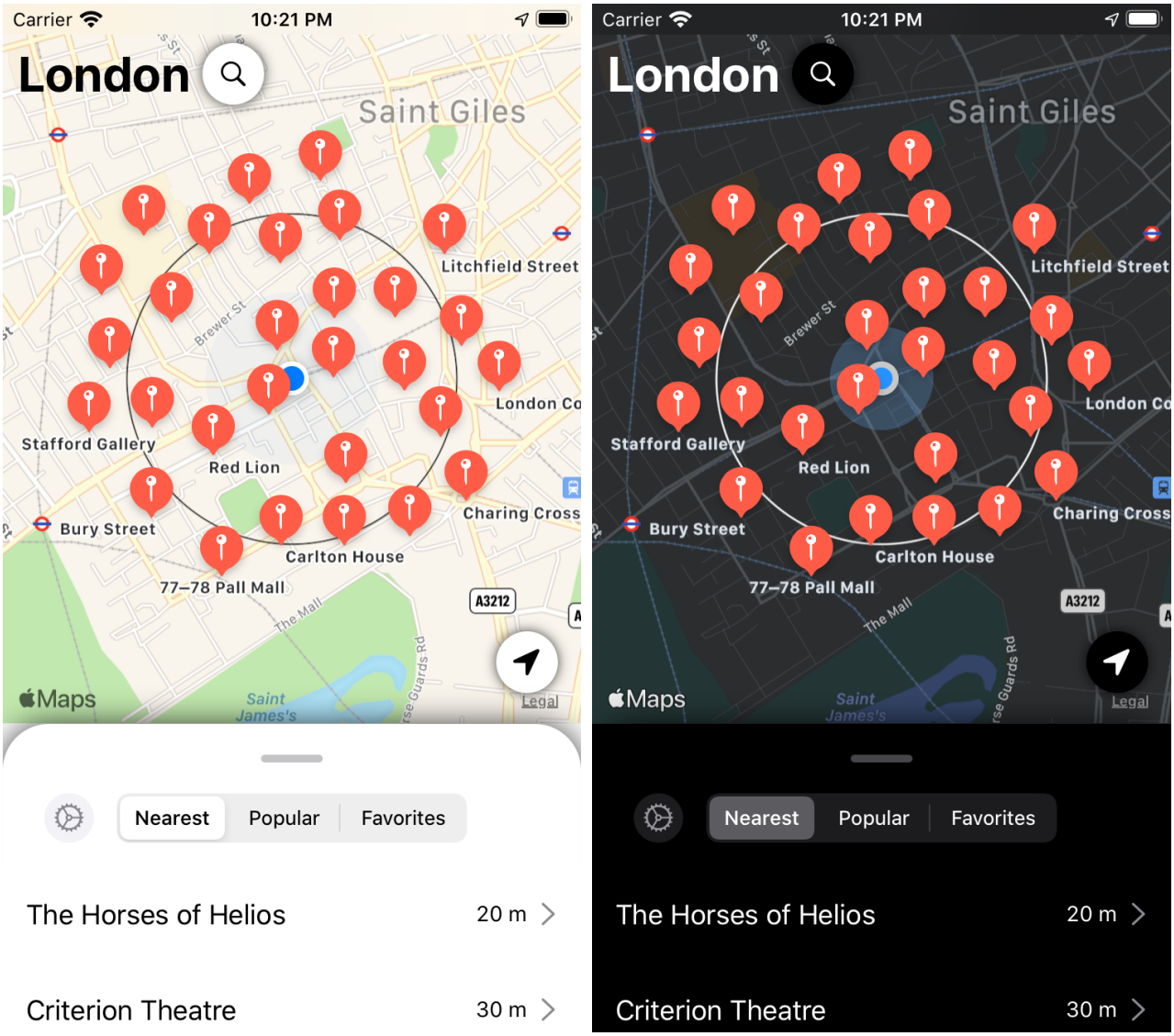


Figure 5.1: Map view with nearest tab selected in both light and dark mode

This is the first view that appears when you open the app. Attractions, up to the maximum number selected, are displayed both in the maps and in the list below. In the list they are ordered by means of their distance to user current position.
In this view you can tap the city name to open the city detail view or an attraction to open the corresponding detail view. By pressing the search button, the user is presented with the common location search bar. Settings are reachable by pressing the gear button near the selectors.
The tab view for attraction list id resizable.

### 5.1.2 Mapview - popular

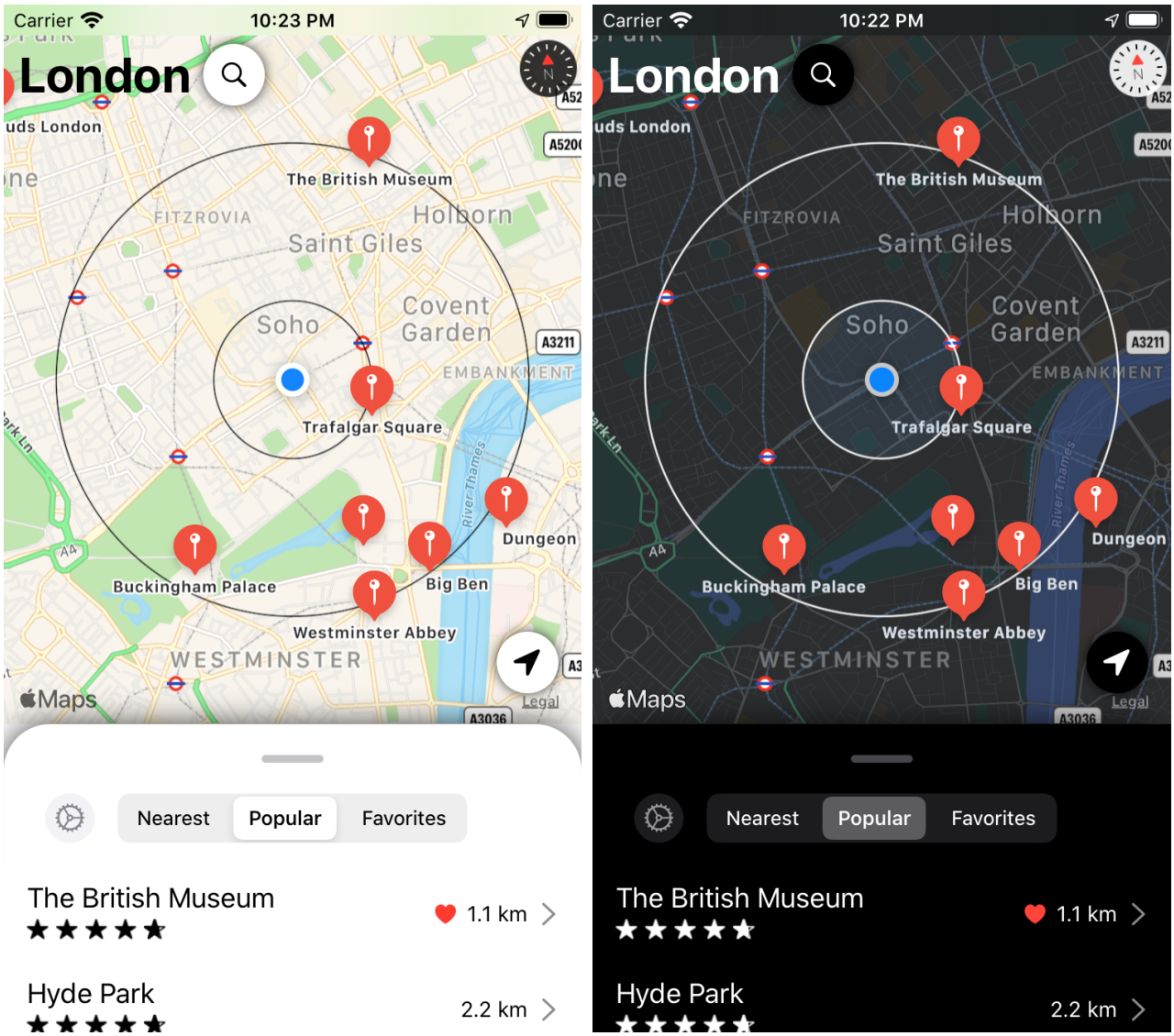


Figure 5.2: Map view with popular tab selected in both light and dark mode

Same as previous view, but here popular attraction tab is selected. Thus only most popular places are displayed. In the list, favourite items are identified by a hearth icon and the rating of each attraction is also displayed.
In this screenshot, circles that represent distance are more visible. They respectively represent the distance that the user can cover in 5 and 15 minutes of walking at her current walking speed.

### 5.1.3 Mapview - favourite

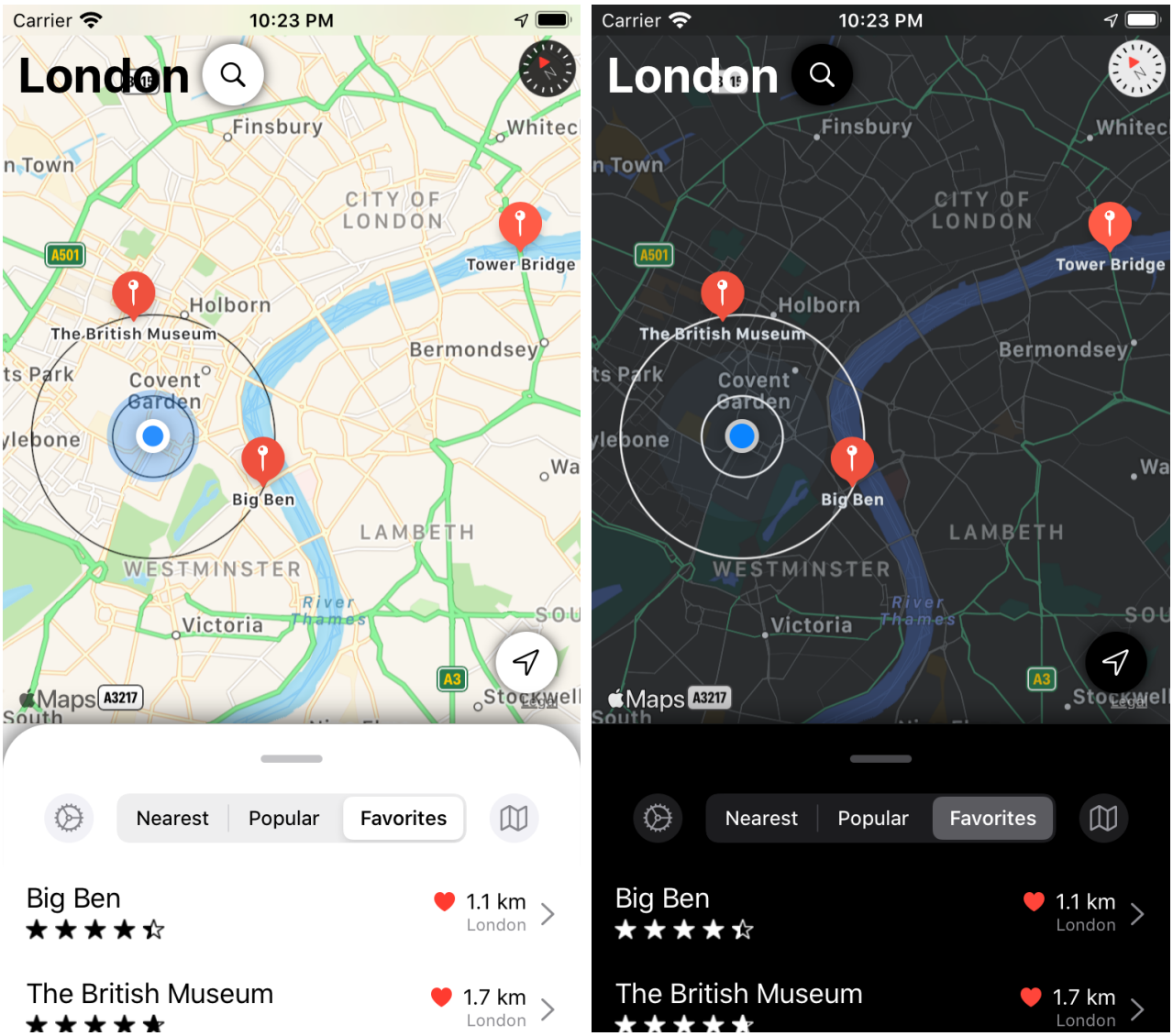


Figure 5.3: Map view with favourite tab selected in both light and dark mode

Same as previous view, but here favourite attraction tab is selected. Since it's a global tab, they are grouped by city, that is also displayed.
An additional button with a map is displayed, it lets the user to access the global view of favourite items.

### 5.1.4 Global favourite view

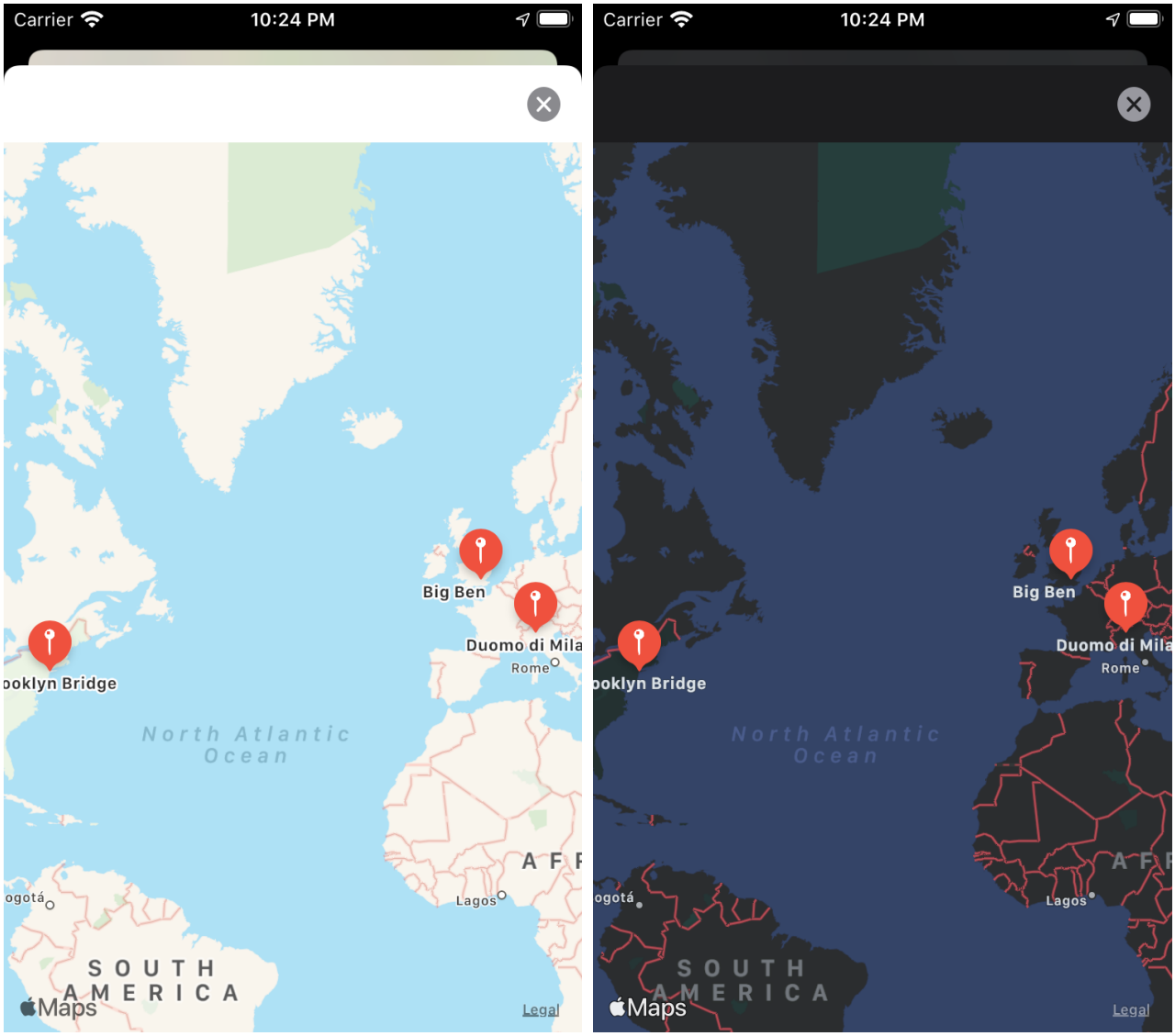


Figure 5.4: Favourite worldwide view in both light and dark mode

In this view, all user's favourite attractions are displayed in a map with fixed zoom.

### 5.1.5 Searching view

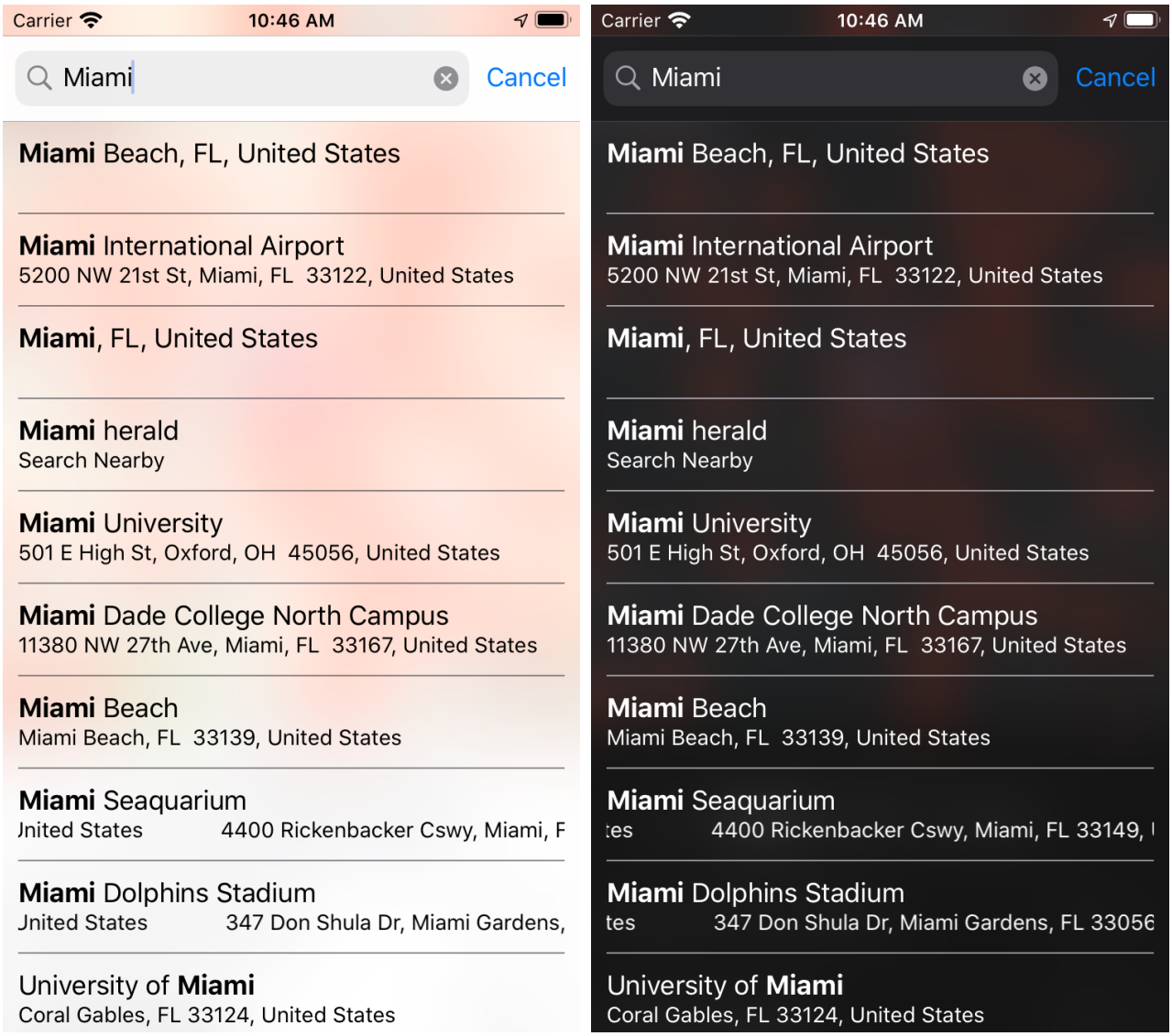


Figure 5.5: Searching view in both light and dark mode

When you press the lens button on the map view, the common searching view provided by iOS is opened. Both cities and attraction can be searched.

### 5.1.6 Attraction detail view

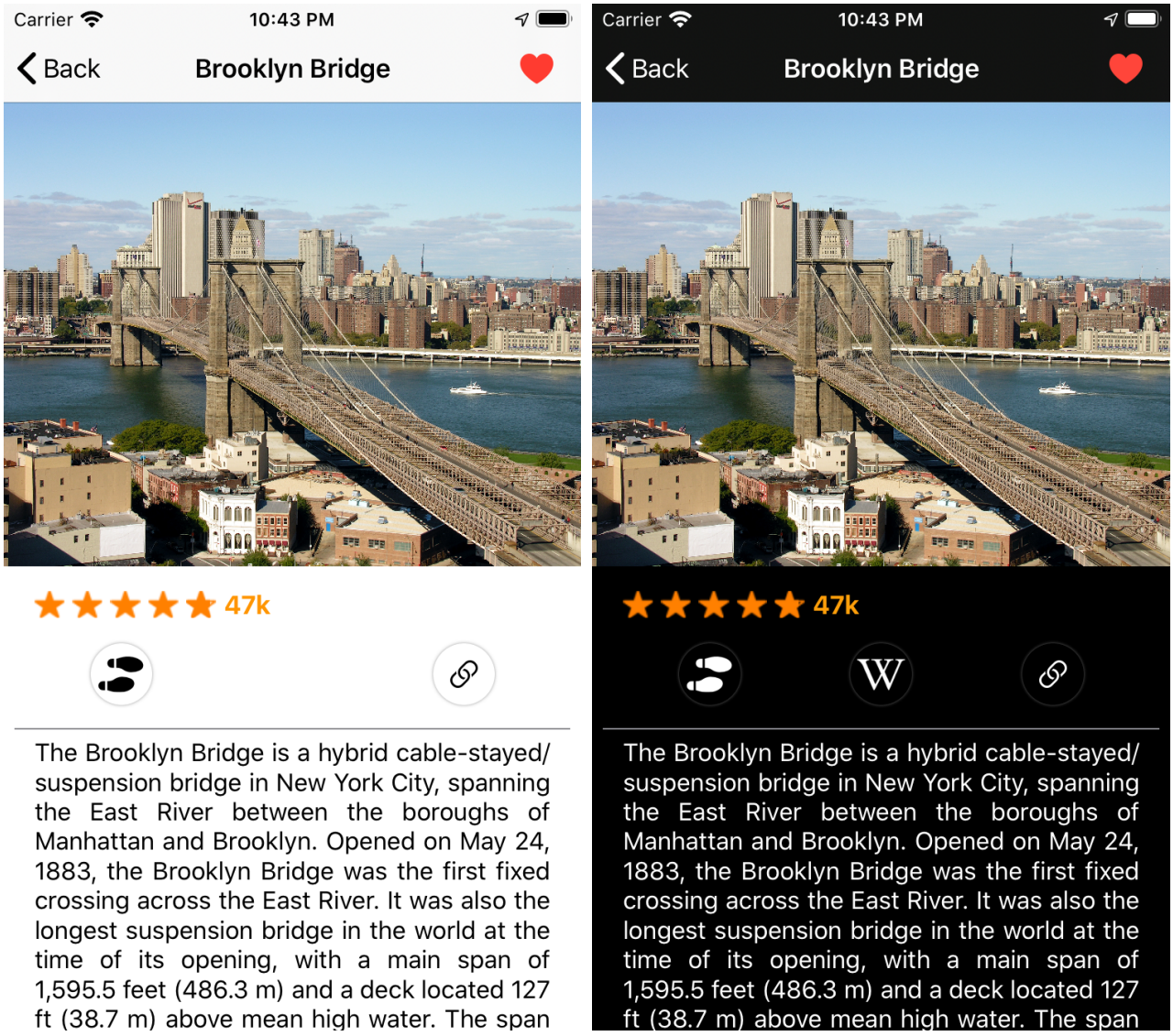


Figure 5.6: Attraction detail view in both light and dark mode

This is the detailed view that is presented when the user tap a specific attraction. A slideshow of picture of the attraction is displayed together with the rating (if available) and buttons that let you access the map direction, the complete Wikipedia Article and the related website. In the navigation bar, the back button is available together with the hearth button that let the user favourite the attraction. The view is scrollable to display the entire place description extract. A detailed map with the specific attraction location is also provided at the end.

### 5.1.7 City detail view

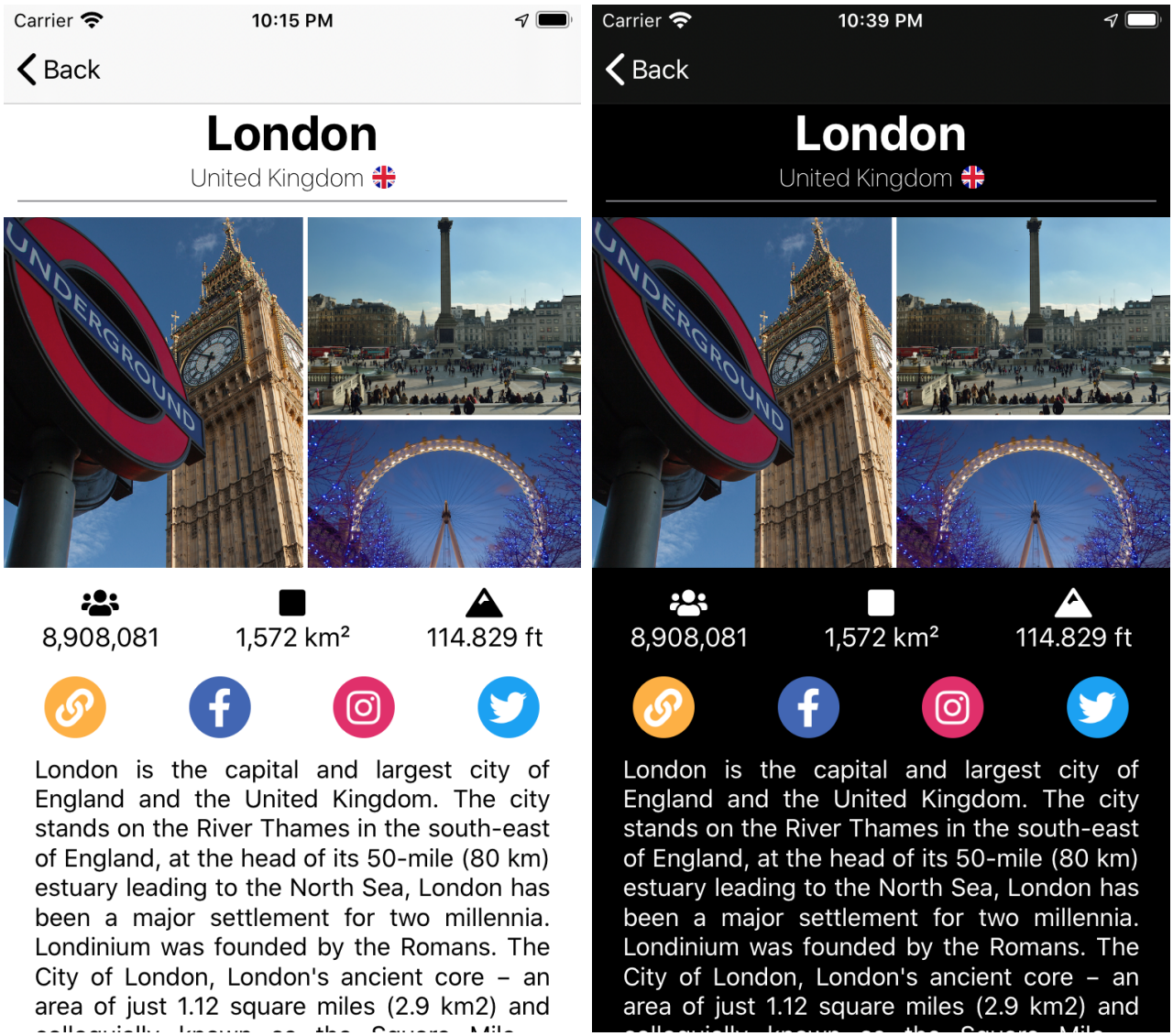


Figure 5.7: City detail view in both light and dark mode

This is the city detail view that is displayed when the user tap the city name in the main view. A pictures slideshow is displayed together with some information such as: population, area and altitude. A set of social link is retrieved and presented to the user. The view is scrollable and a summary description of the city is given.

### 5.1.8 Settings view

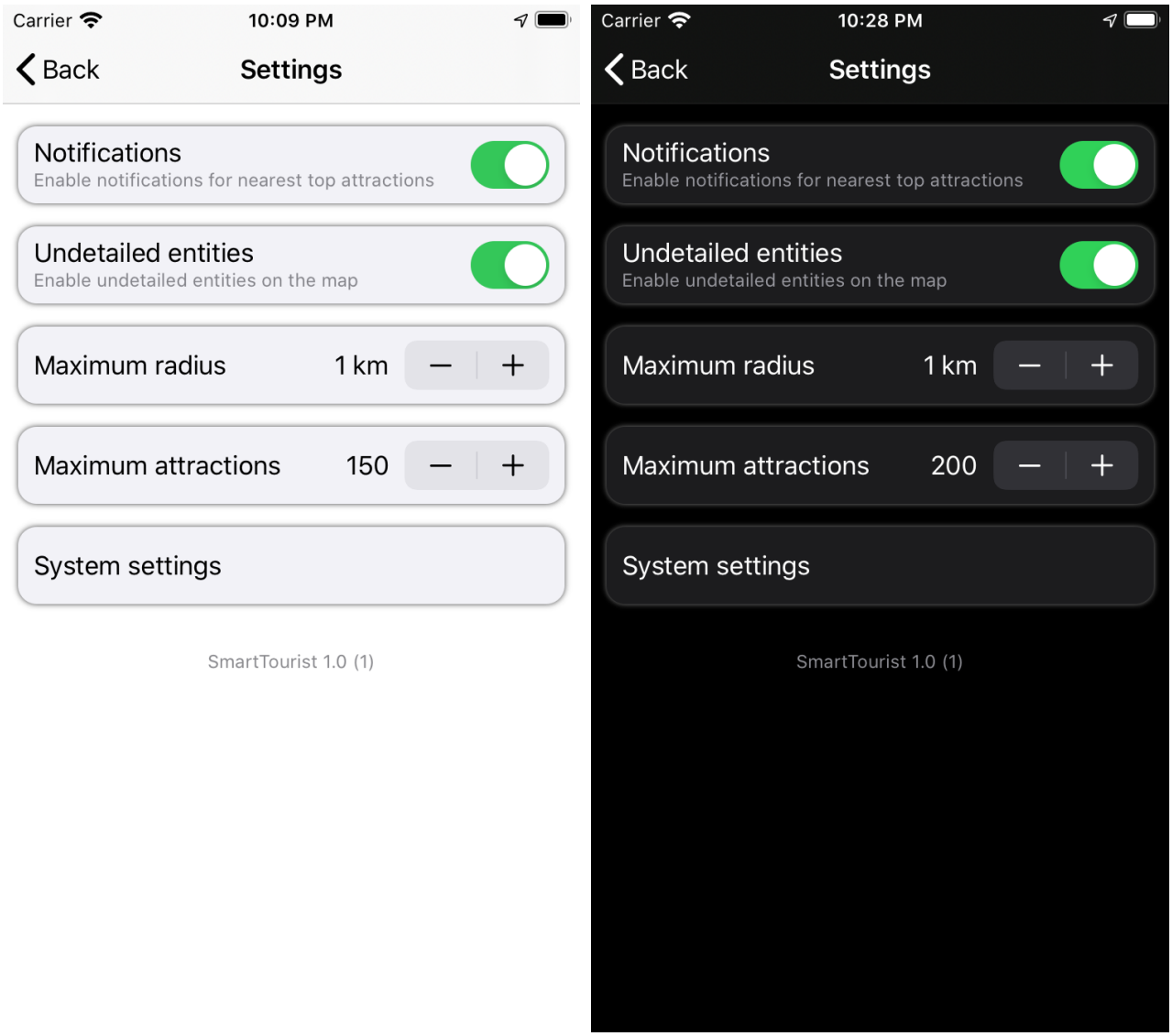


Figure 5.8: Settings view in both light and dark mode

Settings view is displayed when the gear icon on the main view is displayed. This is a quite standard settings view where some parameters about the application can be set.

# 6 Notifications

Notifications are designed as a way for the user not to miss popular attraction near her. Each time the user is in a 5 minutes range of a popular attraction, the notification is triggered. This works in both background and foreground mode ("*always active*" location tracking is required).
Notification is a *rich notification* with callbacks: **take me there** will directly open navigation app with that attraction as destination and **view** will open SmartTourist detail attraction view.

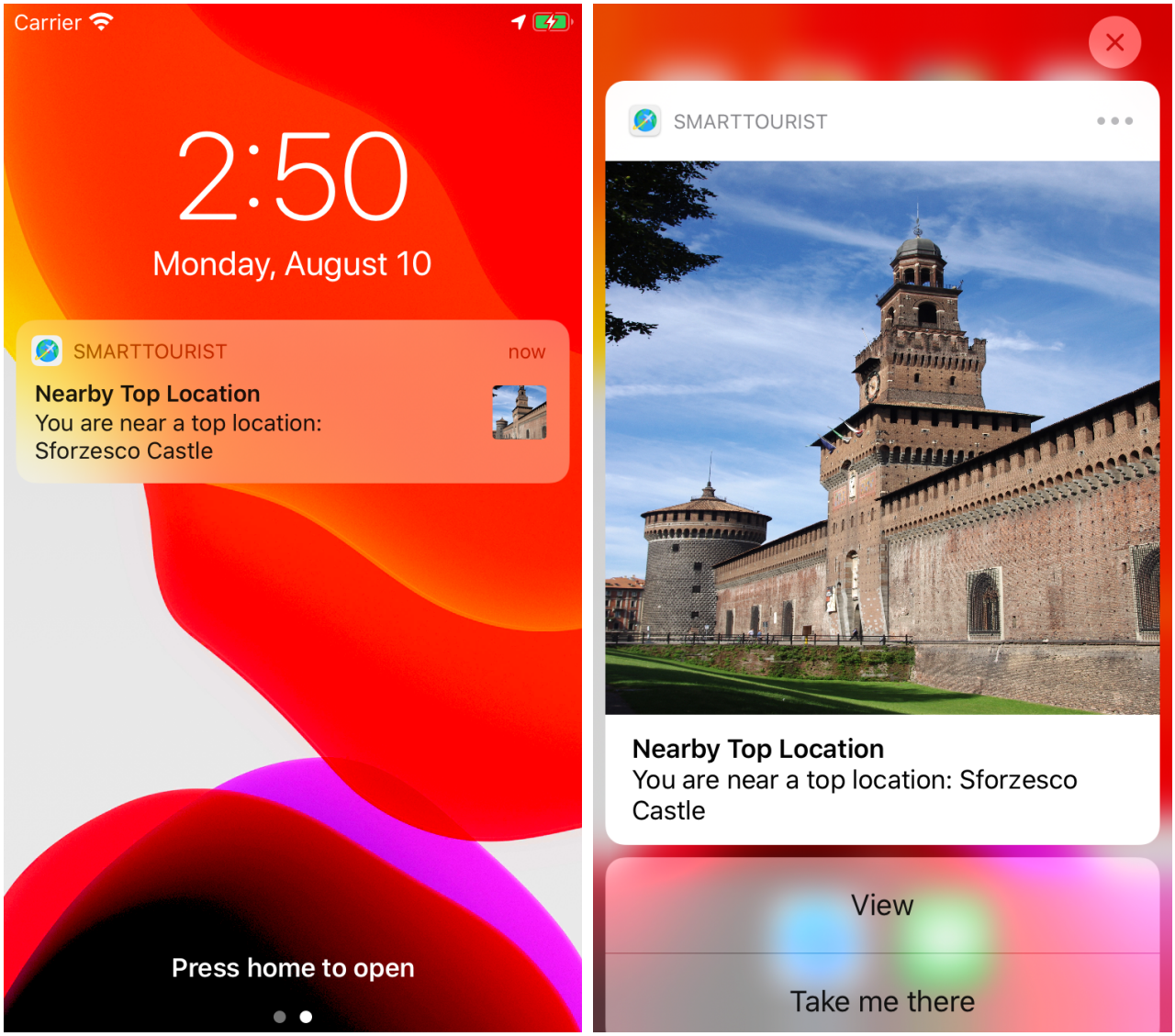


Figure 6.1: Notification layouting

# 7 Testing

# 8 Effort spent

The actual development took place between January 2020 and May 2020 while during July/August 2020 just some minor changes have been put in place together with the drafting of all the needed documents and materials.

The authors worked together on site and remotely to setup the crucial parts of the application, then, they worked mostly on their own with continuous mutual contacts. They also had a couple of starting tutoring session kindly offered by Bending Spoons at their GQ to learn the Katana/Tempura paradigm.

They didn’t use any professional time tracker but it’s safe to assume that the total amount of work spent is around 300 per worker:

- 20% platform greet and reet
- 40% actual development
- 15% free services refactoring
- 10% testing
- 20% documents drafting