



POLITECNICO

MILANO 1863

Smart Tourist

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

Fabio Codiglioni, Alessandro Nichelini

A.Y. 2019/2020

Contents

1	Introduction	1
2	General Overview	2
2.1	Assignment	2
2.2	Features description	2
2.2.1	Exploring	2
2.2.2	Learning	3
2.2.3	Notifications	3
2.2.4	Keep track of favourite attractions	3
3	Architectural design	4
4	Data design	5
5	User interface	6
6	Notifications	7
7	Services and libraries	8
7.1	Internal libraries	8
7.2	External services	8
7.3	External libraries	8
8	Testing	10
9	Effort spent	11

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 assignment is reported below:



Assignment: 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 *Near-est 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 circle are displayed to help the user understand better distances. These circles correspond to the distance she can cover in 5 and 15 minutes. Circles' radius is automatically updated with information from MotionKit and thus they fit always with user's pace.

2.2.2 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.

2.2.3 Notifications

The user will be notified when

2.2.4 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.

3 Architectural design

4 Data design

5 User interface

6 Notifications

7 Services and libraries

7.1 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.

7.2 External services

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

7.3 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.
	SwiftXMLParser	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.

8 Testing

9 Effort spent