Technical Project Report - iOS Module

GoFindIt!

Date: Aveiro, 18 January 2017

Authors: 64090: Rui André da Cruz Lebre
59925: Leonardo Fernandes Pinheiro

Project abstract: GoFindIt! is an iOS application conceived to support peddy pappers. After the proposal from Mobile Computation professor to develop an application for use in a event place, which should be able to store information on a specific location (tagged with iBeacon and/or QR Code), with the possibility to add photos associated with the location and store it locally, we thought about to improve it.

The final project proposal consisted in a peddypaper event standalone app (with the possibility to, eventually, make use of web services, web storage and remote databases), that could offer users the ability to create events, edit it, remove it, add places that should be visited (associated with the beacon identifier) and the ability to take photos and associate it with the current location and current event.

The following text reports all the decisions that had been taken along the development, as well as the architecture and layout of GoFindIt!

Table of contents:

1 Introduction

This project was developed under the supervision of Prof. José Maria Fernandes in the context of the subject Computação Móvel (Mobile Computing) between end November and mid-January of the academic year of 2016-2017, a class of the course BSc/MSc Engenharia de Computadores e Telemática, University of Aveiro (www.ua.pt).

We were proposed to develop an iOS application and together with the professor we defined our goals. The main goal of our application is to create a route with several points to visit during a trip. This points are identified as beacons. Users can also take pictures during their trip and save them bound to the trip.

2 Application concept and scope

GoFindIt is an application conceived to every user who likes to travel, visit new places and spend some time out. With our application, users can easily visit new unknown places by creating a given route. When creating a new event, user introduces the beacon information which represents the places to visit during his trip and its location. Finally, he sets his start point.

In this way, a user can see on the map where each point is located and how to access them. Finally, when a user finds a beacon he/she must scan his QR Code to validate that he really visit the spot, in this way we guarantee that during a trip our users really visit the best spots of a city.

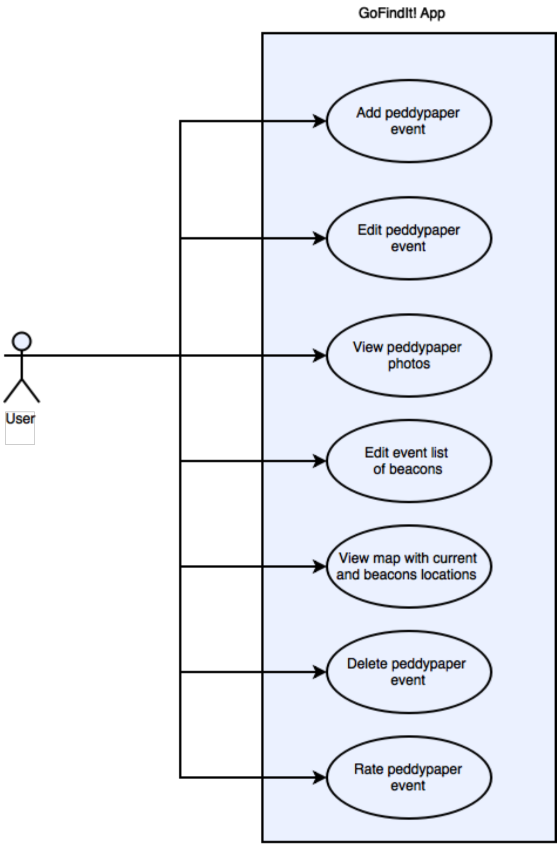


Figure 1 - GoFindIt! use case

3 Technical design options

User experience design

We are not responsible for the paths the user decides to use to access the beacons and go through during the event. The application aims at providing a clear user interface and experience throughout its usage. In this way we created a simple user interface to facilitate the usage of everybody. After launching the application the user will directly access his events and can easily create new events, in this way we keep our main scoop in the foreground.

Our application icon is composed of a merge of a magnifying glass which is seen as a symbol of looking for/searching some thing and a map icon, which represents the art of travelling. Inside we can find a indicator of a direction which represents a objective to conclude. The combination of all this objects as we can see on the picture bellow gets us to the name of our application GoFindIt.

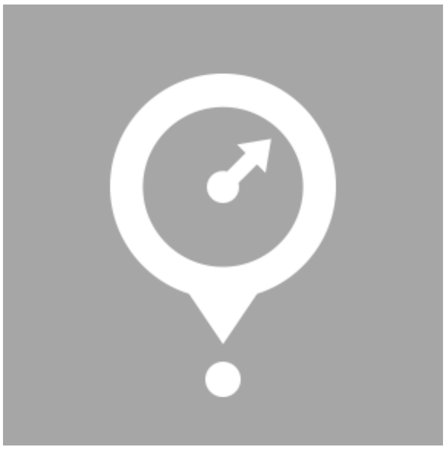


Figure 2 - GoFindIt! logo

The gray color stands for independency, which represents the independency of a touristic guide for visiting an unknown place, like reported before one of our scoops is to facilitate a trip.

→ rationale for the overal look&feel/interaction patterns

→ explain the design concepts (if applicable): what is the application icon and why? wich is the color pallete an why?...

→ you may add prototype views (screenshots from the prototyping design process, the early sketch for application navigation, etc)

→ compromises: ideal vs what was doable

Technical design

Like no one of use ever used a iOS device we began our researches by installing applications the we use every day on our Android devices, with the goal of getting some ideas of the designs and structures that this applications are offered to the users.

During the development of our application we focused on the defined goals. So we implemented the spots using beacons. The beacons can be added in three different ways, we offer users the possibility of adding a beacon manually, by bluetooth, or by scanning a QR Code. After a long search of frameworks and look at any kind of code, we decided to use the kontact.io as framework to implement our beacons.

For the start location of the event and each beacon we decide to use MapKit provided by Xcode, in order to pin the the desired place.

We also allow users to associate a picture as sample of the event and take pictures during the trip which will be saved locally on the device. Each picture can be visualized as many times the user wants and also see on the map the exact location where he tooks the picture. All this information is stored using persistence, CoreData.

→ Discuss architecture planned for the software solution. Include diagrams.

→ supporting data models/data structures, data management strategy & content update and synchronization strategies

→ more advanced app design issues: integration with Internet-based external services, data synchronization strategy, distributed workflows, push notifications mechanism, etc.>

4 Implemented solution in iOS

Figure 3 GoFindIt! splash screen

GoFindIt! is a peddypapper iOS application. With our application, users can create several routes/event they want to do in a some city or location

To create a route, users must give a name to the route and pick the start location on the map of the new route (Figure 5).

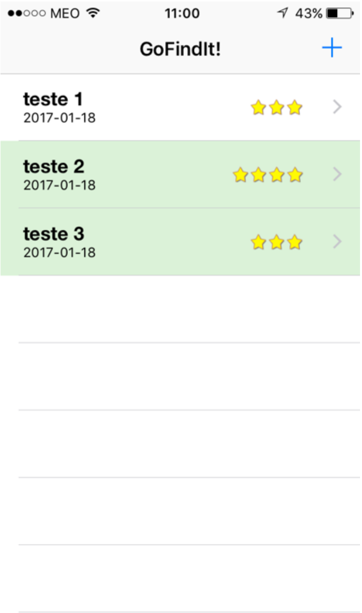


Figure 4 - List of events (completed are in green)

In each route, it is possible to add spots the users must to visit during the ride. The spots are defined as beacons. Users can add beacons in three different ways (Figure 7).

First, user can add beacons using Bluetooth, and selecting the desired beacon, another way is to scan a QRCode of the beacon and last one, the user can add manually the information regarding the beacon.

The first two options are trivial, the last one consists in creating a beacon by identifying it with a name and then link a location to the beacon using maps and picking the desired location. When all of this process is completed, our event is created.

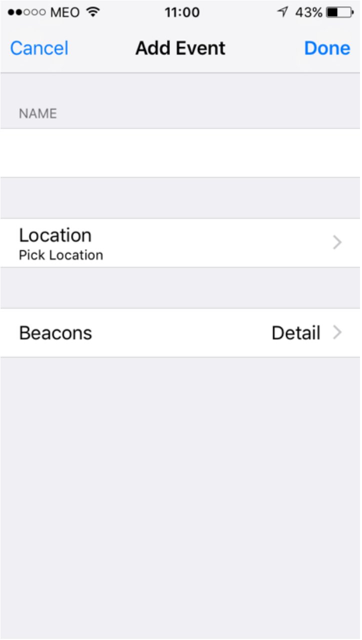


Figure 5 - Creating new event

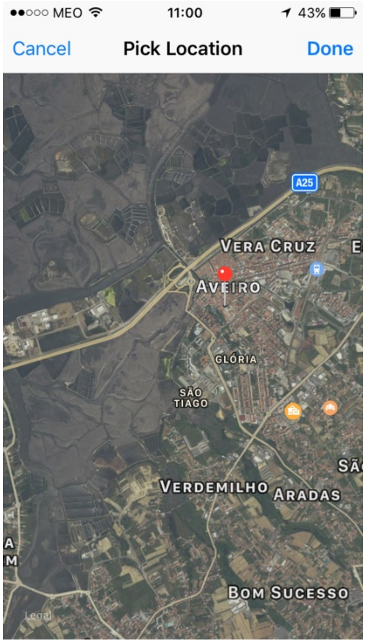


Figure 6 - Picking event location

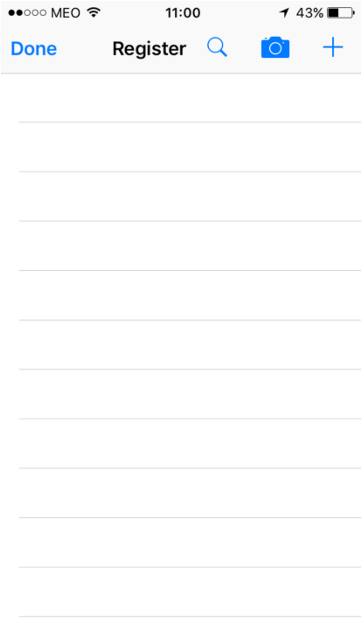
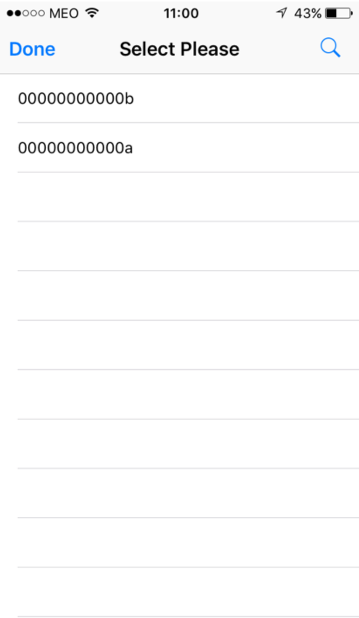


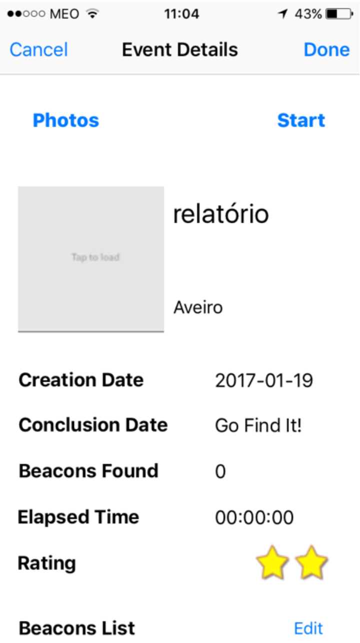
Figure 7 - Registering new beacons



00000000000b

00000000000a

Figure 8 - List of Bluetooth Low Energy beacons



relatório

Aveiro

Creation Date	2017-01-19
Conclusion Date	Go Find It!
Beacons Found	0
Elapsed Time	00:00:00
Rating	★★

Beacons List

Figure 9 - Main event screen, presenting detail

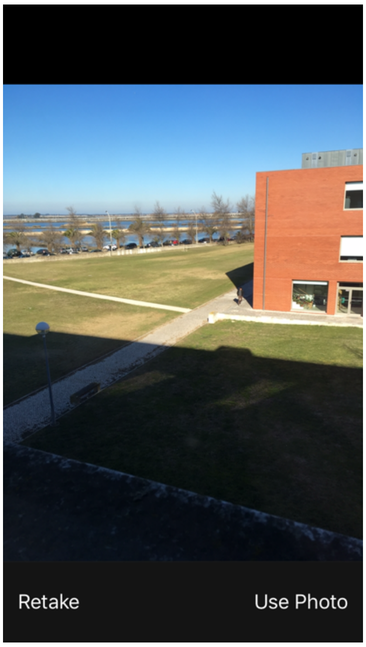
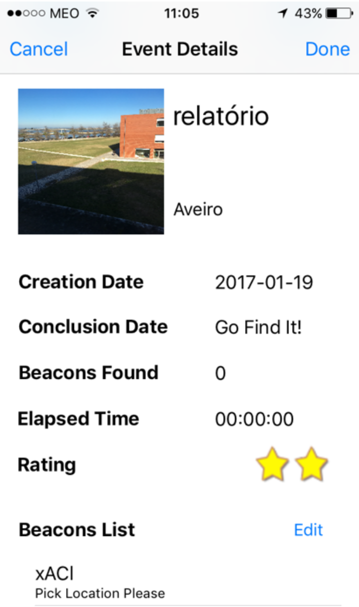


Figure 10 - Taking photo to give event an image



relatório

Aveiro

Creation Date	2017-01-19
Conclusion Date	Go Find It!
Beacons Found	0
Elapsed Time	00:00:00
Rating	★★

Beacons List

xACI
Pick Location Please

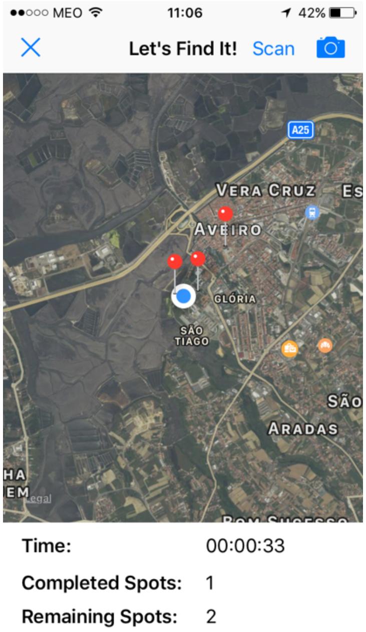
Figure 11 - Main event screen, after editing rating and assign a photo

To try our application, user must select a created event. All events can be edited and customized, by giving a personal rate to the event or even adding/taking a picture related to the event. For instance, in figure 9, it is shown an event specially created to this report and not edited yet. After the action shown in Figure 10, where the user is taking an event picture, the edited report is saved on persistence and turns into the shown in Figure 11.

The user can start his trip by pushing the start button and so start to have fun.

The Map will open and all the beacons will be showed on the map in the attributed location as shown in Figure 12.

On this screenshot (Figure 12), we can see a counter counting the past time. There is also information about the beacons completed and remaining. On top of the screen the user can tap to scan a beacon QR code or to take a photo associated with the current user location.



Time:	00:00:33
Completed Spots:	1
Remaining Spots:	2

Figure 12 - Event running (after 33 seconds running) and 1 beacon already "caught"

If a route is changed, that means beacons are removed or added, the user can update this information by editing his beacon list. The removing of a beacon can be done simply swiping the table cell to the left and then click Delete.

At all the time, the user can check all the taken photos in a photo gallery view. On the top of this Photo Collection View, user can tap on Map to simply view the location on map where the pictures were taken, as we can see on figures below.

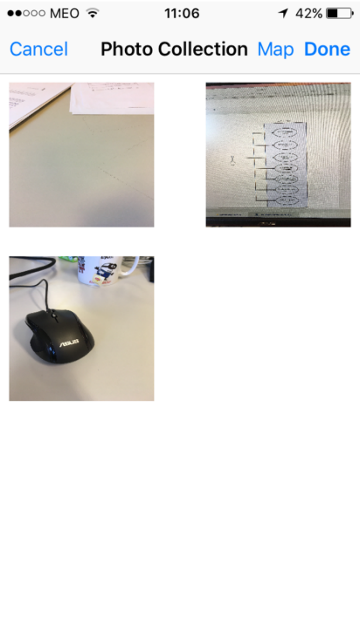


Figure 13 - GoFindIt! Photo collection of a given event

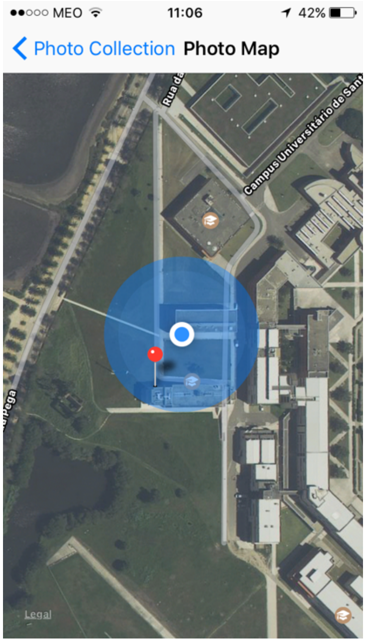
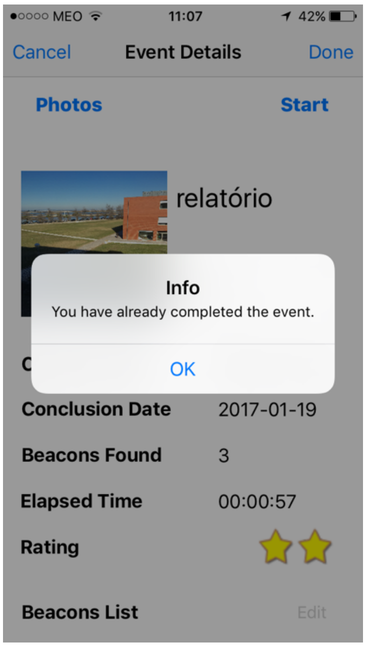


Figure 14 - Map of the photos taken

When a user completes an event, it is shown an information box explaining him that the event has been successfully completed (Figure 15). When the user tries to start the event, a new information box is shown (Figure 16). Besides that, on the event list, the concluded event is shown with green, to differentiate it from the uncompleted events (Figure 4).



relatório

Info

You have already completed the event.

OK

Conclusion Date 2017-01-19

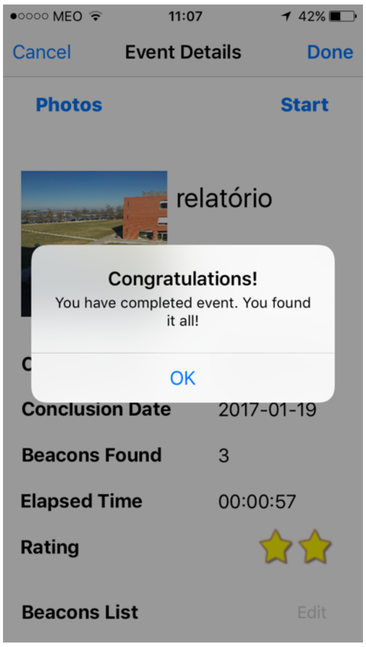
Beacons Found 3

Elapsed Time 00:00:57

Rating

Beacons List

Figure 15 - Information when the user finishes the route



relatório

Congratulations!

You have completed event. You found it all!

OK

Conclusion Date 2017-01-19

Beacons Found 3

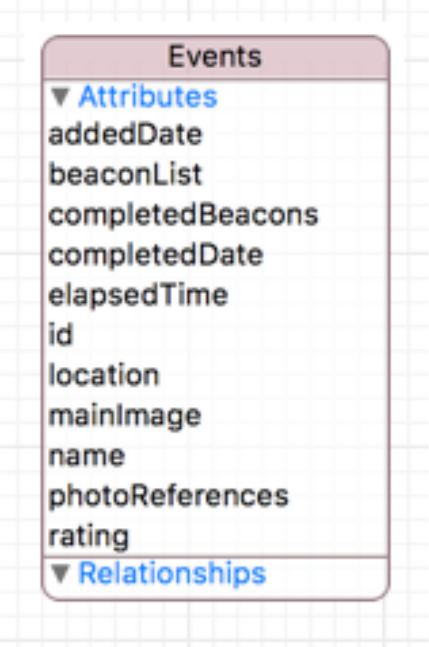
Elapsed Time 00:00:57

Rating

Beacons List

Figure 16 - Information when the user wants to start a new event

The application stores all the data related to the event: main picture, rating, creation date, conclusion date, completed beacons, registered beacons, references to photos that had been taken and its location, the time user spend executing the route and so on. The storage is made with persistence using CoreData, a framework provided by iOS. We can see below the table used to persist the data (Figure 17).



Events

▼ Attributes

addedDate

beaconList

completedBeacons

completedDate

elapsedTime

id

location

mainImage

name

photoReferences

rating

▼ Relationships

Figure 6 - CoreData persistence data model

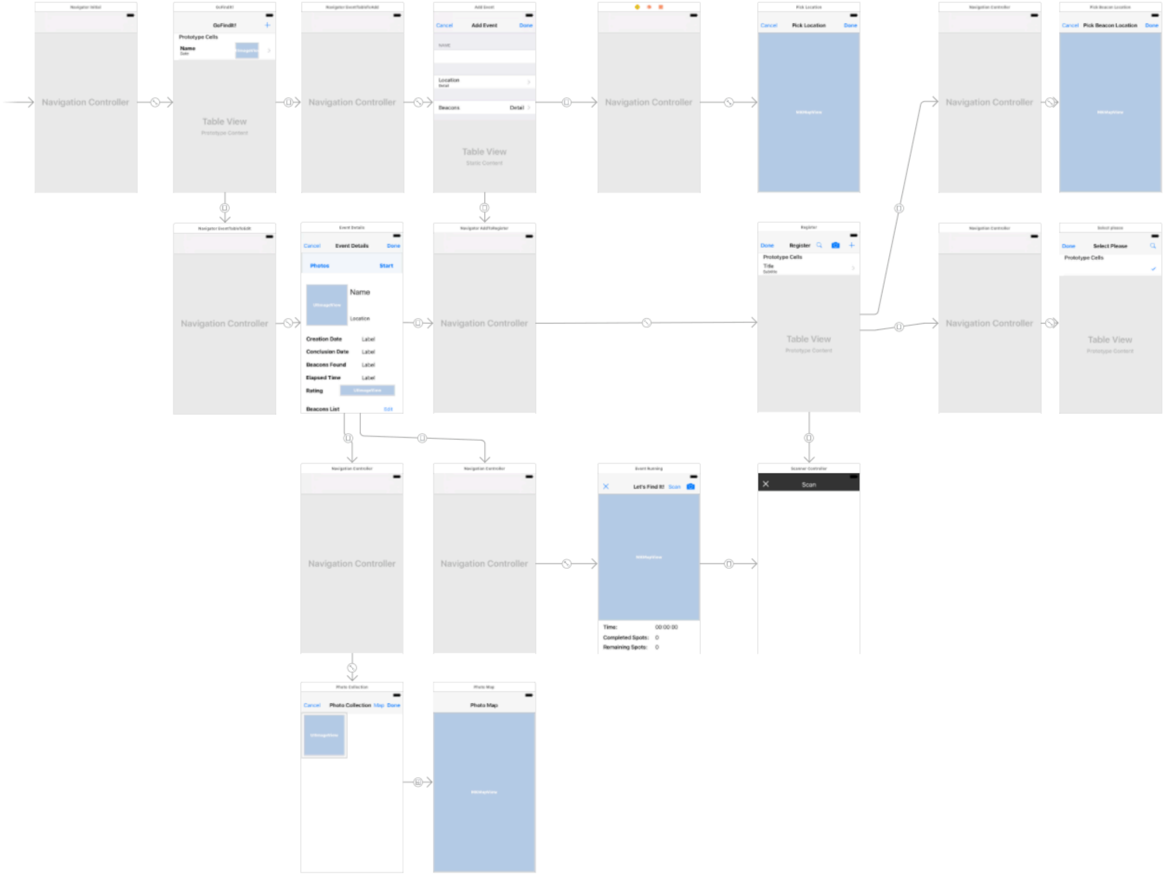


Figure 7 - Application overall scheme

5 Final Notes

To start developing this application, we searched on AppStore for applications related or not to ours to find out how to design an application in iOS. Applications like Publico or SofaScore were installed on the provided equipment (iPhone 6 SE) and tested.

Relating to beacons, there was made an intense research about iBeacons and Eddystone development frameworks. At the end, there was used Kontakt.io framework, the SDK provided from the received beacon manufacturer.

6 Conclusion

During the development of our application we had to face with a problem concerning the search of beacons with Bluetooth. Sometimes our application can't find beacons without a concrete reason. During the debugging, we can conclude that the bug is provided from the framework, since the running cycle stays in loop on the first line which is a print function.

During the presentation, we were informed by the teacher that the problem can be related to the low battery of the beacons, since they already have two or three years.

For future work, the application could be able to share events and use online storage of data, that would improve the usability and might even change completely the application success.

One of the recommendations for a future edition of the CM course is to change the way the projects are defined. We think that it would be more productive to define the projects at the beginning of the year together with the two teachers, of Android and iOS. In this way it would be much more easier to define goals and to create the same application for both operating systems, in order to have a real perspective of the differences of both.

7 References and resources

Frameworks used:

- KontaktSDK.framework
- CoreData
- CoreLocation
- MapKit

Project resources:

Project resources for the Android module:

- Code repository: https://github.com/rlebre/goFindIt.git