ODTUG GeekAthon Solutions Document
Team: Father and Son

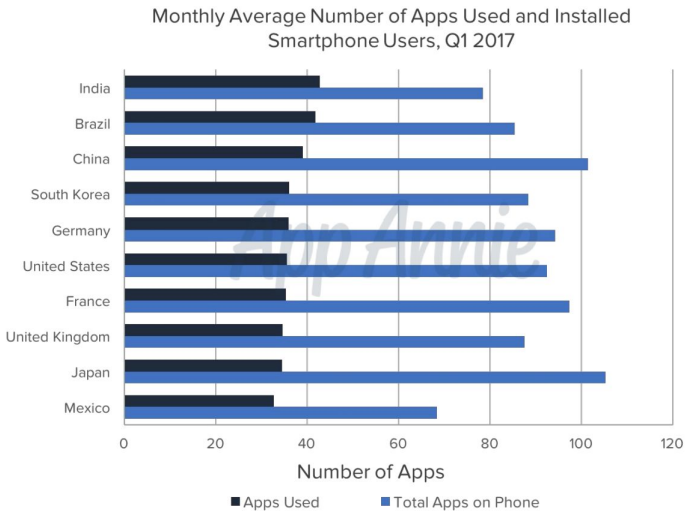
This document describes the entry for the ODTUG GeekAthon Competition involving the use of iBeacons.

Team
The consists of Michiel and Dick Dral.

Michiel Dral is almost 21 years of age, but already a seasoned programmer. At primary school he started his IT career with GameMaker. At thirteen years of age he got into jailbreaking iPods and iPhones. At fifteen he had his own Minecraft server hosting company, for which he learned some Java, Scala, Linux, PHP... whatever was necessary... Through creating mobile websites with PHP and MySQL he is currently programming mostly in React, Javascript and React Native. Since two years he is teaching programming to people all over the world, and enjoying it really well.

Dick Dral (58) has been working with Oracle since 1988. Through Oracle Forms and Designer he arrived at Oracle Application Express in 2006, and has stayed with that ever since. Since 2008 he is an independent contractor and almost all of his assignments he uses Apex. In the Apex Dashboard Competition he ranked second and in last years Geekathon he and Michiel scored the third place. He enjoys to blog and speak about Apex development. At his demo site http://speech2form.com/ords/f?p=opfg you can find a few plugins and tools that make your development work easier.

Problem
The average Smartphone user has 80-100 apps installed on his phone and uses about 30 monthly. This means many of the apps are not on the home screen. Dick has got about 250 apps installed on his phone.



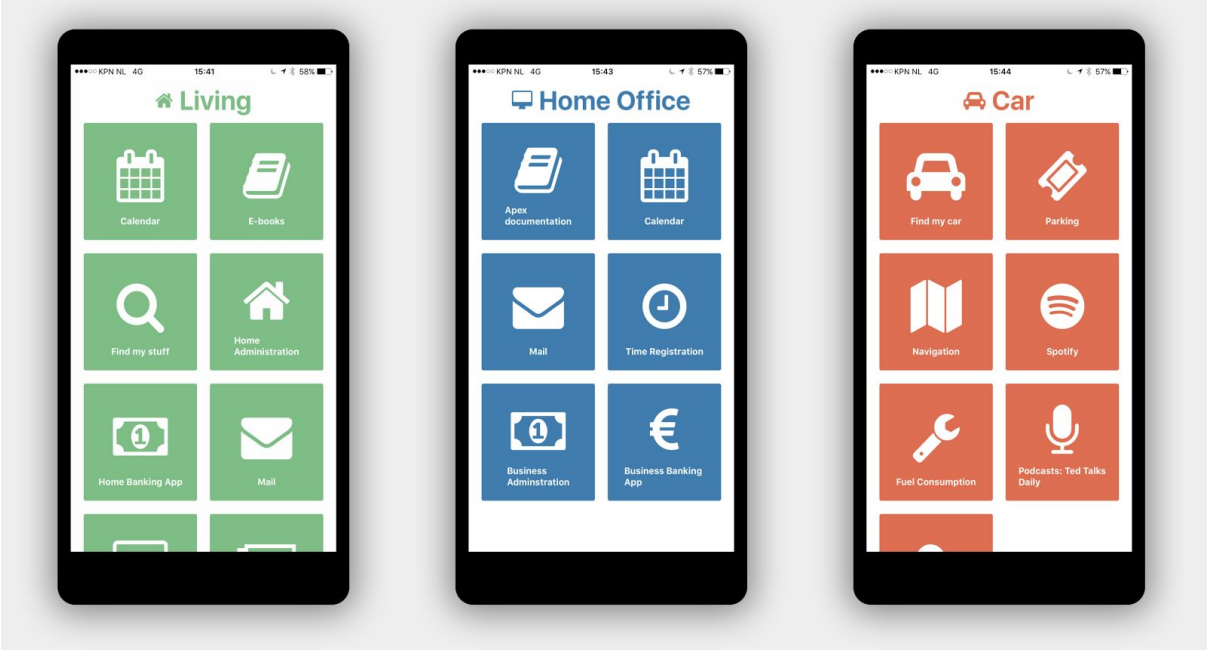
This means a lot of searching or swiping to find the right app. Yet the need to use apps is in many cases bound to a location. The parking app, navigation or fuel usage registration will

be used in the car, while mail, calendar and business administration will be of use in the office. In the living room I enjoy watching Netflix, listening to Spotify or reading an e-Book (or a combination ;-)).

Morpheus

Morpheus offers an app menu based on the user and the location. This way in every situation you have the apps you need at hand. Morpheus determines the location by detecting iBeacon signals. By placing iBeacons in relevant locations and registering those iBeacons to these locations, Morpheus is able to present relevant menu's in each situation (=location).

Below you some screenshots of the Morpheus menu's.



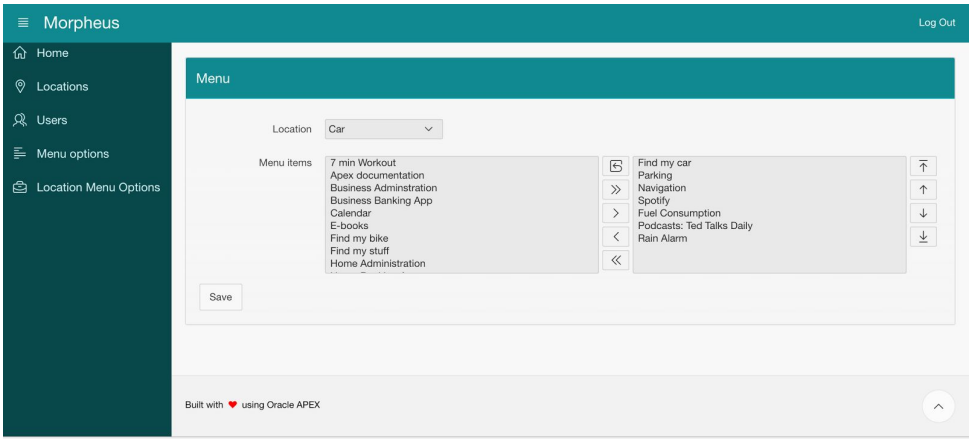
Because of the local caching of the menu's Morpheus also works off-line. On the other hand, because the menu's are stored in the cloud, you can also use Morpheus on your iPad. A version for Apple Watch is in consideration.

Architecture

The menu is displayed by a native app built using React Native. This app also receives the iBeacon signal. Based on the received signal it determines the location and thus the menu to be displayed.
The data is maintained using a back-end Oracle Apex application. Morpheus gets the menu information by calling a RESTful web service. The web service returns all the information for a user in one call. This enables the app to function while off-line because all the information needed is available on the device.

In the back-end application locations and menu options can be entered. Per location a menu can be constructed by choosing the appropriate menu options. The constructed menu's are specific for a user. So each user can have his or her own menu's.

For a location you can define a color and an icon. The color is very useful for easy distinguishing between the various menu's.
The definition of a menu options consist of a label and an icon.



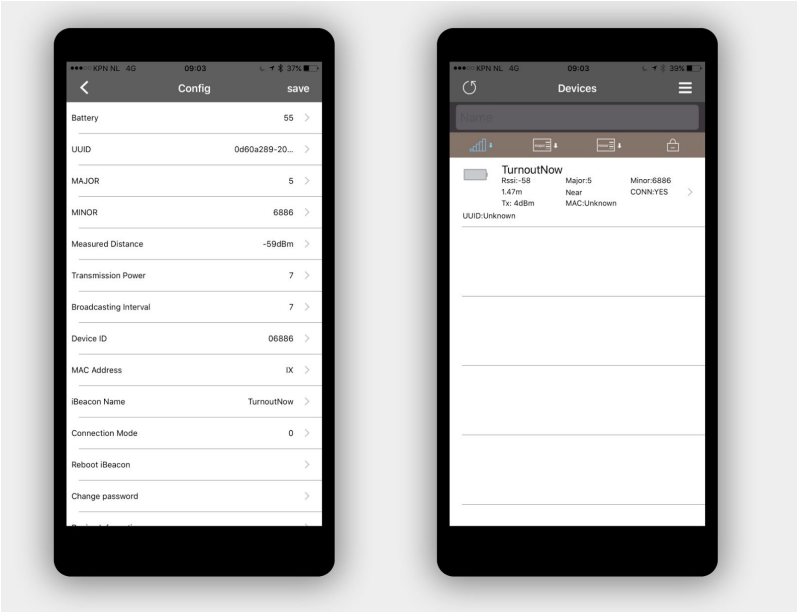
Use of iBeacons

Each iBeacons emits a specific signal. The receiving device gets the following information:

- UUID
- Major ID
- Minor ID

The UUID is the identifier for a group of iBeacons. In the application you have to define which UUID(s) should be listened to. So for Morpheus all iBeacons have the same UUID, which is also defined within the application.
Major and Minor ID can receive values that are useful within the application. If you have a room with several iBeacons, you can assign the Major ID to the room and use the Minor ID to recognize the individual iBeacons.
In Morpheus the Major ID is bound to a location. The value of Minor ID is not used.

The iBeacon used are made by Shenzhen Yunli-Wuli Network Co.,Ltd. They provide a configuration app **beaconSet,** to change the ID's:



Detecting the iBeacon

There are a number of solutions available to detect iBeacons. For our use case we investigated:

- detection from a web application
- detection from a web application wrapped in Cordova
- detection from a native app built with React Native

Detection from a web application
There is a JavaScript library for BlueTooth available on GitHub, called **web-bluetooth** (https://github.com/WebBluetoothCG/web-bluetooth). But we soon found out that the function to detect iBeacons, **watchAdvertisements**, is not implemented yet.
So the use of a web application is not feasible.

Detection from a web application wrapped in Cordova
This option seems more feasible. A Google search for **iBeacon** and **web application** points to this solution. So it seems possible, but the use of Cordova asks for a lot of setup. Therefore this direction was not investigated further.

Detection from a native app built with React Native
With React Native is easier to wrap native code, so Michiel has written a bridge between the Apple implementation for finding iBeacons and the react native code without too much effort.

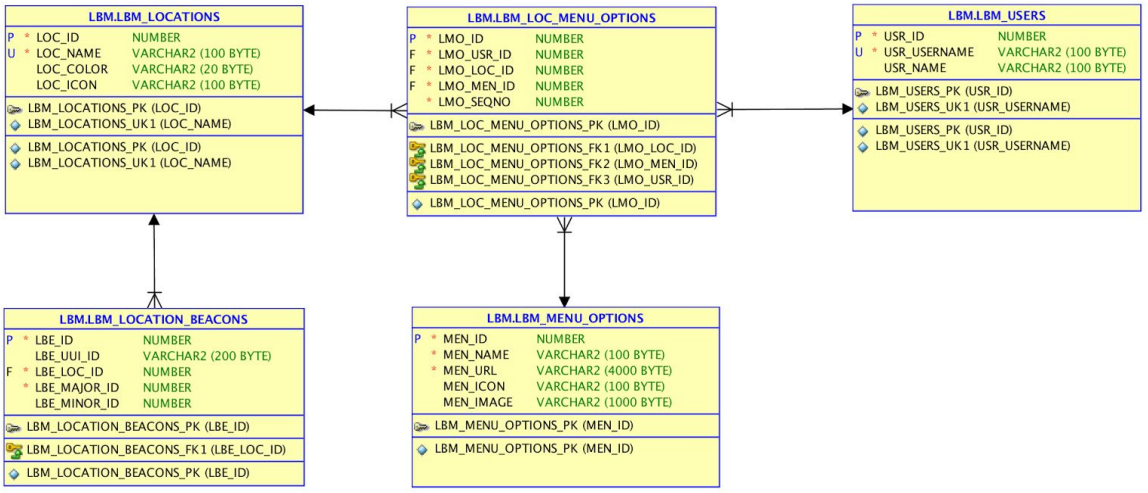
Building the Morpheus app

The interface is build in React Native, which mimics many of the webs properties. Styles are written in (a subset of) CSS, all logic is written in javascript. To fetch the information from the http service, the `fetch` api was used. Fetch, like a lot of API's native to the browser, are ported to run on React Native, allowing us to write modern javascript that works on every device. React, the library that gives React Native it's name, has a special way of composing

UI that a lot of developers like. It allowed me to seamlessly integrate my beacon listener into the program, to update whenever a new beacon was found.

Managing the data

Because of the available experience in the team Oracle Apex was chosen for managing the data. The application was created in Oracle Apex 5.1 using Universal Theme. Most of the pages use the Form with Reports design pattern with a simple Master Detail page for the Beacons assigned to a location.
Only the assigning of the menu options to the location for a specific user was designed to use a shuttle item. In a shuttle item the appropriate options can be chosen and also the order can be determined. So there is a bit of PL/SQL involved in storing the personalised location menu's.
The logged in user is taken as the user for which the menu's are defined.



Accessing the data

A web service is built using Oracle REST Data Services. This service returns a JSON list of all menu options with location and beacon major id for a given user.
This list is retrieved at the start of the app. If the server is not available - maybe the device is off-line - then the app uses a locally stored list.

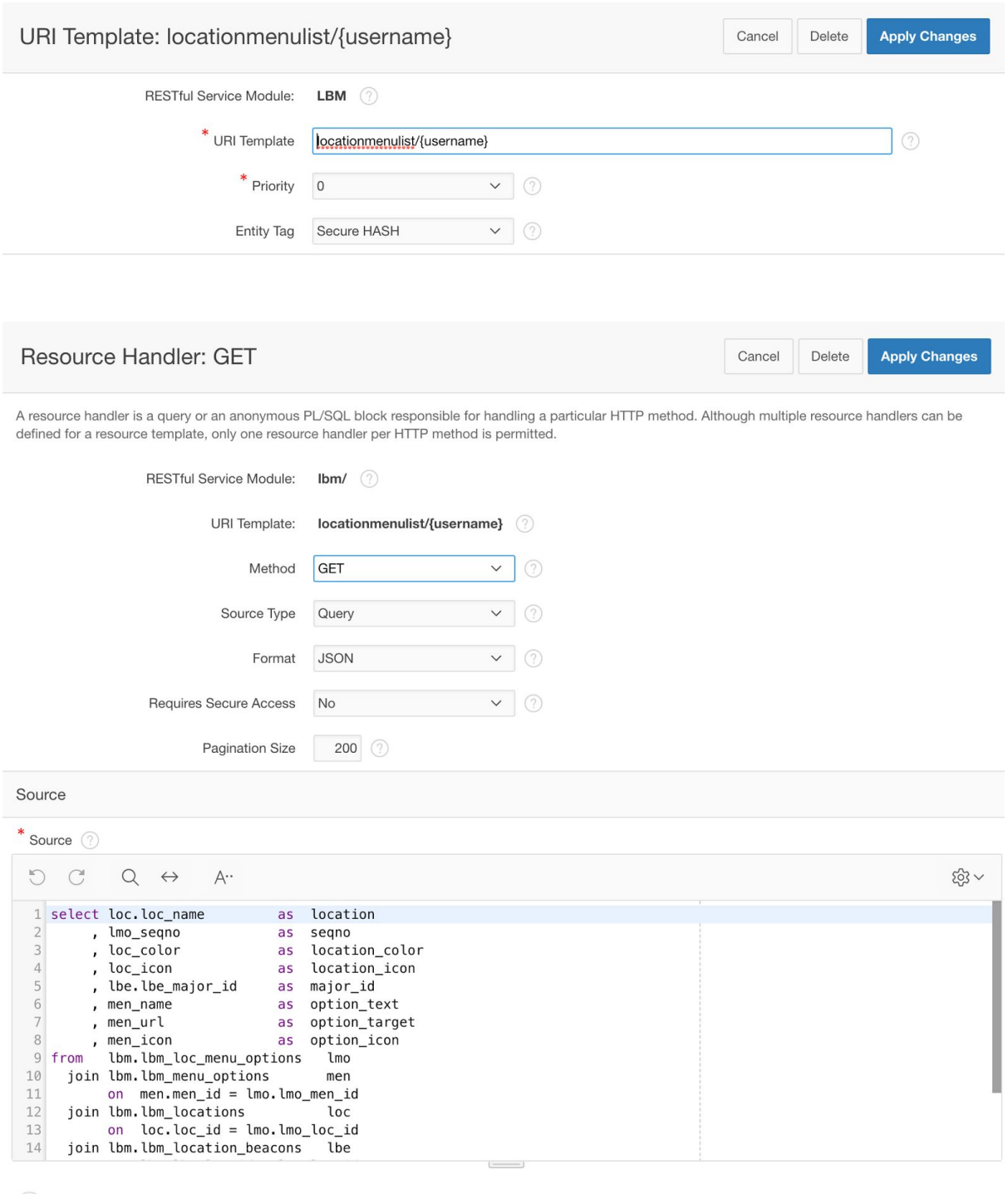
The following images show the definition of the Web Service in the SQL Workshop.

RESTful Module: LBM

Cancel | Delete | **Apply Changes**

A RESTful Service Module is a grouping of common templates under a common URI prefix. This prefix is prepended to all templates.

* Name	LBM
* URI Prefix	lbm/
Origins Allowed	
Required Privilege	- Assign Privilege -
* Pagination Size	200
Status	Published

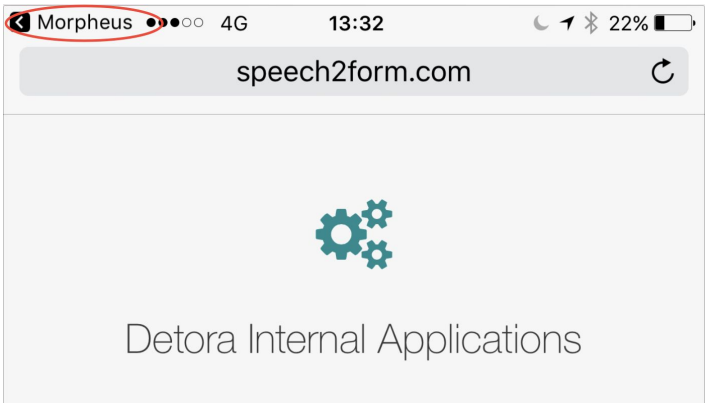


Calling web applications from the Morpheus app

When creating menu items you have to define the target, i.e. how the web application or app is called.

To define a menu option that points to a web application only an URL is needed. You can copy this URL from the Address line in your browser.

The Morpheus app uses a Safari Web View. This means that the web application is called within the app and that there is a button top left to return to the app.



Calling apps from the Morpheus app

The target for apps is an Uniform Resource Identifier (URI). These URI's have the form ***name*://**.
For example the URI to call the iOS calendar is **calshow://**, while Spotify can be activated with **spotify://**. These URI will direct you to the home screen of the apps.
If you want to jump to another point than the home screen you can add context information at the end of an URI. For example you can use **spotify://user/spotify/playlist/*{id}*** to start Spotify with a specific playlist active.
For Spotify the URI to a playlist or artist can be found in Spotify Desktop. Right click on a playlist, chose **Share** and click on **URI**.

The URI for apps are not always easy to find. For one app, Michiel had to download the IPA-package, unzip it and find the URI in the file **info.plist**. Also the format for passing parameters usually is not (publicly) documented.

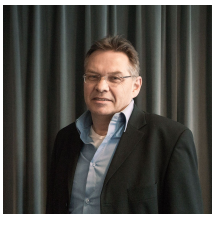
When calling an app for the first time you have to grant access to the app. The next time this question will not be asked any more.

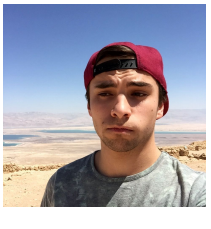
Btw. This way you can also call apps using an URI from a web application running on your phone!

Resources

Sources can be found at: **https://github.com/dickdral/Morpheus**
The server side application is read-only available at:
http://www.speech2form.com/ords/f?p=morpheus

Brought to you by team Father and Son





Dick Dral
Blog: dickdral.blogspot.nl
E-mail: dick.dral@detora.nl
Twitter: @dickdral
Github: dickdral
Linkedin: Dick Dral

Michiel Dral

E-mail: michiel@dral.eu
Twitter: @dralletje
Github: dralletje
Linkedin: Michiel Dral