

Simone Staffa

Andrea Misuraca

Simone Silvestri

Alessio Cesaretti

Mentors:

Daniele Paolacci

Camillo Ferlito

Google Technologies for Cloud and Web Development 2018

Index

[Introduction](#Introduction)

[Our idea](#Our_Idea)

[What is Parkidle](#What_is_parkidle)

[Who we are](#Who_we_are)

[Evolution from the first steps to the final submission](#final_submission)

[First Milestone](#first_milestone)

[Second Milestone](#second_milestone)

[Third Milestone](#third_milestone)

[Final Submission](#final_submission)

[Project](#project)

[Aimed Necessity](#aimed_nece)

[Our clients identification](#clients_identific)

[Our partners](#partners)

[Why Parkidle?](#why_parkidle)

[Manual Reporting apps](#manal_repo)

[Private Parking apps](#manal_repo)

[Business Model](#business_model)

[Use Cases](#use_cases)

[Case 1: student needs to park near its university](#case_one)

[Case 2: Saturday night](#case_two)

[Feedbacks](#Feedbacks)

[Technology](#Feedbacks)

[Structure](#Structure)

[Third parties services (API)](#API)

[App Overview](#App_overwiev)

[Server](#Server)

[How it works?](#how_works)

[Introduction to the detection world](#Introduction)

[Departed event detection](#departed_detect)

[Arrival event detection](#Arrival_detect)

[Events sharing](#Event_sharing)

[Rendering Events](#Rendering)

[Test](#Test)

[Conclusions](#Conclusi)

[Future](#Futu)

[Contacts and Download](#Contacts)

INTRODUCTION

Our Idea

In most of the big cities in the world, one of the most critical problems is to find a free parking spot. In that chaotic environment, it would be very helpful an assistant for find it easily.

This assistant can be Parkidle!

What is Parkidle:

Parkidle is a mobile application developed for Android OS that shows to the users empty parking spot around them.

The application shows to the user a map with the empty parking spot left by other users, both free and paid park. The app uses a Google technology called “Activity Recognition API” to understand what users are doing; our technology can collect the exact position where the user took its car and share it with everyone else as a simple marker on the map.

Parkidle was born inside the Google Technologies for Cloud and Web Development 2018.

Who we are

Our team is composed by four members:

Alessio Cesaretti

Backend and Frontend developer, studying Computer Engineering at Sapienza University of Rome.

 Andrea Misuraca

Backend and Frontend developer, studying Computer Engineering at Sapienza University of Rome.

 Simone Silvestri

Backend and Frontend developer, studying Computer Engineering at Sapienza University of Rome.

 Simone Staffa

Backend and Frontend developer, studying Computer Engineering at Sapienza University of Rome.

It was a great challenge to present and improve milestone after milestone our product, trying to present an application that solves real solutions in the smartest and technological way.

Like a real startup, we had problems to extends the number of users, the difficulties to test the product and to validate them. In the good atmosphere of the workshop, was born an application that is fully automated and really offer a different and an unseen solution to this problem.

Evolution from the first steps to the final submission:

The problem of parking in big cities is a theme that concerns us in the first person because we live in Rome and, to be fair, isn’t the most vehicle-optimized city in the world.

In this case we can say that our solution comes from everyday suffering and stress bound to find parking.

First Milestone

Already from the first milestone, we want to tell the story of the common driver and also build the main scenario for a clear application’s use case. Regarding the technology, we started initially with a third part service (Predict.IO) that uses the same Google’s services that will see in the most advanced phases of our project. Also, for us, was important the initial feedback that arrived from the first survey we made, surely was a boost for us to continue and improve the idea.

Second Milestone

In the second milestone, we collected the best feedback regarding features for the user to implement them by making the application closer to users. We started to evolve Parkidle about the automation of the detection, also continuing to monitor performance and making the application more stable.

Third Milestone

The idea is definitively formed from the technological point of view, we finish the background services and also formed a plan to have an increase in users. The project is definitely done, the experience was complete.

Final submission

We improve the service by focusing on the public release, arranging the repositories and the social pages of the application.

Project

Aimed Necessity

Finding parking spots is a crucial point in our life, especially in big cities like Rome where the average time to find a single spot is 10 minutes.

Parkidle takes action in this situation facilitating the research and showing people free parking spots on the map.

Our clients identification

Our project could serve a lot of clients, but we know that at the beginning it’s important to select a specific target, so we identified it.

The early adopters are the students of the universities in Rome (Sapienza, Luiss, Roma3) because they are very familiar with social networks and they share a restricted area in which parking is a struggle.

In the future we plan to enlarge the community to every driver in Rome.

Our Partners

From the very beginning of the project we tried to contact some companies that operate in the mobility field, but still we couldn’t get an engagement.

Why Parkidle?

Parkidle is not for sure the first parking application, maybe we are the thousandth, but we think that we can be the best. We are going now to analyze how Parkidle places itself between other services.

There are a lot of alternatives but there are a few points to prefer Parkidle:

● Automation: using our algorithm we can predict if the user is leaving a parking spot without him doing anything;

●Battery and

data saving feature: we use the lightweight MQTT technology that won’t finish your data;

●Precision: Other app just estimate the number of parking spot in a zone, but we can show you the spots accurately using the GPS.

Manual Reporting apps

Their weakness is the users’ laziness, because no one wants to open the app while driving and click the “report button”. When you use the application, you can see only 3-4 parking spots reported in the whole country.

Obviously, the automatic detection guarantees a better service.

Private Parking apps

These apps only show the position of private paid parking spots, so the number of the users is very little, since no one wants to pay for a parking spot.

Our app will have a larger community.

Business Model

Parkidle is a mobile app, made by four Rome based students, so we planned to implement Ads based on location, making agreements with small-medium business for sponsorship them and to gain the users privileges such as discounts, to cover those paid service, like AWS, web-APIs and licenses.

This is our mainly solution to give Parkidle as a free service to all user out there.

Use Cases:

Case 1: student needs to park near its university

One main use case, thinking about our early adopters, is clearly the needs for the students to don’t lose time.The student itself can check the application for some free parking spots near there, reducing the possibilities to be late to the lesson.

Case 2: Saturday night

Going out, on Saturday night, with a car in the central places in Rome means deal with the climax point of this whole problem.

With Parkidle, you have a support to choose efficiently which roads to take, seeing in real time which parking are going to being freed.

We want to say hi to Michele for that time he can’t achieve to reach us after going around for 1 hour and half, searching for some places to leave the car.

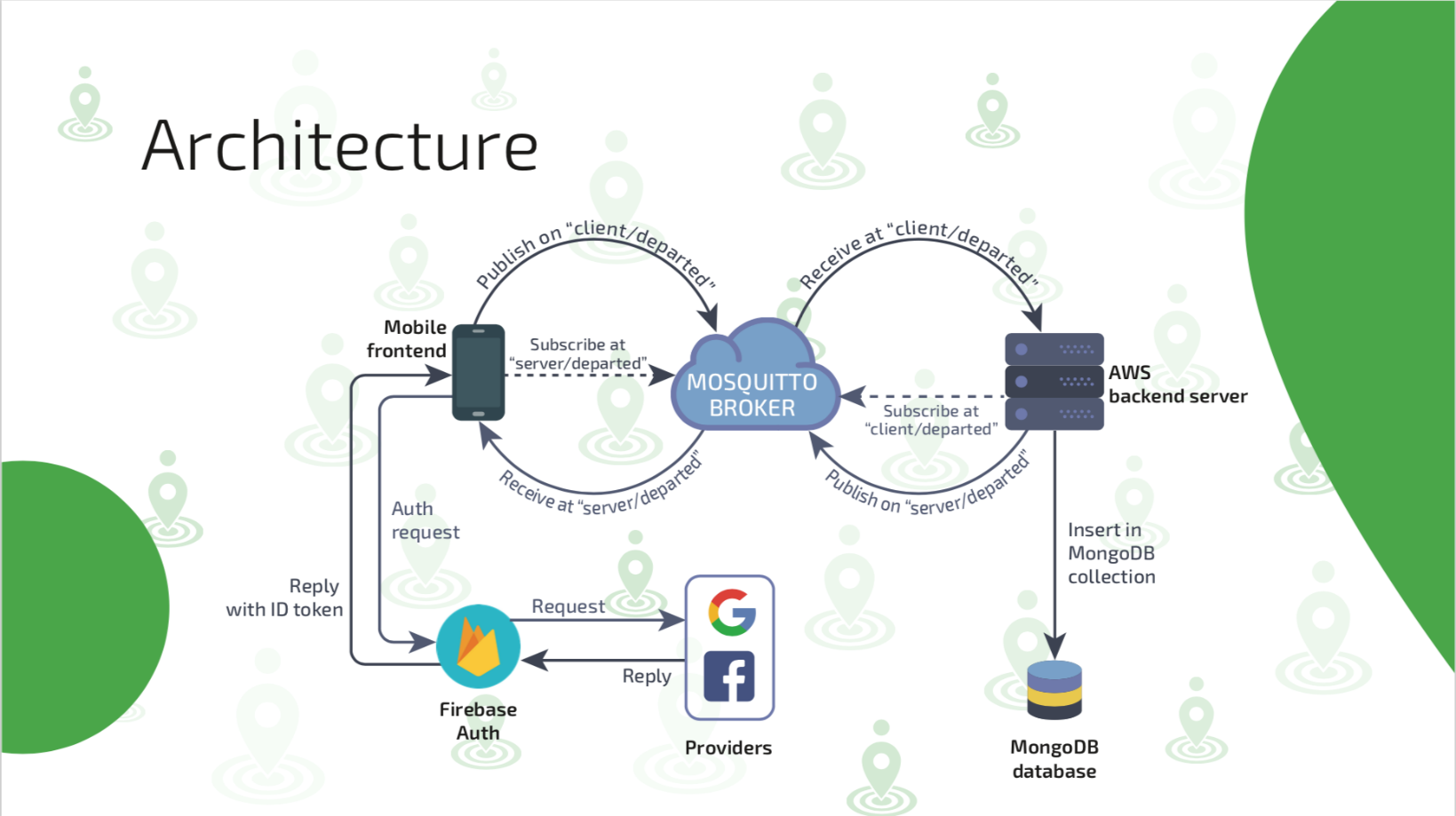
Feedbacks

We did several survey about idea validation and how this should be implemented in the application, starting from what functionalities they expect to find there.

[inseerire immagini survey??]

User feedbacks lead us in the right path and, at the moment, our closed alpha testers group (around 100) is searching for us bugs and indirect problems not due to the application.

Technology



General Communication Architecture Schema

Parkidle code is public on our GitHub repository: <https://github.com/ParkIdle/ParkIdle-Android>

Structure

•Mobile-app (Android) - Front end for the user

•Server - Back end for us to handle the communication between the users with the MQTT Broker and to store the events in our MongoDB database

•MongoDB database to save all the messages about the events shared by the users, using them soon to provide a kind of “event previsioning”

•Mosquitto Broker hosted on our server to handle the MQTT Protocol communication between users and server

•Firebase Authentication to let user authenticate with Google or Facebook

Third parties services (API)

•Mapbox: free maps service for a maximum of 50.000 requests/month. It includes also a navigation API

•Google Activity Recognition Client: accurately detect users' activity using low power signals from multiple sensors in the device

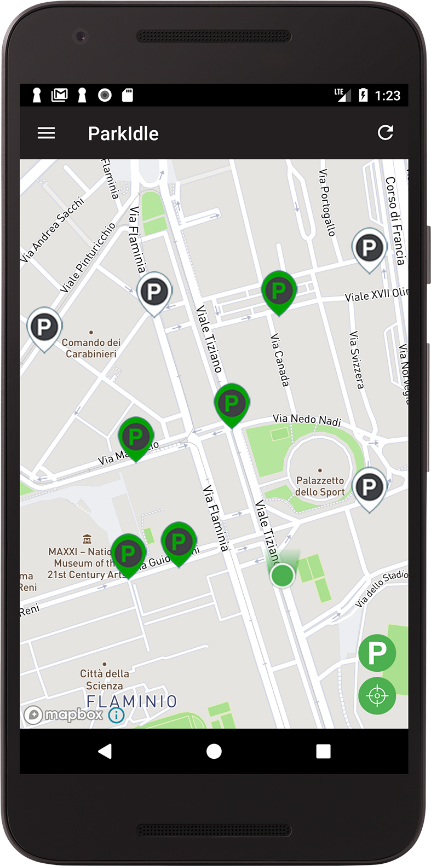
•Bugfender: remote logger to get fast remote access to applications’ log files on users’ devices

•Crashlytics: lightweight crash reporting solution, used as a Firebase extension

App Overview



Login with Google or Facebook



The main screen consists in a real-time Map with the empty parking spot markers.

4 colors based on marker’s life-time:

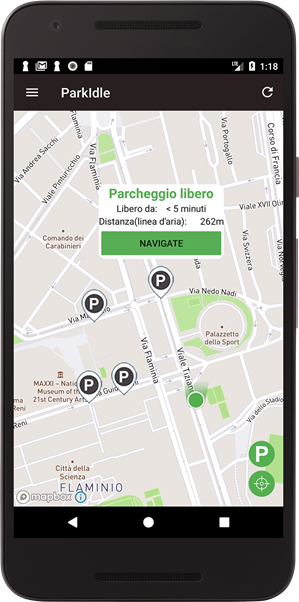
* White (0-5 mins)
* Green (5-10 mins)
* Yellow (10-20 mins)
* Red (more than 20 mins)



shows the nearest parking spot

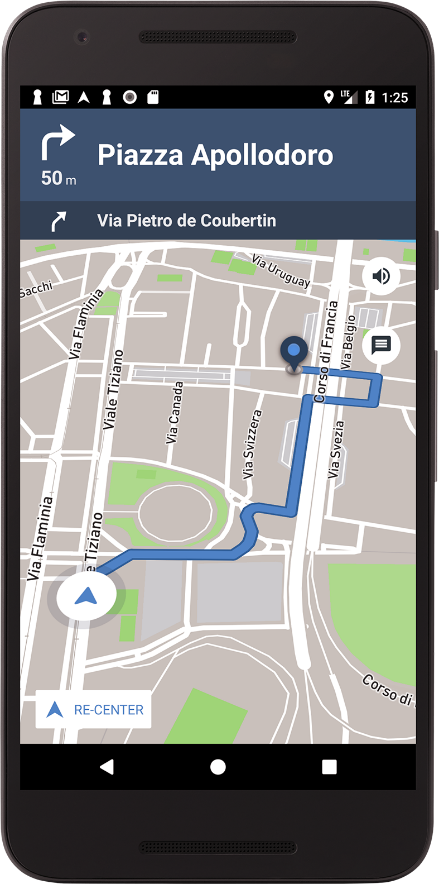


centers the camera on user’s location

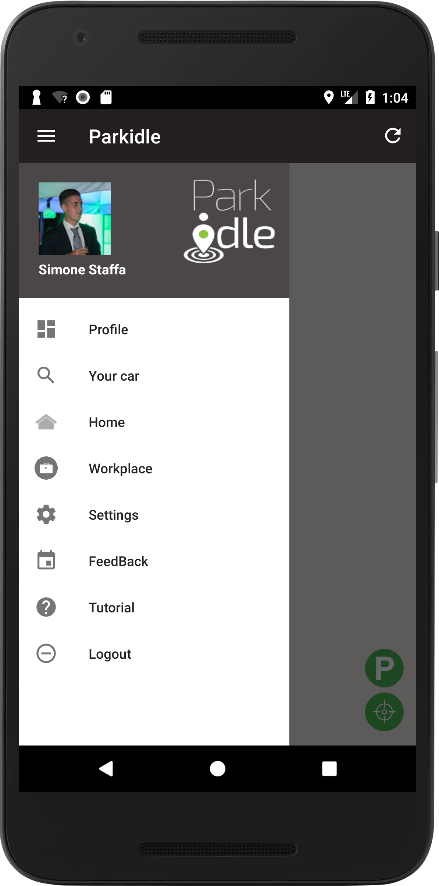


Clicking on a marker will pop up an info window

* Marker’s life time
* Airline distance
* Navigate button

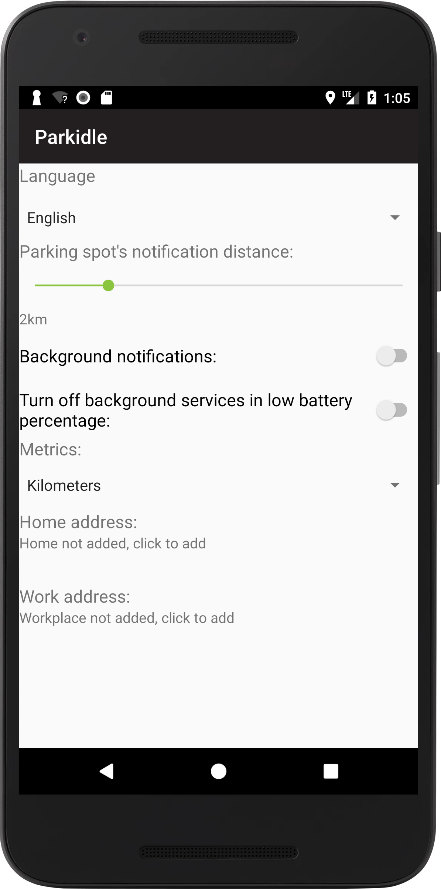


**Navigation** to help reaching the selected parking spot



**Side-left menu** with utilities:

* **Profile**: not yet implemented, will contain number of shared parking spots, the more you share, the greater will be your level (1 to 5 stars)
* **Your car**: shows where is your car parked
* **Home**: shows your home position (set in the “Settings”)
* **Workplace**: shows your workplace position (set in the “Settings”)
* **Feedback**: lets the user to send us a message (bug reporting, feedbacks etc.)
* **Tutorial**: starts a tutorial to help the user know how to use Parkidle and its core functionalities
* **Logout**: logout the user and get back to the initial login screen



**Settings screen** to customize the user experience

* **Language**: italian or English
* Parking spot’s **notification distance**: to choose the range in which notification should be received
* **Background Notifications**: set on/off notification receiver when the process is not even running
* **Turn off** background services **on low battery** percentage: set on/off activity recognition to preserve battery consumption when the percentage is low
* **Metrics**: switch from kilometers to miles
* **Home address**: set your home address to use it in the side-left menu
* **Work address**: set your workplace address to use it in the side-left menu

Server

Currently the system is hosted on Amazon Web Services platform on an Ubuntu 16.04 LTS operating system.

Other server-side technologies:

• **MongoDB**: non-relational database service to store data in json format

• **Nodejs**: javascript open-source server framework + a lot of “node-modules”.

How it works?

The core functionality of the application, the unique feature against competitors, is the automatic parking spot detection which is made with **Google Activity Recognition Client** (GARC).

Thanks to GARC, we can detect the **user’s activities** (STILL, WALKING, RUNNING, VEHICLE) every “x” seconds (we decided to use a 13sec detection interval).

Each detection is stored in a list of 10 elements. Every detection, we scan the list to check if it matches with a “**pattern**”. A match corresponds to a **valid event**.

In Parkidle we can distinguish **2** types of events: **departed** and **arrival**.

While a departed event is shared with all the users to let them know where the empty parking spots are, an arrival event is not shared, it only helps a user to find the position where he parked his car.

Introduction to the detection world

The detected activity’s list (made of 10 elements) is divided in 2 minor lists: the first five activities (oldest ones), and the last five

activities (recent ones). Firstly, we check if in the recent ones happened **something interesting**, then we confirm it checking

the oldest ones to **avoid false positives** (it may happen that in certain traffic condition we get some error).

Departed event detection

A departed event consists in comparing the recent activities with these patterns:

1. !VEHICLE VEHICLE VEHICLE VEHICLE VEHICLE
2. !VEHICLE !VEHICLE VEHICLE VEHICLE VEHICLE
3. !VEHICLE !VEHICLE !VEHICLE VEHICLE VEHICLE

!VEHICLE stays for any activity except VEHICLE

If we have a **match** between the recent activities (the last five from the list) and one of these patterns, we can check the oldest activities (the first five from the list).

About the oldest one the check will consist in **finding** if there’s a VEHICLE activity to avoid **false positives**. If there’s a VEHICLE, the event is **not launched**, if there isn’t a VEHICLE the event is sent to the server and broadcasted to all the users who will see an empty parking spot in that position.

About the **position**, to have an accurate one we always get it from the first interesting detection, in this case we use the position of the first VEHICLE activity in the recent ones.

Now **focus on false positives**. A false positive in our real use-cases is mainly caused by **certain traffic condition**, in which a user accelerate and decelerate continuously, causing a detection list to one of the 3 patterns above (we may have a !VEHICLE sequence, followed by a VEHICLE sequence). To avoid this, we decided to enlarge our list to 10 elements, so we can know if a user in the **last 2 minutes was in a traffic queue**, **discarding** a lot of **false event** detection.

Arrival Event Detection

An arrival event consists in comparing the recent activities with these patterns:

1. VEHICLE !VEHICLE !VEHICLE !VEHICLE !VEHICLE
2. VEHICLE VEHICLE !VEHICLE !VEHICLE !VEHICLE

!VEHICLE stays for any activity except VEHICLE

As in the departed event case, firstly we **search** for a **match** in the **first ones**, then we check the oldest ones. In the first we have to match these 2 patterns, then in the oldest one (to avoid false positives) we check if we haven’t any VEHICLE activity. If we have it, the event is not launched, if we haven’t it, the position of the first !VEHICLE in the recent ones is **saved**, and it will be used for the “My Car” function in the side-left menu. That position will be removed if a departed event is detected.

Event sharing

The communication between users and server is handled with **MQTT Protocol**, because “is an extremely lightweight publish/subscribe messaging transport, ideal for mobile applications because of its **small size**, **low power usage**, minimised data packets, and efficient distribution of information to one or many receivers”.

MQTT messages passes through a broker who dispatch them to specific queues (topics). Our **Mosquitto Broker** (hosted on AWS) uses 2 topics: **server/departed** and **client/departed**.

Every user subscribes to the server/departed topic, the server subscribes to the client/departed one.

Our MQTT messages have all this format:

“ID-TYPE-DATE-LATITUDE-LONGITUDE”.

* **ID** - client-id, who sent the message
* **TYPE** - departed or arrival (currently we only use the departed one)
* **DATE** - date of the detection (with both day and timing)
* **LATITUDE** - latitude of the detection
* **LONGITUDE** - longitude of the detection

When on user-side is detected a departed event, the client publishes on **client/departed topic**, the server receives on that topic and **publishes the same message** on server/departed (with QoS = 1) to assure that all the users will receive that message, even if they’re not yet connected. The publish of the server is like a **broadcasting** of the message.

Rendering Events

Once a message is received a “P” marker (white) will be **added to the map**, and according to user preferences a notification will be shown (notification range). Clicking on it will open the app, taking the camera to that parking spot.

Test

Testing the reliability of the detection algorithm we made to detect when somebody is leaving/occupying a parking spot is crucial to have **credibility**. We struggled a lot to find the best way to test our app.

Finally, we found the proper method to make it easier: we **collected real trip data** from our users and used them as **test-cases for our detection algorithm**.

In a 100 test-cases, **75%** of them **passed** our algorithm in the right way, it means that for every trip we were able to detect 1 departure and 1 arrival. The other 25% were a lot different from each other, for example we had some cases in which the 1st departure wasn’t detected, but we also had others in which we detected more and more departures, what we call “false positive”, this may happen when the user is in a long traffic queue or stopped at a particularly long semaphore. This means that the algorithm still be improved and we’re working on it.

The fact that the 1st user departure isn’t detected depends on the time the user opens the app before leaving, because sometimes the silent session of Parkidle is not working properly, so it starts detecting activities too late to get all the events needed to detect a departure. Fortunately, this problem doesn’t affect the last arrival detection which has always been detected.

About the false positive we think that them can be avoided enlarging the “interesting activities interval”, it means we probably must keep track of more activities, reducing the detection interval. Right now, a new activity is detected every 14 seconds, so if we hold a list of 10 elements, we have a 140 second “interesting activities interval” (2 minutes and 20 seconds).

Conclusions

Future

Parkidle project will continue even after the workshop experience. Since we participated in a startup incubation program, our team will continue working on finding the best solution to solve the parking problem, finding ways to build up a startup, providing a stable business model and planning to develop an iOS version.

About the workshop experience and the progress presented during the milestones, we have to mention our mentors from NTTData: Daniele Paolacci and Camillo Ferlito. They really helped us during these months, constantly giving tips and suggestion about the developing process and the content presentation, since we’ve never experienced something like this. They also gave us the possibility to present Parkidle in their company.

Contacts & Download

For any enquires:

[info@parkidle.com](mailto:info@parkidle.com)

download link (betatesting):

<https://play.google.com/apps/testing/pi.parkidle>

