A Programming Framework for People as a Service *

David Bandera[0000−0002−548−7940], Alejandro Pérez-Vereda[0000−0002−0195−0062], Carlos Canal[0000−0002−8002−0372], and Ernesto Pimentel[0000−0002−7125−8434]

Departament of Computer Science, University of Malaga (Spain)
http://scenic.uma.es/

Abstract. The number of devices connected to the internet is constantly growing, which implies an increased complexity when interacting with so many heterogeneous devices. Automating this process is key to keep up with this growth. This People as a Service model works towards developing virtual profiles for every user in their own mobile devices and under their full control. These profiles allow to establish user preferences and predefined parameters, which are then applied by the devices they connect to. By integrating both the information in the virtual profiles and these devices, we can create a context in which to make smart decisions and apply them automatically, all of this in a decentralised way. In order to show our proposal in action, we have developed a treasure hunting game as a proof of concept to bring to the spotlight the utility of an environment with programmatically adapted devices.

Keywords: People as a Service · PeaaS · Beacons · Virtual Profile · IoT

1 Introduction

The increase in the capabilities of smart devices has brought a growth in the amount of embedded systems and devices we can find everywhere. However, these devices are highly heterogeneous, which causes an increase in difficulty and complexity of intercommunication between them, and an increase in security threats [1]. To help alleviate this issue, we need to work towards automating the task of configuring multiple devices and interacting with them in an easy and personalised way for each user. For this purpose we have adopted the People as a Service (PeaaS) model [2]. The idea behind PeaaS is to give the users a way to offer as a service a virtual profile with personal-related information. The virtual profile can be accessed by the devices the user interacts with, but at the same time giving the user full control over their data. Not only that, smart devices must be able to adapt to the user's situation. This would be achieved by modifying the virtual profiles in a programmatic way.

* This work has been funded by the Spanish Government under grant PGC2018-094905-B-100.

As a proof of concept, in this work we have developed a treasure hunting game. Treasures will be represented by Eddystone beacons. These simple devices can be easily deployed, require very little maintenance and can be programmatically adjusted. The same approach can then be extrapolated to other areas of application, such as smart cities. For example, for informing the users of how long the bus will take to arrive, to detect how many people are waiting at the bus stop, and from that inferring if an extra bus would be needed.

This paper is structured as follows. Section 2 explains how the system was implemented and the technologies involved. Section 3 presents the treasure hunt game as a proof of concept of the proposal. Finally, Section 4 summarises the conclusions and possible utilities of this work, along with future work. A more detailed description of the proposal can be found in [3].

2 Overview of the proposal

Our goal is to develop a framework that implements the PeaaS model. PeaaS implies a shift from a server-centric structure to a distributed environment, where the smartphones are the focus of the system and becomes an interface through which the virtual profile is accessed, via an specific API. The framework allows to develop generic mobile and server applications that download and run the scripts provided by the devices and interact with the profile. This allows dynamically updating the user's virtual profiles and modifying the behaviour of the devices, building this way a context of the situation the user is currently in.

In this scenario, the functionality of the system can be updated by modifying the script, without the need to deploy new applications on the server and mobile layers, or to change the settings of the deployed IoT devices. Scripts can be also modified by user's interaction. Depending on their virtual profile and context, some variables of the script can be updated to change how the device behaves. This way, devices automatically adapt to suit the users' needs in a seamless way.

In this work we present a treasure hunt game to show a working example of our proposal, and highlight the advantages and disadvantages we found.

3 Motivating scenario: a treasure hunt

The framework is composed of three elements, the mobile application, the server, and the beacons (Fig. 1). In the treasure hunt, the players look for treasures hidden around the city by following a set of hints, and each treasure found gives a new hint to figure out the location of the next one. The treasures are represented by Eddystone beacons, small devices that broadcast Bluetooth packets, allowing nearby devices to connect to them. For finding the treasures, players employ a mobile application which acts both as the platform where the game is played, detecting and interacting with the beacons, and as an entrance point to their virtual profile for the elements involved in the game, via an API.

Beanshell (http://www.beanshell.org) is a simple Java interpreter capable of uploading and executing code during runtime. It allows us to write simple scripts

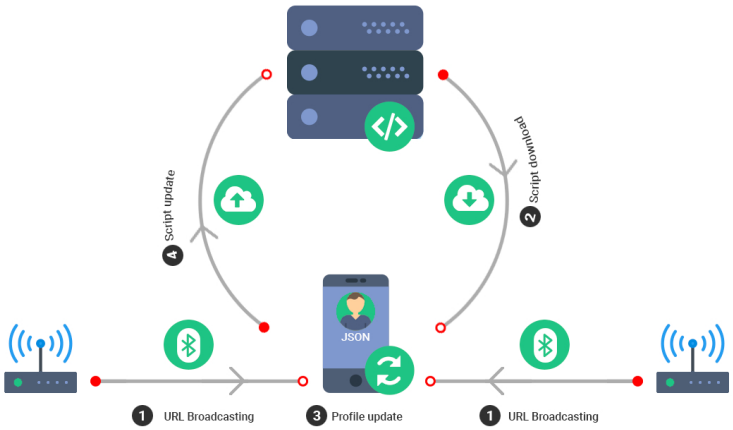

Fig. 1. Dynamic programming framework

for querying and updating the virtual profile, as well as to display notifications and messages.

The server is a Node.js server written in JavaScript, using Express as the framework for the server API. It hosts the scripts and the information needed to keep all the players synchronised, and it is accessed by the mobile devices. The information about the treasures and the available hints is kept in a MongoDB database. We keep track of which player has visited each treasure in order to inform others users of how many players have already visited the treasure they just found.

Each beacon holds the shortened URL linking to the location of a script on the server. When a player accesses it, the server sends the Beanshell script to the mobile device. The script is interpreted and run locally on the player's device.

We have developed one single script that works the same for all the treasures, but it could be possible to have specific scripts for different types of beacons, so they behave in a different way. The variables of the script are set up each time it is downloaded with the relevant values, based on the current user and context.

The mobile application has been developed for Android. It's purpose is to hold the player's virtual profile, to communicate with the beacons and the server as well as provide access to the virtual profile to other devices, and to execute the Beanshell scripts holding the logic of the game.

The application downloads and execute the scripts in a controlled fashion, accessing the user profile through the API, updating the information contained to add the treasure and the new hint just found. Once one player finds the last treasure, she will be informed of being the winner, and the rest of the players

will receive a notification of the game ending and the name of the winner the next time they find a treasure.

The functionality of the script is as follows. First, it will check if the treasure was already found by the user to avoid giving more than one hint per treasure. This is done by looking for treasure ID in the player's virtual profile, in which case it will inform the user that they already obtained this particular treasure. If the treasure is a new one, the script connects to the server and informs that the player has found this treasure, in order to update the database. The server will then send a response with the current state of the game. If the game had already finished, the player will receive a notification informing them about it and the name of the winner. Otherwise, the script will try to give the user a new hint. If the user already has all the hints, a message will appear congratulating them for having won the game, while at the same time connecting to the server to declare them as the winner to set the game as finished.

In the case where the player still hasn't finished the game, the script displays the new hint.

4 Conclusions

The ability to infer virtual profiles of people according to their daily routines and activities is a key element to create a world where technology adapts to the people in a seamlessly way [4]. In this work we have presented a working example of a system able to adapt to the user based on their virtual profile in an automated way. Using Bluetooth devices give us a high degree of flexibility when developing these systems. There are many types of devices with different characteristics available on the market, allowing for a more complex behaviour. By replacing the beacons we used in our system by devices with more capabilities, we can extend the functionality of the system, taking advantage of the extra processing and qualities they offer. This would also allow to have a more interactive system where the different devices share information to work towards a common goal, improving the quality of the solutions obtained.

References

1. Covington, M.J., Carskadden, R.: Threat implications of the internet of things. In: 2013 5th International Conference on Cyber Conflict (CYCON 2013). pp. 1–12. IEEE (2013)
2. Guillen, J., Miranda, J., Berrocal, J., Garcia-Alonso, J., Murillo, J.M., Canal, C.: People as a Service: a mobile-centric model for providing collective sociological profiles. IEEE software **31**(2), 48–53 (2014)
3. Pérez-Vereda, A., Flores-Martín, D., Canal, C., Murillo, J.M.: Towards dynamically programmable devices using beacons. In: Current Trends in Web Engineering, LNCS, vol. 11153, pp. 49–58. Springer (2018)
4. Sadeeq, M.A., Zeebaree, S.R., Qashi, R., Ahmed, S.H., Jacksi, K.: Internet of things security: A survey. In: 2018 International Conference on Advanced Science and Engineering (ICOASE). pp. 162–166. IEEE (2018)

Appendix. Technical requirements

Our technical requirements for the demo involve:

- Several Eddystone beacons for representing the treasures (they will be provided by the authors).
- Mobile phones running Android OS 5.1 or higher for the users that want to join the treasure searh, with our Beacon scanning app installed.
- A remote server running Node.js and Express. Again it will be provided by the users.
- A projection screen to display the video of the demo and also the screen of mobile phones playing the game.

In order to give the demonstration, the beacons need to be deployed in suitable places. Each beacon would be already configured for broadcasting the URL of the corresponding script. Any user that want to participate in the treasure hunt would just need an Android mobile phone with beacon scanning application installed. The application APK can be downloaded from our repository, its URL available below. The remote server hosting the scripts would be already deployed and working, with the server application installed in it. Finally, during the demo we will cast our phone screen to the display via AirDroid, so everyone can watch how the application works.

The code of the Android app and the SDK can be found at `https://bitbucket.org/apvereda/beacons/src/master/`. The code of the server application is also available at `https://bitbucket.org/Traffen/treasuregame/src/master/`

A demonstration video can be found at the following URL:
`https://www.youtube.com/watch?v=uTxLp4pke-U`