IoTRec: The IoT Recommender for Smart Parking System

Yasir Saleem[a], Pablo Sotres[b], Samuel Fricker[c], Carmen López de la Torre[b], Noel Crespi[a], Gyu Myoung Lee[d], Roberto Minerva[a] and Luis Sánchez[b]
[a]Institut Mines-Telecom, Telecom SudParis, CNRS UMR 5157, France
[b]Network Planning and Mobile Communications Laboratory, University of Cantabria, Santander, Spain
[c]Institute for Interactive Technologies, Fachhochschule Nordwestschweiz, Switzerland
[d]Department of Computer Science, Liverpool John Moores University, Liverpool, UK

***Abstract*—This paper proposes a General Data Protection Regulation (GDPR)-compliant Internet of Things (IoT) Recommender (IoTRec) system, developed in the framework of H2020 EU-KR WISE-IoT (Worldwide Interoperability for Semantic IoT) project, which provides the recommendations of parking spots and routes while protecting users' privacy. It provides recommendations by exploiting the IoT technology (parking and traffic sensors). The IoTRec provides four-fold functions. Firstly, it helps the user to find a free parking spot based on different metrics (such as the nearest or nearest trusted parking spot). Secondly, it recommends a route (the least crowded or the shortest route) leading to the recommended parking spot from the user's current location. Thirdly, it provides the real-time provision of expected availability of parking areas (comprised of parking spots organized into groups) in a user-friendly manner. Finally, it provides a GDPR-compliant implementation for operating in a privacy-aware environment. The IoTRec is integrated into the smart parking use case of the WISE-IoT project and is evaluated by the citizens of Santander, Spain through a prototype, but it can be applied to any IoT-enabled locality. The evaluation results show the citizen's satisfaction with the quality, functionalities, ease of use and reliability of the recommendations/services offered by the IoTRec.**

***Index Terms*—GDPR, Internet of Things (IoT), parking statistics, recommendations, smart parking.**

I. Introduction

During the past decade, there is a significant number of cars circulating in the cities which makes the parking and the traffic, two serious issues. Customarily, drivers try to find available parking spots on the streets by driving around, only locating a parking spot empirically due to their local knowledge and luck. This practice wastes a significant amount of both time and fuel, and sometimes it is impossible to find a free parking spot during high vehicle traffic times. One solution would be to find a parking area with high capacity of free parking spots, increasing the chances of getting a parking spot. However, this parking area could be very far from the user's destination. Another solution is to design a system that shows free parking spots to the driver and lets the driver chooses a free parking spot manually. However, this is not an optimal solution because firstly, it is an extra task for the drivers to select a parking spot by themselves. Secondly, the path leading to the selected parking spot could be very congested, causing the parking spot to be occupied when the driver arrives.

IoT technology has revolutionized almost all fields of daily life, including parking systems, by exploiting the immense developments in technology. Inspired by these new possibilities, a smart parking system has been designed to automate the recommendation of free parking spots to the drivers looking for them, thereby minimizing the time spent on finding free parking spots, as well as minimizing the cost associated with hiring humans for manual parking management [1], [2], [3]. Such a solution is based on a parking spot reservation system that utilizes various wireless networking technologies, such as ZigBee, Radio Frequency Identification (RFID) and the Internet. The system provides information about nearby free parking spots and allows drivers to reserve the parking spots through their devices by using the ID that univocally identifies each vehicle in a parking spots reservation system [1]. However, it is not always possible to reserve parking spots in advance because of the regulations in some cities (e.g, Santander, Spain). Therefore, there is still a need for a system that can recommend the best possible parking spots based on certain metrics.

In additional, in the recommendation of parking spots, it is very important to consider the traffic on the route to each available parking spot and to recommend the least congested route leading to the recommended parking spot. To better understand the importance of traffic congestion, let us consider a rush hour scenario when the traffic in the city center is at its peak. In this scenario, the streets will be very congested, and many people will be looking for parking spots. When a driver finds an available parking spot and heads towards it, there is a high likelihood that when the driver reaches the parking spot, it will already be taken because multiple drivers are looking for similar parking spots, and the traffic congestion caused a delay in reaching the parking spot. We solve this problem by selecting a parking spot that is the nearest to the user and by providing the expected availability (occupancy statistics) of parking areas to the users.

Many of the IoT applications available today have been developed in a vertical manner by focusing on a specific scenario or use case without considering data exchange and reuse with other IoT applications. This very specific focus results in poor service because of the lack of integration of different data and hence the interoperability in the IoT data and systems. However, if IoT applications could collaborate by exchanging and reusing each other's data, opportunities for new value-added and more efficient services could be generated. The semantic web is a promising technology with which to achieve the needed interoperability [4]. Hence, newer IoT applications, including smart parking system should be semantics-enabled adopting semantic data modeling and semantic web technologies for supporting the interoperability in IoT.

With the enforcement of the EU General Data Protection Regulation (GDPR), protecting the privacy of EU citizens throughout the data collection, data storage and data processing of a user's personal data is now a basic requirement [5], [6]. Parking systems gather a lot of contextual data and it is quite possible that the users' personal data can be collected indirectly. GDPR affects also smart parking applications and hence, the smart parking systems should therefore be designed in a way that protects user's privacy and thus be GDPR-compliant.

A smart city infrastructure was deployed in Santander, Spain in 2010−2013 as part of an EU project, SmartSantander [7], [8], [9]. Quite a number of sensors were deployed, including traffic, parking, bus stop, bus line, irrigation, environmental and mobile sensors [10]. Fig. 1 presents the deployment of traffic and parking sensors in the city of Santander.

In this paper, we propose the IoT Recommender (IoTRec), a smart parking system that is GDPR-compliant and considers the road traffic conditions. We utilized the semantic IoT data of the deployed parking and traffic sensors as presented in Fig. 1 for the recommendation of available parking spots and the best routes based on some metrics. The envisaged metrics for parking spots are the nearest, or nearest trusted parking spot, while the metrics for a route are the least congested or the shortest route. A trusted parking spot is defined as a parking spot that is trusted by the user based on his past experience, and that is also trusted based on the quality assessment of the parking sensor. For the nearest trusted parking spot, a Trust Monitoring component developed in our recent work [11] calculates the trust scores of parking spots by analyzing the user's experience through a feedback mechanism [12] and sensor quality assessment. Such trust scores are utilized by the IoTRec in the calculation of the nearest trusted parking spot. The Trust Monitoring component is not the main scope of this paper; the readers are referred to our recent work [11] for details. The IoT recommender is novel and different from other recommender systems in four primary aspects. Firstly, the IoT recommender mainly considers the IoT data of parking and traffic sensors, rather than the data shared or acquired by users' terminals. Secondly, the IoT recommender provides trusted recommendations by integrating with Trust Monitoring component of WISE-IoT project. Thirdly, it provides GDPR-compliant, as well as normal implementations that work in a privacy-aware and non-privacy-aware environment. Fourthly, it offers the expected availability (occupancy statistics) of parking areas to users and enable them to analyze the weekly, monthly and yearly statistics by themselves. To the best of our knowledge, these features do not exist in current recommendation systems.

In this paper, we first present the semantic data modeling of parking and traffic sensors data utilized by the IoTRec in the recommendations of parking spots and routes and we then present an overview of our proposed IoTRec. To allow the drivers to analyze the expected availability (occupancy statistics) of parking areas (parking spots organized into groups) in a user-friendly interface and to select a parking area, we develop and present the statistics of Santander's parking areas. Our proposed IoTRec is integrated into the real world in Santander in a prototype of the H2020 EU-KR WISE-IoT project [13], [14] called the Rich Parking application, making use of REST APIs. REST APIs make the integration with the smart parking application easy and simple, and also ease the interoperability and reusability by allowing other IoT applications to integrate these APIs and offer new efficient services. It is important to note that although the IoTRec is evaluated by the citizens in Santander, it can be applied to other cities that have similar infrastructure.

The contributions of this paper are summarized as follows:

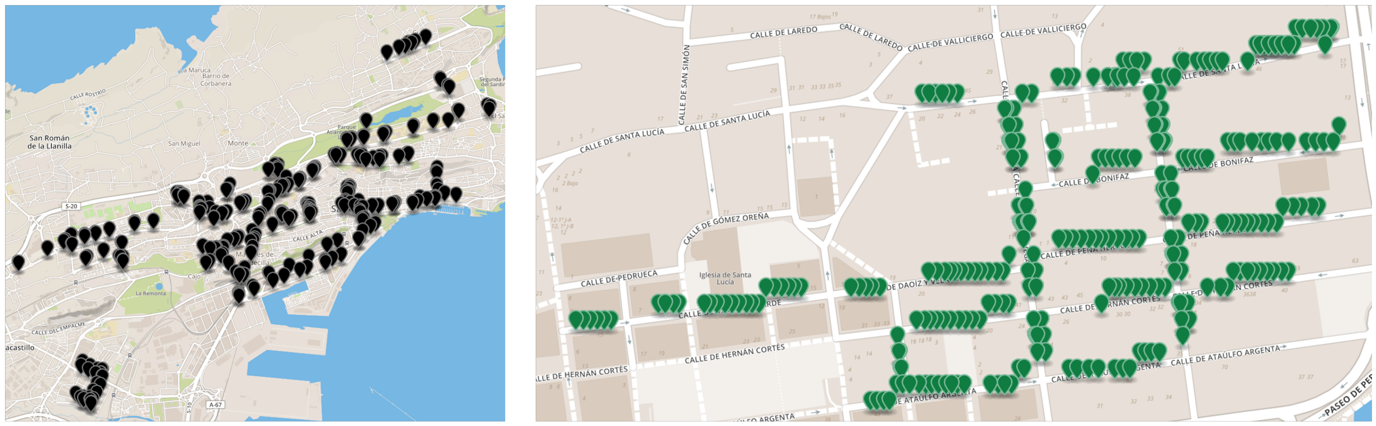
- Development of a parking spot (nearest or nearest trusted) and route (least congested or shortest) recommendation system by exploiting the semantic IoT data of traffic and parking sensors;
- The real-time provision of expected availability (occupancy statistics) of parking areas based on historical IoT data;
- The development of a GDPR-compliant implementation that can work on a privacy-aware environment; and
- A prototype for the demonstration and evaluation of our proposed IoTRec by citizens of a smart city.

This paper is organized as follows. Section II discusses the related work. The semantic data modeling of parking and traffic sensor is presented in Section III, followed by an overview of IoTRec in Section IV. Section V discusses the mechanisms of parking spot and route recommendation, as well as the GDPR-compliant implementation. Section VI presents the provision of real-time expected availability (occupancy statistics) of parking areas with some proposed algorithms using historical IoT data. The IoTRec prototype and its evaluation are presented in Section VII, and then Section VIII concludes the paper and presents future work.

II. Related Work

Much work has been done on smart parking systems since the emergence of smart cities. However, our proposed system is significantly different in terms of recommendations of nearest parking spots, as well as routes by considering the traffic congestion on the streets and the trust scores of parking spots, expected availability (occupancy statistics) of parking areas, a prototype and evaluation. We present the state-of-the-art and explore the need for the features we offer in IoTRec.

Pham et al. [1] developed a cloud-based smart parking system using the Global Positioning System (GPS) coordinate data of vehicles, the number of free parking spots in parking areas, and the distance between parking areas to



(a) (b)

Fig. 1. Deployment of traffic and parking sensors in Santander, Spain (a) traffic sensor deployment, (b) parking sensor deployment.

calculate the costs of a parking request made by a driver. They also developed a prototype using an open-source platform based on Arduino, RFIDs and smartphones. Mainetti et al. [2] designed a smart parking system by integrating RFID, Ultra-High Frequency (UHF) and Wireless Sensor Network (WSN) technologies. This system comprises software features to collect parking spot occupancy and was developed into an application to navigate drivers to the nearest free parking spot. The application also enables users to pay their parking fee through an Near-Field Communication (NFC)-based e-wallet system. It uses Java REST APIs and Google cloud messaging, installed on a central server for managing alerts (such as the expiration of purchased time and the abuse of the reserved space) which promptly alerts the traffic police. This system was demonstrated with a proof-of-concept prototype. However, the authors mainly implemented the prototype and did not evaluate the performance of the parking system.

Hsu et al. [3] proposed SmartValet, a parking guidance system in which drivers can make parking spot reservation by smartphone thirty minutes in advance. SmartValet was developed for outdoor as well as indoor parking. It reserves a parking spot using a vehicle ID. The location of the reserved parking spot is passed to the driver on the map using Dedicated Short-Range Communication (DSRC) technology at the entrance to the parking area. SmartValet implements a navigation system, called the inertial navigation system, to guide the vehicle to the reserved parking spot. The status of the parking spot is updated periodically, ensuring system's accuracy. The authors used the accuracy of the inertial navigation system as a parameter for evaluating the system performance, and GPS accuracy as a parameter for evaluating the system implementation. Similarly, Barone et al. [15] designed an architecture called Intelligent Parking Assistant (IPA) for parking management in Smart Cities. IPA provides information about parking spot availability to drivers and allows them to reserve the most suitable parking spot before their arrival for their destination by using RFIDs and magnetic loops. When a vehicle parks or leaves the parking spot, the magnetic loop and RFID reader identifies such an action and informs the unit controller. The unit controller subsequently updates the status of the parking spot. However, it is not always possible to reserve parking spots in advance because of the regulations in some cities. Therefore, there is a need for a system that can recommend the best possible parking spot instead of reserving parking spots.

Shiyao et al. [16] also proposed and implemented a smart parking system. Their proposed system is based on ZigBee technology which forwards the information at the server through a gateway, and the server subsequently updates the database. For people looking for free parking spots, the application layer of this system obtains the parking spot availability information through the Internet, gathers all the scattered parking spots' information using web services and passes the information to drivers. However, it is a simple application that does not consider complex problems, such as traffic congestion, navigation and expected availability of parking spots. Furthermore, Shiyao et al. did not evaluate the performance of their system.

Lambrinos et al. [17] designed a parking management system for disabled people called DisAssist. This system is built upon the IoT and smart cities' capabilities by integrating smartphones, sensors and mobile/wireless communications. DisAssist offers real-time availability information about disabled parking spots in the area of interest to disabled drivers (or clients) and allows them to reserve parking spots. However, similar to other existing work, DisAssist considers the reservation of parking spots, which is not always possible.

Yavari et al. [18] centered on contextualization and providing personalized parking recommendation to the users by utilizing the data collected in smart cities. Although the authors provide context-aware parking recommendations, however, their main focus is on the contextualized information of users' preferences and habits. We, on the other hand, provide recommendations of parking spot and route based on various metrics, two implementations of parking recommender system for privacy-aware and non-privacy-aware environments, as well as occupancy statistics of parking areas.

Rathore et al. [19] proposed an architecture of smart city using IoT-based Big Data analytics. The authors consider various sensors deployment, such as sensors deployed for

TABLE I
PROPERTIES OF PARKING SPOT, PARKING AREA (ONSTREETPARKING) AND TRAFFIC INFORMATION (TRAFFICFLOWOBSERVED) ENTITIES.

Entity	Property	Type	Description
Parking spot, parking area & traffic information	`id`	String	Unique identifier of a parking spot, parking area and traffic flow
	`type`	String	Type of entity. Must be either a *ParkingSpot, OnStreetParking* or *TrafficFlowObserved*
	`dateModified`	DateTime (ISO8601)	The last modified time of the entity in ISO8601 format (e.g., 1900-12-31T23:59:59.000Z))
	`location`	geo:json	The location coordinates of the entity in geo:json format
Parking spot & parking area	`name`	String	The name of the entity for identification and distinguishing
	`category`	String	The category of the entity (e.g., *onStreet* or *offStreet* for parking spot. *free, forElectricCharging, feeCharged* or *forDisabled* for parking area)
Parking spot	`status`	String	The occupancy status of the parking spot, e.g., *free* or *occupied*
	`refParkingSite`	String	A reference to *OnStreetParking* or *OffStreetParking* based on the value of the *category* property
Parking area	`chargeType`	List<Text>	The type of charges for the parking site, such as free, monthlyPayment, annualPayment and flat rate
	`requiredPermit`	List<Text>	The permit required to park in that area, such as residentPermit, governmentPermit, emergencyVehiclePermit and noPermitNeeded
	`permitActiveHours`	List<Text>	Hours/days during which the permit is required
	`allowedVehicleType`	String	The type of vehicle that is allowed, such as a car, bicycle, motorcycle, bus, small truck, minivan
	`areBordersMarked`	Boolean	Indicates whether parking spots are separated with borders (painted lines) or not
	`totalSpotNumber`	Integer	Number of spots in the parking area
	`occupancyDetectionType`	List<String>	Technique of identifying the occupancy of a parking spot, such as modelBased, singleSpaceDetection, balancing and none
Traffic information	`dateObserved`	DateTime (ISO8601)	The observed date and time of the traffic sensor in ISO8601 format (e.g., 1900-12-31T23:59:59.000Z))
	`intensity`	Integer	The number of vehicles detected during the observation period (e.g., twenty vehicles in one minute)
	`occupancy`	Integer	The fraction of observation time in which vehicles occupy the road
	`roadLoad`	Integer	Estimation of the traffic congestion calculated by the intensity and occupancy

the monitoring of surveillance, parking, weather, water, smart home and vehicles. However, instead of providing smart parking recommendations, their main focus is on smart city architecture using Big Data analytics.

Sadhukhan [20] developed a prototype of IoT-based E-parking system that exploits parking meters to detect improper parking, estimation of the duration of parking spot utilization by a vehicle and automatic payment of parking fee. However, authors do not focus on parking recommendation services.

Chatzigiannakis et al., [21] worked on public key cryptography-based privacy preservation of smart parking system, known as elliptic curve cryptography, that is suitable for resource constraint devices and is platform independent. The authors achieved privacy by using zero knowledge proofs that avoids the exchange of confidential information and evaluated the performance by studying system overhead and execution time. However, the main focus of authors is on preserving the privacy of smart parking system instead of providing parking recommendations.

The work mentioned above either considers the reservation of parking spots or their proposed architecture has been implemented without real-time planned routes, or are designed specifically for disabled persons, or are mainly focused on user-based contextualized information or focused on privacy preservation of existing smart parking system. To the best of our knowledge, none of the systems recommends the nearest parking spots and routes by considering the traffic congestion on the streets and the trust scores of parking spots, offers the expected availability (occupancy statistics) of parking areas and evaluates the system. Additionally, none of the systems uses semantic data models for their parking systems which is nowadays a very important requirement to an IoT system for the interoperability with other IoT applications and systems.

III. SEMANTIC DATA MODELING

The purpose of semantic data modeling in the IoT is to facilitate data interoperability, data sharing and data reuse across cross-domain IoT applications. The IoTRec is designed for the smart parking system of a WISE-IoT project [13] that is mainly focused on the: i) interoperability between two IoT platforms: FIWARE [22] and oneM2M [23]; ii) interoperability between two continents: Europe and Asia (South Korea); and iii) interoperability between cross-domain IoT applications. Therefore, the parking and traffic sensors' data are semantically modeled for interoperability and easy integration with other IoT applications and platforms.

One of the best practices of the semantic web is to reuse existing ontologies and data models instead of creating new ones from the scratch. Therefore, we have reused the data models provided by FIWARE. FIWARE [22] is an open-source IoT platform that offers various data models [24] for the IoT. For our IoTRec, we use FIWARE Data Models that provide a semantic representation of our required entities (e.g., parking spot, parking area and traffic information).

Table I presents a consolidated data model of the parking sensors (parking spot and parking area) and traffic sensors, and Fig. 2 illustrates the ontology of the parking spot, OnStreetParking (parking area) and TrafficFlowObserved (traffic information) entities. The rest of this section provides the details about their semantic data modeling.

A. Semantic Data Modeling of Parking Sensors (Parking Spots and Parking Areas)

There are hundreds of parking sensors deployed in the city of Santander to identify the status of parking spots (i.e., free or occupied). The parking sensor data is structured using an NGSI [25] model in JSON format. A parking sensor represents a parking spot, and parking spots are grouped into various parking areas to provide additional valuable information. The IoTRec offers the expected availability (occupancy statistics) of parking areas using the parking sensor data.

The FIWARE Data Model offers a list of attributes for defining the characteristics of a parking spot entity [26]. We have selected some parking sensor attributes that are mainly required for our case. Similarly, for defining the characteristics of a parking area entity, FIWARE offers a data model, called *OnStreetParking* [26] that provides the attributes required to describe parking areas. Both of these groups of attributes (as entities and their properties) are presented in Table I and in the ontology in Fig. 2.

B. Semantic Data Modeling of Traffic Sensors

Traffic sensors provide traffic information, such as occupancy, intensity and load. The IoTRec utilizes traffic information to calculate the least congested route leading to the recommended parking spot. FIWARE provides a data model of *TrafficFlowObserved* [27] entity for them that offers the required attributes to describe the traffic flow information in a city. We have selected some of the attributes from the *TrafficFlowObserved* data model to define the characteristics of our traffic information entity, and add a new attribute *roadLoad* that is not offered by the *TrafficFlowObserved* FIWARE data model. *roadLoad* estimates the level of traffic congestion calculated by intensity and occupancy parameters. These attributes (as entities and their properties) are presented in Table I and in the ontology presented in Fig. 2.

IV. Overview of The IoT Recommender (IoTRec)

This section presents an overview of the IoTRec for a smart parking system. The IoTRec exploits the semantic IoT data of parking and traffic sensors (see Section III) to offer recommendations of available parking spots and optimal routes leading to the recommended parking spots based on users' preferences (or metrics). For the recommendations of parking spots, the users' preferences (or metrics) include the nearest or nearest trusted parking spot, while for the recommendations of routes, the users' preferences (or metrics) include the least crowded or shorted route.If the user does not provide any preference, then the IoTRec selects the nearest trusted parking spot and the least crowded route, by default. From a recommendation perspective, the IoTRec is different from non-IoT based recommendation systems in four primary aspects. Firstly, the IoTRec mainly considers the IoT data provided by parking and traffic sensors, rather than the data shared or acquired by users' terminals. Google Maps[1], for example, provides a routing path from one point to another by showing the real-time Google Traffic and recommends a least congested route. However, Google Traffic is based on crowdsourced GPS-based locations collected from a large number of users through their smartphones [28]. Google analyzes Google Traffic by calculating the speed of users along the road to calculate the congestion in the streets. The IoTRec considers and analyses traffic sensors data to estimate the level of congestion on the streets to recommend the least crowded route. Hence, if there is a lot of traffic on the streets but very few (or none) of the users has GPS location enabled on their smartphones, Google Traffic will not be able to estimate the congestion, while this is not the case for the IoTRec because it considers IoT data from the traffic sensors. Secondly, the IoTRec provides the recommendations of trusted parking spots (based on users's past experience and sensors quality) by integrating with Trust Monitoring [11], a component of the WISE-IoT project. The Trust Monitoring component calculates trust scores by analyzing users' experience through a feedback mechanism [12] and sensor quality assessment. To the best of our knowledge, this feature does not exist elsewhere in the literature. Thirdly, the IoTRec provides GDPR-compliant, as well as normal implementations that work in a privacy-aware and non-privacy-aware environment. Fourthly, the IoTRec offers the expected availability (occupancy statistics) of parking areas to users and enables the users to analyze the weekly, monthly and yearly statistics of their preferred parking areas in a user-friendly interface. Hence, in addition to the automatic recommendation of parking spots, the IoTRec also allows users to select a parking area themselves by analyzing the occupancy statistics.

To better understand how this system functions, we summarize the main mechanisms in the operation of the IoTRec below. The IoTRec:

- exploits the semantic IoT data of parking sensors and the user's current location to identify the nearest parking spot to the user;
- uses trust scores from the Trust Monitoring component in the recommendation of the parking spot to determine the nearest trusted parking spot;
- utilizes the semantic IoT data of traffic sensors for analyzing the road congestion;
- provides GDPR-compliant implementation for privacy-aware environments;
- exploits the historical semantic IoT data of parking sensors to generate the real-time expected availability (occupancy statistics) of parking areas and shows the statistics in a user-friendly manner to allow a user to analyze parking areas; and
- uses the extracted congestion data and selected parking spot/area to recommend the least congested route to the user.

Fig. 3 presents the proposed IoTRec architecture. The IoTRec first obtains a request (containing the user's current location for GDPR-compliant implementation, or the user's current location and user id for normal implementation) from the smart parking application to find a parking spot and a route. The IoTRec then accesses the semantic IoT data of parking sensors from the FIWARE Orion Context Broker which is an NGSIv2 server implementation and is mainly used for the management of context information and its availability. It allows users to create context elements and to then use updates and queries to manage them. It receives all the heterogeneous IoT sensor measurements, annotates the data using semantic technologies and provides semantic IoT sensor data via REST API. The readers interested in Orion Context Broker are referred to [29] for complete details. Subsequently,

[1]https://www.google.com/maps

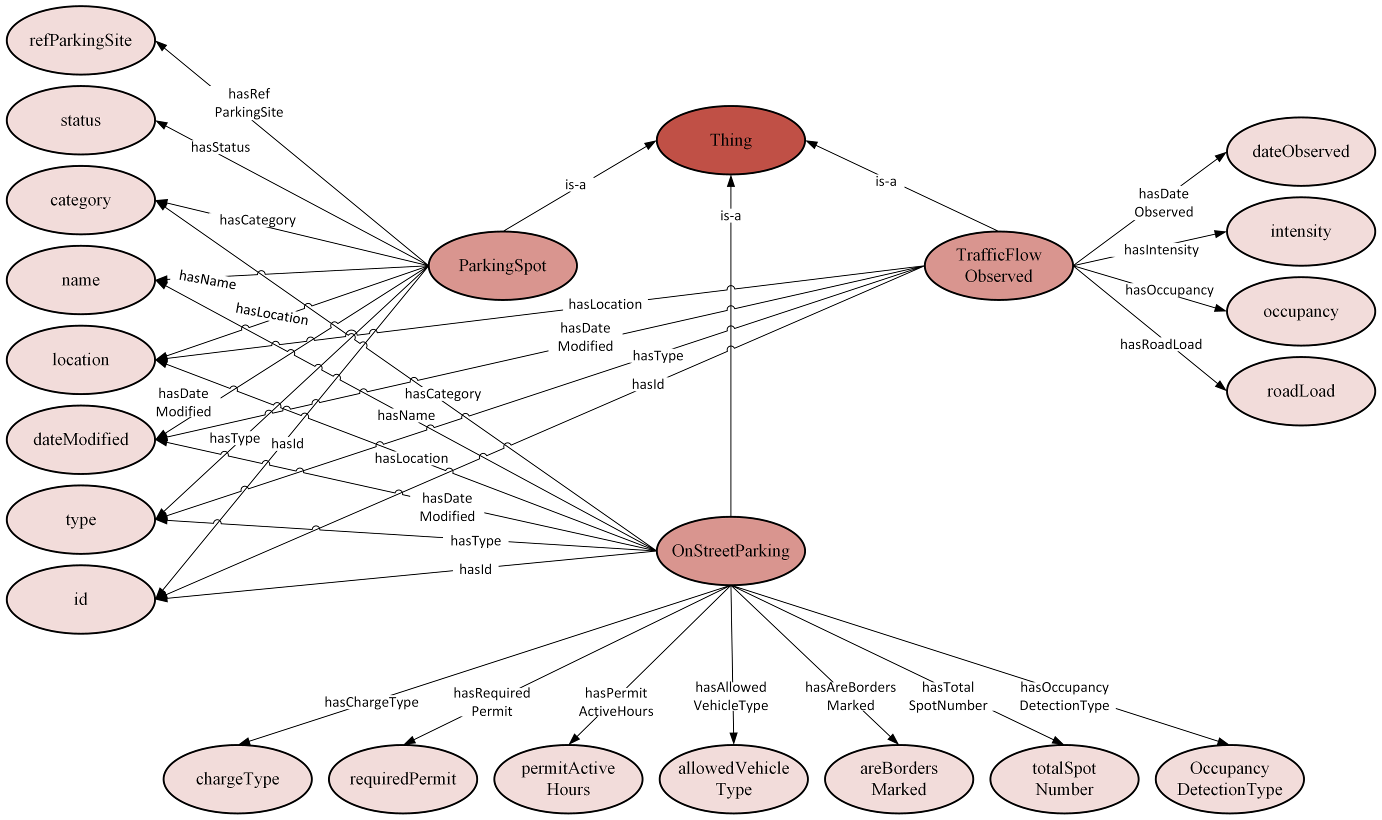


Fig. 2. Ontology of the parking spot, OnStreetParking (parking area) and TrafficFlowObserved (traffic information) entities.

the Trust Monitoring component [11] obtains the trust scores by interacting with Orion Context Broker and user feedback database. The trust scores are used by the IoTRec to find the nearest trusted parking spot. Once the IoTRec finds the most suitable parking spot (i.e., the nearest trusted parking spot), it interacts with a BRouter routing engine [30] to obtain various routes from the user's current location to the selected parking spot. BRouter is a routing engine that offers both online and offline versions and is built on the top of Open Street Maps (OSMs) [31]. It calculates the routes using elevation data and OSMs. After obtaining the routes, the IoTRec accesses the semantic IoT data from the FIWARE Orion Context Broker for appropriate traffic sensors and maps the traffic sensors into the routes obtained from the BRouter routing engine using the algorithm proposed in our previous work [32] on the mapping of sensor and route coordinates. The IoTRec then selects the least congested route and recommends the parking spot and the route to the smart parking application using REST API. The details of how the recommendation of parking spots and routes functions are provided in Section V.

Additionally, the IoTRec also provides the expected availability (occupancy statistics) of parking areas using historical IoT data to allow a user to manually select a parking area. The complete details about the expected availability (occupancy statistics) of parking areas are provided in Section VI.

V. Recommendation of Parking Spots and Routes

This section presents the mechanisms for parking spot and route recommendation. We provide two parking spot recommendation systems, a normal implementation for scenarios

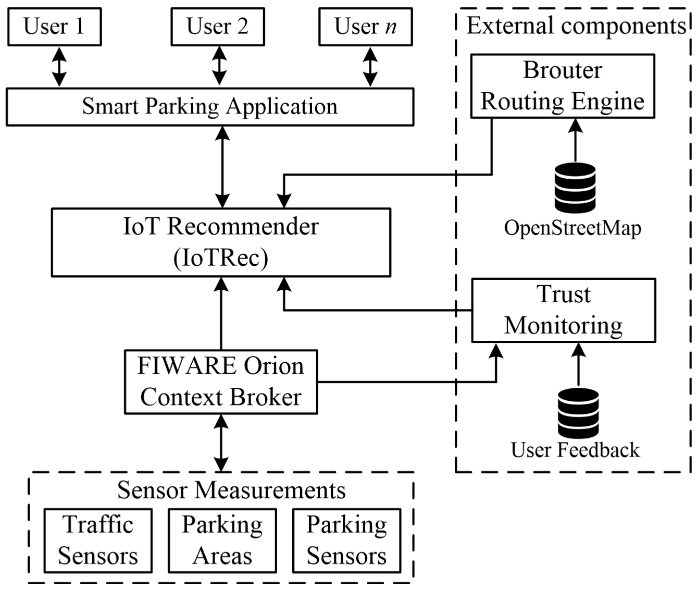


Fig. 3. Architecture of IoTRec system.

that do not operate in a privacy-aware environment (e.g., South Korea) and one for GDPR-compliant implementation in a privacy-aware environments (e.g., Europe).

A. Normal Implementation of Nearest and Nearest Trusted Parking Spot Recommendation

In a normal implementation that does not operate in a privacy-aware environment, when asking for a recommendation, the application provides the user's current location and user id to the IoTRec. Next, the IoTRec obtains the list of available parking spots and calculates the distance from the user's current location to the available parking spots using the BRouter routing engine [30]. It then sorts the available parking spots based on their distance from the user and selects the nearest parking spot for the nearest parking spot recommendation. For the nearest trusted parking spot recommendation, the IoTRec obtains the trust score of the selected nearest parking spot from the Trust Monitoring component associated to the user's id. This trust score is dependent upon the user and on sensor quality. For example, the trust score for the same parking spot for two different users may vary. Similarly, the trust scores for two different parking spots for the same user will also vary. If the obtained trust score is below the defined threshold (the threshold value lies between 0 and 1, and was here set to 0.5 based on experimentations and to find a balance), it indicates that the selected parking spot is not appropriate for the user due to some specific reasons. For example, the user had a bad experience with the selected parking spot in the past, or the selected parking spot has some technical problems (e.g., the sensor is not working properly). After finding an available parking spot that does not have a trust score above the threshold value, the IoTRec selects the second nearest parking spot and obtains its trust score from the Trust Monitoring component. If the trust score is again below the threshold value, the IoTRec will keep checking the parking spots one by one until it finds a parking spot with a trust score that meets the threshold value. When the trust score for the selected parking spot meets the threshold value, the IoTRec selects the parking spot as the recommended parking spot. Subsequently, it finds a route (the least congested or the shortest) from the user's current location to the recommended parking spot (see Section V-C) and sends the recommended parking spot and route back to the smart parking application.

B. GDPR-compliant Implementation of Nearest Trusted Parking Spot Recommendation

We also provide GDPR-compliant implementation for the recommendation of parking spot and route for scenarios that require users' privacy protection and that are operated in a privacy-aware environments. In a normal implementation, the application provides the user's current location and user id to the IoTRec and the IoTRec then passes the user id to the Trust Monitoring component to obtain the trust scores of the parking spots based on the user's past experience and on the sensor quality. However, since the IoTRec receives user ids from the application, it breaches the user's privacy because, through the `user id`, the IoTRec could acquire much information about users, including tracking their movements, and perhaps deducing their routines. This knowledge is a breach of user privacy if the user is not willing to share his or her information. To cope with this problem, we offer privacy-protected and GDPR-compliant implementation in which the IoTRec does not receive `user ids` from the application and does not directly interact with the Trust Monitoring component. Instead, the application first obtains the trust scores of all the parking spots for a specific user from the Trust Monitoring component, then passes the trust scores (comprised of `trusteeId`, `score` and `timestamp`) to the IoTRec. Subsequently, the IoTRec utilizes these trust scores in a similar way as discussed above to select the nearest trusted parking spot. In this manner, the IoTRec does not have any direct knowledge of the users.

The selection of the normal or the GDPR-compliant implementation depends upon the scenario. For instance, if the application/service needs to operate in privacy-aware environment, they should use the GDPR-compliant implementation. Otherwise the applications/services that do not have privacy issues can choose the normal implementation, which reduces the overhead on the application. For example, since the IoTRec is reusable by other applications/services through REST APIs, applications that do not have privacy issues would generally not be willing to undertake the complications of first obtaining the trust scores for the specific user from the Trust Monitoring component at each request and then passing the user's current location and trust scores to the IoTRec. Such applications/services would most likely prefer making the IoTRec responsible for interacting with the Trust Monitoring component and thus should utilize the normal implementation.

C. Least Congested and Shortest Route Recommendation

After identifying the recommended parking spot, the IoTRec selects the route (the least congested or the shortest route based on the user's preference). The least congested route is identified by exploiting the semantic IoT data of traffic sensors that provide the measurements of road load, occupancy and traffic intensity. The traffic sensors are magneto-resistive loops buried under the asphalt and deployed across the city in different roads (as shown in Fig. 1). These sensors identify the presence and mobility of vehicles, providing the real-time measurement of lane occupancy and traffic intensity parameters. To identify the route possibilities, the IoTRec uses a third-party BRouter routing engine [30] to obtain the coordinates of various available routes from the user's current location to the recommended parking spot. Next, the IoTRec applies our previously proposed algorithm [32] to map the traffic sensor coordinates into the route coordinates. After mapping the coordinates, in the case of the shortest route, it selects the shortest available route from user's current location to the recommended parking spot. For the case of the least congested route, it utilizes the road load measurements provided by traffic sensors to find the least congested route. Road load measurements provide the estimated level of congestion on the streets by interpreting the traffic intensity (i.e., the number of vehicles per hour). Since a route is comprised of multiple traffic sensors and hence multiple measurements of the road load, the IoTRec calculates the average of the road load measurements provided by the traffic sensors on each route. Subsequently, it selects the route with the the least average road load (i.e., the least bottlenecked road load) as the least congested route. In the future, we plan to study the bottleneck road loads of routes and develop a route selection mechanism by applying machine learning techniques.

A sequence diagram of the interactions between IoTRec and other systems to calculate the aforementioned parking spot and route recommendations can be seen on Fig. 4.

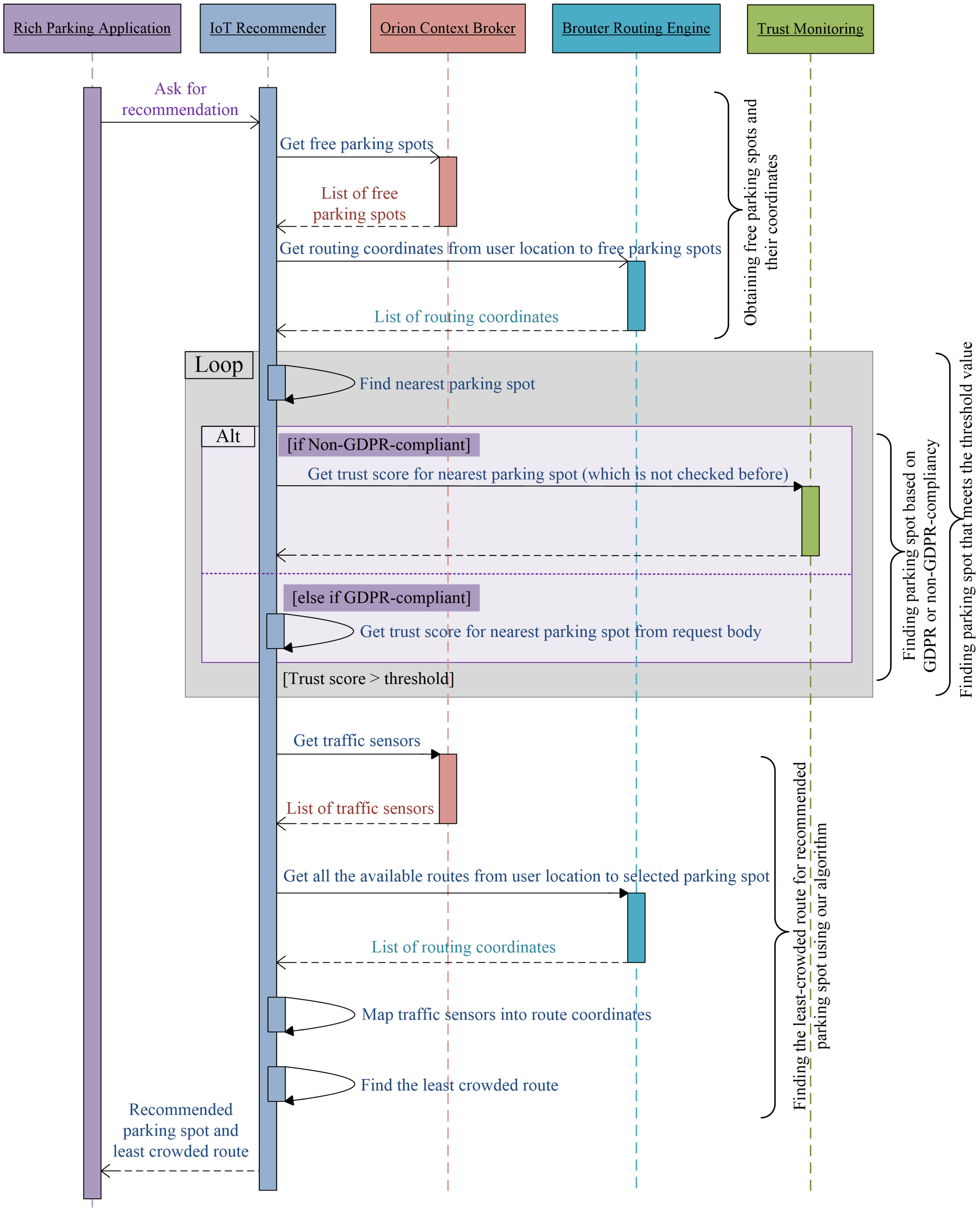

Rich Parking Application
IoT Recommender
Orion Context Broker
Brouter Routing Engine
Trust Monitoring
Ask for recommendation
Get free parking spots
List of free parking spots
Get routing coordinates from user location to free parking spots
List of routing coordinates
Obtaining free parking spots and their coordinates
Loop
Find nearest parking spot
Alt
[if Non-GDPR-compliant]
Get trust score for nearest parking spot (which is not checked before)
[else if GDPR-compliant]
Get trust score for nearest parking spot from request body
[Trust score > threshold]
Finding parking spot based on GDPR or non-GDPR-compliancy
Finding parking spot that meets the threshold value
Get traffic sensors
List of traffic sensors
Get all the available routes from user location to selected parking spot
List of routing coordinates
Map traffic sensors into route coordinates
Find the least crowded route
Finding the least-crowded route for recommended parking spot using our algorithm
Recommended parking spot and least crowded route

Fig. 4. Operation of the IoTRec via sequence diagram for both cases of normal and GDPR-compliant implementation for the recommendations of nearest trusted parking spot and the least crowded route.

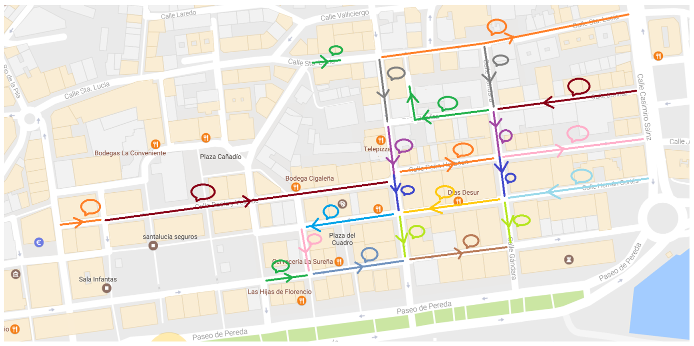


Fig. 5. Aggregation of parking spots into parking areas.

VI. Expected Availability (Occupancy Statistics) of Parking Areas

The IoTRec also offers the real-time expected availability of parking areas by calculating the occupancy statistics to allow users to select a parking area themselves by analyzing the statistics presented in a user-friendly manner. In this section, we explain the mechanisms and the calculation of parking area occupancy statistics. First, we provide an overview of parking area occupancy statistics, then we discuss their calculation and finally present the algorithms utilized to store the information used to calculate the statistics.

A. Overview

To better organize the expected availability (occupancy statistics) of parking areas, parking spots are aggregated into parking areas as presented in Fig. 5. Each line in a different color represents a parking area that comprises multiple nearby parking spots. For parking area occupancy statistics, storing the occupancy data of parking sensors is the first step. The parking sensors' data is updated at a FIWARE Orion Context Broker [26] every $T_{upd} = 120$ seconds. For expected availability (occupancy statistics) of parking areas, the parking sensors' data is collected for the duration of nine months, i.e., from October 2017 to June 2018. For the storage of parking sensors' occupancy data to be used in calculating parking areas occupancy statistics, instead of storing a new record of parking sensor data every T_{upd} duration, we store the status-wise data. For example, if a record R_i for a parking sensor PS_i data having status $PS_{i,status} =$ *free* is stored at time instant T_n, then at the next time instant T_{n+1}, if the status remains the same (i.e., $PS_{i,status} =$ *free* at T_{n+1}), we update the duration of the record R_i rather than adding a new record R_{i+1}. Otherwise, if the status of parking spot PS_i is changed (i.e., $PS_{i,status} =$ *occupied* at T_{n+1}), we add a new record R_{i+1}. This approach helps to avoid one step of the processing to aggregate all the consecutive records having the same status while calculating the duration of the *free* and *occupied* status. Table II presents an example of status-wise storage of two parking sensors' data which shows the ID of the parking sensor (i.e., 3601 and 3602), the status (*free* or *occupied*), start time, end time and the duration.

B. Calculation of Parking Areas' Occupancy Statistics

The status-wise storage of parking sensor data is used in calculating the occupancy statistics of parking areas. At the beginning of each day, we calculate and store the statistics for the previous day in the database. We calculate the parking areas' occupancy statistics for a timing window of every hour to provide more accurate statistical information. We offer three levels of statistic granularity: weekly (past 1 week), monthly (past 4 weeks) and yearly (the last 52 weeks) through a REST API and present them in a user-friendly interface to be analyzed by the end user.

TABLE II
Example of Status-wise Storage of Parking Sensor Data.

ID	Status	Start Time	End Time	Duration (sec)
3601	*occupied*	2018-07-26 08:00:00	2018-07-26 09:30:00	5400
3602	*occupied*	2018-07-26 08:00:00	2018-07-26 10:00:00	7200
3601	*free*	2018-07-26 09:30:00	2018-07-26 10:00:00	1800
3602	*free*	2018-07-26 10:00:00	2018-07-26 10:30:00	1800
3601	*occupied*	2018-07-26 10:00:00	2018-07-26 10:15:00	900
3601	*free*	2018-07-26 10:15:00	2018-07-26 10:30:00	900

For better understanding of this process, we present a scenario in which a user looking for a parking spot wants to analyze the parking area occupancy statistics by himself. The smartphone parking application shows the parking area statistics to the user. These are shown to the user in terms of occupancy percentage, i.e., parking area A was occupied 65% on Monday from 13:00 to 14:00. The user clicks on a parking area on Monday at 13:10 to see the statistics. The user will receive three types of statistics: the statistics of last Monday from 13:00 to 14:00 (i.e., weekly statistics); the statistics of the average value of the last four Mondays from 13:00 to 14:00 (i.e., monthly statistics); the statistics of the average value of last fifty-two Mondays from 13:00 to 14:00 (i.e., yearly statistics). All three levels of statistics are shown to the user in terms of occupancy percentage.

The calculation of parking areas occupancy statistics is divided into two parts. In the first part, we calculate the hourly occupancy duration of each parking spot for each day for fifty-two weeks (one year) and store them in our database. In the second part, we calculate the weekly, monthly and yearly occupancy statistics of parking areas for each hour of the day by aggregating and averaging the hourly occupancy duration of parking spots belonging to parking areas, and store them in another database.

For the first part, to store the parking spot occupancy statistics of 52 weeks for each hour of a day, we present the structure of the database in Table III. In this table, the parking area is the area where the parking spot is situated. There are two parking areas in this table, i.e., HernanCortesCentre and DaoizVelardeEast. Parking spot ids are the identifiers of the parking spots, which are grouped into parking areas. There are four parking spots in this table, i.e., parking spots 3601 and 3602 belonging to parking area HernanCortesCentre, and parking spots 3620 and 3623 belonging to parking area DaoizVelardeEast. The day indicated in the Day column is the day for which the statistics are being calculated, which is Monday in our example. The start time and the end time delineate the one-hour timing window. In our example table, we considered two timing windows of one hour each: 09:00:00−10:00:00 and 10:00:00−11:00:00. Week 1, Week 2, ... Week 52 show the occupancy duration in seconds of

TABLE III
EXAMPLE OF STORAGE OF PARKING SPOT OCCUPANCY STATISTICS FOR 52 WEEKS.

Parking Area	**Parking Spot ID**	**Day**	**Start Time**	**End Time**	**Week 1**	**Week 2**	**Week 3**	**Week 4**	**...**	**Week 52**
HernanCortesCentre	3601	Monday	09:00:00	10:00:00	1500	1800	1500	900	...	1200
HernanCortesCentre	3601	Monday	10:00:00	11:00:00	1800	1200	1500	2100	...	1800
HernanCortesCentre	3602	Monday	09:00:00	10:00:00	1200	2400	1800	2100	...	1200
HernanCortesCentre	3602	Monday	10:00:00	11:00:00	1800	2700	1800	1200	...	1500
DaoizVelardeEast	3620	Monday	09:00:00	10:00:00	2700	2100	3000	3600	...	2700
DaoizVelardeEast	3620	Monday	10:00:00	11:00:00	3000	2700	2400	3600	...	2700
DaoizVelardeEast	3623	Monday	09:00:00	10:00:00	2100	2700	1800	2100	...	3000
DaoizVelardeEast	3623	Monday	10:00:00	11:00:00	2400	2100	2400	3600	...	2400

TABLE IV
EXAMPLE OF STORAGE OF PARKING AREA OCCUPANCY STATISTICS.

Parking Area	**Day**	**Start Time**	**End Time**	**Weekly Stats**	**Monthly Stats**	**Yearly Stats**
HernanCortesCentre	Monday	09:00:00	10:00:00	1500	1650	1560
HernanCortesCentre	Monday	10:00:00	11:00:00	1650	1762	1740
DaoizVelardeEast	Monday	09:00:00	10:00:00	2850	2512	2580
DaoizVelardeEast	Monday	10:00:00	11:00:00	3600	2775	2730

the parking spot during the considered hourly timing window for the specific day. For example, in the first row of Table III, 1500 in the column of Week1 represents the occupancy duration in seconds of parking spot 3601 which is in the HernanCortesCentre parking area from 09:00:00 to 10:00:00 on Monday for the first week of the year.

The second part generates the occupancy statistics of parking areas by aggregating and averaging the hourly occupancy statistics of parking spots within each area. Table IV presents the structure of the database table that stores the final occupancy statistics of parking spots. Let us assume we are currently in week 5. Then in the first row of Table IV, 1500 in the Weekly Stats column shows the average occupancy duration of all the parking spots in the HernanCortesCentre parking area (e.g., 3601 and 3602 in Table III) for Week 4. The value of 1650 in Monthly Stats shows the average occupancy duration of all the parking spots in parking area HernanCortesCentre for Week 4, Week 3, Week 2 and Week 1. Similarly, 1560 in the Yearly Stats column shows the average occupancy duration of all the parking spots in the HernanCortesCentre parking area for the last fifty-two weeks.

For a detailed discussion and complete understanding, we present the algorithms in the next subsection.

C. Algorithms

We present the algorithms for the status-wise storage of parking spots, hourly parking spot occupancy statistics and for the calculation of parking area occupancy statistics.

1) Status-wise Storage of Parking Sensor Data: Algorithm 1 presents the mechanism for status-wise storage of parking spot data. This algorithm runs periodically every $T_{upd} = 120$ seconds and collects parking spot data from an Orion Context Broker and stores the data in the database based on their status (i.e., *free* or *occupied*). In each cycle, this algorithm starts by fetching the parking sensors' data PS_* from an Orion Context Broker. Next, it processes each parking sensor's data PS_i by first fetching the latest record R_i from the status-wise parking spot database corresponding to the parking spot PS_i. As presented in Part I of Algorithm 1, if there is no record in the database corresponding to PS_i (i.e., $R_i = \emptyset$), it shows

Algorithm 1 Status-wise storage of parking sensor data.

```
1: Fetch parking sensor data $PS_*$ from Orion Context Broker every $T_{upd}$ duration;
2: for $PS_i$ in $PS_*$ do
3:     $R_i \leftarrow$ latest record corresponding to $PS_i$ in DB;
4:     /* Part I: The first entry. Add a new record. */
5:     if $R_i = \emptyset$ then
6:         /* It is the first entry */
7:         Add a new record $R_i$ for $PS_i$;
8:         $R_{i,parkingSpotId} \leftarrow PS_{i,id}$;
9:         $R_{i,status} \leftarrow PS_{i,status}$;
10:        $R_{i,startTime} \leftarrow \texttt{currentTime}$;
11:        $R_{i,endTime} \leftarrow \texttt{currentTime}$;
12:        $R_{i,duration} \leftarrow 0$;
13:        /* Part II: The status stays the same. Update the existing record. */
14:    else if $R_{i,status} = PS_{i,status}$ then
15:        Update record $R_i$;
16:        $R_{i,duration} \leftarrow (\texttt{currentTime} - R_{i,startTime})$;
17:        $R_{i,endTime} \leftarrow \texttt{currentTime}$;
18:        /* Part III: The status changes. Update the existing record and add a new record. */
19:    else if $R_{i,status} \neq PS_{i,status}$ then
20:        Update record $R_i$;
21:        $R_{i,duration} \leftarrow (\texttt{currentTime} - R_{i,startTime})$;
22:        $R_{i,endTime} \leftarrow \texttt{currentTime}$;
23:        Add a new record $R_{i+1}$;
24:        $R_{i+1,parkingSpotId} \leftarrow PS_{i,id}$;
25:        $R_{i+1,startTime} \leftarrow \texttt{currentTime}$;
26:        $R_{i+1,endTime} \leftarrow \texttt{currentTime}$;
27:        $R_{i+1,status} \leftarrow PS_{i,status}$;
28:        $R_{i+1,duration} \leftarrow 0$;
29:    end if
30: end for
```

that this is the first entry of the PS_i. Therefore, it adds a new record R_i for PS_i in the database by setting the parking spot Id $R_{i,parkingSpotId}$ and status $R_{i,status}$ to be same as those of the parking spot PS_i (i.e., $PS_{i,id}$ and $PS_{i,status}$). Since it is the first entry, it sets the start time $R_{i,startTime}$ and end time $R_{i,endTime}$ as the current time (`currentTime`), and sets the duration $R_{i,duration}$ as zero, a value which will be updated in the next round. Otherwise, as presented in Part II, if the current status of the parking spot did not change from its previous status and there is a matching record R_i in the database for

Algorithm 2 Calculation of hourly parking spot statistics.

```
1: Input: dateToCalculate, database of status-wise statistics of parking spots;
2: /* Part II: Add hourly parking spot statistics from status-wise statistics */
3: $PS_* \leftarrow$ parking spots data from status-wise DB having $status =$ occupied;
4: dayOfWeek $\leftarrow$ GetDayOfWeek(dateToCalculate);
5: weekOfYear $\leftarrow$ GetWeekOfYear(dateToCalculate);
6: numHoursOfDay $\leftarrow$ 24;
7: for $PS_i$ in $PS_*$ do
8:     parkingArea $\leftarrow$ Mapping($PS_{i,id}$, parkingAreas);
9:     /* Part I: Loop on 24 hours of the day for hourly statistics */
10:    for $h$ in numHoursOfDay do
11:        startHourWindow = $h$;
12:        endHourWindow = $h + 1$;
13:        excessDuration $\leftarrow$ 0;
14:        currentDuration $\leftarrow$ 0;
15:        actualDuration $\leftarrow$ 0;
16:        /* Part II: Add an hourly entry $R_i$ of parking spot in hourly stats DB */
17:        if no $R_i$ for $PS_i$ in hourly stats DB then
18:            $R_{i,parkingArea} \leftarrow$ parkingArea;
19:            $R_{i,parkingSpotId} \leftarrow PS_{i,id}$;
20:            $R_{i,day} \leftarrow$ dayOfWeek;
21:            $R_{i,startTime} \leftarrow$ startHourWindow;
22:            $R_{i,endTime} \leftarrow$ endHourWindow;
23:        end if
24:        /* Part III: Calculate the occupancy duration */
25:        if startTimeWindow $> PS_{i,startTime}$ then
26:            excessDuration$\leftarrow$ startTimeWindow$-PS_{i,startTime}$;
27:        end if
28:        if $PS_{i,endTime} >$ endTimeWindow then
29:            excessDuration.append($PS_{i,endTime}-$ endTimeWindow);
30:        end if
31:        currentDuration = $PS_{i,duration}-$ excessDuration;
32:        Fetch $R_i$ corresponding to $PS_i$;
33:        occupancyValue $\leftarrow R_{i,weekOfYear}$;
34:        actualDuration $\leftarrow$ occupancyValue + currentDuration;
35:        Update $R_i|R_{i,weekOfYear} \leftarrow$ actualDuration;
36:    end for
37: end for
```

PS_i with the same status (i.e., $R_{i,status} = PS_{i,status}$), it updates the record R_i by calculating the new duration (i.e., currentTime$-R_{i,startTime}$) and updating the end time $R_{i,endTime}$ to the currentTime. Finally, as presented in Part III, if the status of the latest record $R_{i,status}$ is different from the current status $PS_{i,status}$, it first updates the duration $R_{i,duration}$ and end time $R_{i,endTime}$ as explained in Part II, and then it adds a new record R_{i+1} as explained in Part I.

The same process continues for all the parking spots. At the end of each cycle, the status-wise database is maintained as presented in Table II.

2) Calculation of Hourly Parking Spot Occupancy Statistics: Algorithm 2 presents the mechanism for calculating hourly parking spot occupancy statistics. This algorithm takes as input the dateToCalculate (date of the previous day because this algorithm starts at the beginning of each day) and the status-wise data of parking spots. The algorithm starts by fetching the status-wise occupied parking sensors' data PS_* and initializing the dayOfWeek, weekOfYear and numHoursOfDay. Next, it processes each record of parking spot PS_i in PS_*. It first extracts the parkingArea from the mapping of the parking spot id and parking areas (i.e., Mapping($PS_{i,id}$, parkingAreas)) and runs a loop h for each hour of the day in numHoursOfDay, as presented in Part I of Algorithm 2. To store the hourly occupancy statistics and calculate the occupancy duration in each hour, it then sets the startHourWindow as h, endHourWindow as $h+1$, and initializes excessDuration, currentDuration and actualDuration to zero. excessDuration is the duration outside of the pre-set window time. For example, let us consider the first row of Table II with totalDuration = 5400. If we want to calculate hourly statistics from 08:00:00$-$09:00:00, the excessDuration is 1800 seconds (i.e., 09:00:00$-$09:30:00) which we need to exclude in our calculation, and hence, the currentDuration is totalDuration - excessDuration (i.e., 5400$-$1800=3600 seconds). Finally, the actualDuration is the sum of the previous occupancy duration occupancyValue and the currentDuration.

Part II of Algorithm 2 adds an hourly entry R_i of parking spots in the hourly statistics database without a duration. The duration will be calculated in Part III of the algorithm. Part III first checks whether the startTimeWindow is higher than the parking sensor's start time $PS_{i,startTime}$, and if so, it calculates the excessDuration by taking the difference of the startTimeWindow from the parking sensor start time $PS_{i,startTime}$. Part III also checks whether the parking sensor end time $PS_{i,endTime}$ is higher than the endTimeWindow, which is the case in our considered example, e.g., endTimeWindow=9:00:00, while the parking sensor end time $PS_{i,endTime}$ =9:30:00. Hence, it updates the excessDuration by appending the difference of the parking sensor end time $PS_{i,endTime}$ to the endTimeWindow and calculates the currentDuration by subtracting the excessDuration from the total duration $PS_{i,duration}$. Finally, to calculate the actual duration for the current week of the year for the current day, it first fetches the record R_i corresponding to PS_i, obtains the occupancyValue (i.e., the sum of previous occupancy duration) and calculates the actualDuration by taking the sum of the currentDuration and the occupancyValue. Subsequently, it updates the occupancy statistics of the current week of the year of the current day $R_{i,weekOfYear}$ with the actualDuration. The same process follows for all the parking spots. In the end, this algorithm creates a database as shown in Table III.

3) Calculation of Parking Areas' Occupancy Statistics: Algorithm 3 presents the mechanism of calculation of parking areas' occupancy statistics. This algorithm is comprised of two main parts. In the first part, it creates arrays of weekly, monthly and yearly occupancy statistics for each hour of the parking spots associated with the parking areas. Each element within an array corresponds to the occupancy duration of all the

Algorithm 3 Calculation of parking areas occupancy statistics.

```
1: /* Part I: Create arrays of weekly, monthly and yearly occupancy statistics for each hour of the parking spots associated to parking areas.*/
2: parkingAreasJson ← new JSON;
3: currentWeek ← GetWeekOfYear();
4: $RS$ ← Occupancy stats of parking spots from hourly stats DB;
5: for $PS_i$ in $RS$ do
6:     parkingAreaId ← $PS_{i,parkingAreaId}$;
7:     day ← $PS_{i,day}$;
8:     startTime ← $PS_{i,startTime}$;
9:     endTime ← $PS_{i,endTime}$;
10:    /* Part I(a): Define nested JSON objects/arrays for parking area, daily, hourly, weekly, monthly and yearly stats for those not already defined*/
11:    parkingAreasJson.parkingArea ← new JSON;
12:    parkingAreasJson.parkingArea.day ← new JSON;
13:    parkingAreasJson.parkingArea.day.hour ← new JSON;
14:    parkingAreasJson.parkingArea.day.hour.weeklyStatArray ← new JSON;
15:    parkingAreasJson.parkingArea.day.hour.monthlyStatArray ← new JSON;
16:    parkingAreasJson.parkingArea.day.hour.yearlyStatArray ← new JSON;
17:    /* Part I(b): Calculate weekly, monthly and yearly average stats*/
18:    $j$ ← currentWeek−1;
19:    weeklyStatArray.append($PS_{i,week_j}$);
20:    monthlyStatArray.append(AVG($\sum_{k=j-4}^{j} PS_{i,week_k}$));
21:    yearlyStatArray.append(AVG($\sum_{k=j-52}^{j} PS_{i,week_k}$));
22: end for
23: /* Part II: Calculate the accumulated weekly, monthly and yearly occupancy statistics of parking areas for each hour of the day and each day of the week and store the statistics in the database*/
24: for parkingAreaJson in parkingAreasJson → dayJson in parkingAreaJson → hourJson in dayJson do
25:    /* Part II(a): Calculate the accumulated weekly, monthly and yearly occupancy statistics of parking areas*/
26:    parkingArea ← key(parkingAreaJson);
27:    day ← key(dayJson);
28:    hour ← key(hourJson);
29:    weeklyStatArray ← $\mathrm{hourJson}_{weeklyStatsArray}$;
30:    monthlyStatArray ← $\mathrm{hourJson}_{monthlyStatsArray}$;
31:    yearlyStatArray ← $\mathrm{hourJson}_{yearlyStatsArray}$;
32:    $\mathrm{length}_w$ ← length(weeklyStatArray);
33:    $\mathrm{length}_m$ ← length(monthlyStatArray);
34:    $\mathrm{length}_y$ ← length(yearlyStatArray);
35:    avgWeeklyStat ← AVG($\sum_{k=1}^{\mathrm{length_w}}$weeklyStatArray);
36:    avgMonthlyStat ← AVG($\sum_{k=1}^{\mathrm{length_m}}$monthlyStatArray);
37:    avgYearlyStat ← AVG($\sum_{k=1}^{\mathrm{length_y}}$yearlyStatArray);
38:    /* Part II(b): Store the statistics in the database*/
39:    Add a new record $R_l$ in parking areas stats DB;
40:    $R_{l,parkingArea}$ ← parkingArea;
41:    $R_{l,day}$ ← day;
42:    $R_{l,startTime}$ ← startTime (fetched from hour);
43:    $R_{l,endTime}$ ← endTime (fetched from hour);
44:    $R_{l,weeklyStats}$ ← avgWeeklyStat;
45:    $R_{l,monthlyStats}$ ← avgMonthlyStat;
46:    $R_{l,yearlyStats}$ ← avgYearlyStat;
47: end for
```

parking spots within a parking area. For example, if parking area A has five parking spots, then in this part, the weekly, monthly and yearly arrays will have five elements each. In the second part, this algorithm calculates the average weekly, monthly, and yearly occupancy statistics of parking areas for each hour of the day and each day of the week and stores the occupancy statistics in the database.

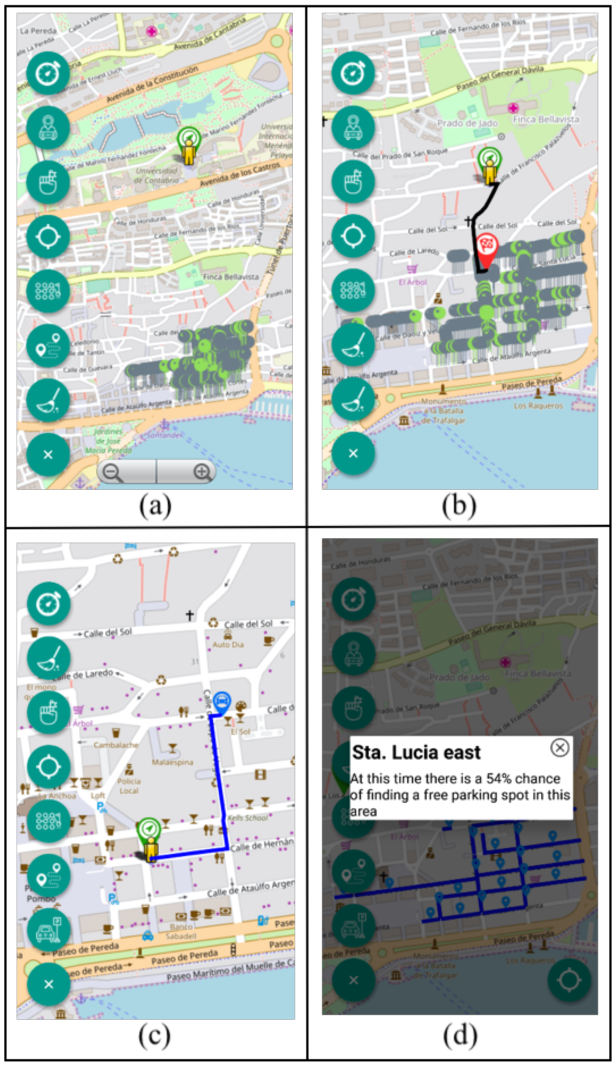
The algorithm starts by initializing the variables `parkingAreaJson`, `currentWeek` and the result set RS of occupancy statistics of parking spots from the hourly statistics database. As presented in Part I(a) of Algorithm 3, the algorithm processes each parking spot PS_i in the result set RS and defines nested JSON objects/arrays (for those not already defined) of `parkingArea`, `day`, `hour`, `weeklyStatArray`, `monthlyStatArray` and `yearlyStatArray` statistics. Part I(b) calculates the average weekly, monthly and yearly statistics. For weekly statistics, since that is comprised of just one past week which does not require an average, it appends the occupancy statistics of the past week. For monthly statistics, it first takes the sum of the occupancy statistics of last four weeks, then takes their average and appends that value into a monthly statistics array. The yearly statistics are calculated similar to the monthly statistics, but for fifty-two weeks instead of four weeks. Similarly, the weekly, monthly and yearly occupancy statistics for other parking spots belonging to the same parking area are also appended into the same weekly, monthly, and yearly arrays.

Part II(a) of Algorithm 3 calculates the accumulated weekly, monthly and yearly occupancy statistics of parking areas. It iterates on each JSON object in `parkingAreasJson` (i.e., `parkingAreaJson`, `dayJson` and `hourJson`), and obtains the ids of `parkingAreas`, `days` and `hours`, respectively, from their `keys`. Next, it obtains the `weeklyStatArray`, `monthlyStatArray` and `yearlyStatArray` from `hourJson` and calculates their lengths. Then for weekly statistics, it first takes the sum of all the weekly statistics in the array and takes their average, which is the accumulated weekly occupancy statistics of a `parkingArea` for the particular `hour` of the specific `day`. The monthly and yearly accumulated statistics are calculated in a similar fashion. Finally, as presented in Part II(b) of the algorithm, the accumulated weekly, monthly and yearly statistics are stored in the database of parking area occupancy statistics, which is used to generate the REST API for parking area occupancy statistics. Table IV presents a snapshot of the final parking area occupancy statistics database where the weekly, monthly and yearly statistics are presented for each hour of the day and each day of the month.

VII. The Prototype and Evaluation

A. The Prototype

We developed an Android application, called Rich Parking, as a prototype for the IoTRec in the WISE-IoT project. Rich Parking application was tested and evaluated as a prototype by the citizens of Santander, Spain. The prototype serves as the first step towards developing a fully-fledged application



Sta. Lucia east

At this time there is a 54% chance of finding a free parking spot in this area

Fig. 6. Screenshots of the prototype Rich Parking application: (a) view of free and occupied parking spots; (b) recommendation of a parking spot and a route; (c) walking route to the parked car; and (d) an example of parking areas occupancy statistics.

to improve the parking experience of users in the city. The Rich Parking application provides various functionalities to the users using the services offered by the IoTRec through REST APIs. The screenshots of these functionalities are depicted in Fig. 6. We present the features of the Rich Parking application in this section.

1) Show Parking Spots: The Rich Parking application allows users to see the status of parking spots. To request the status of parking spots, a user clicks the corresponding button. The Rich Parking application calls the REST APIs of the IoTRec for free and occupied parking spots and obtains the list of free and occupied parking spots. It subsequently shows the free parking spots as green markers and occupied parking spots as grey markers, as presented in Fig. 6(a).

2) Recommendation of Parking Spot and Route: A user can request a parking spot recommendation and that of a route from their current location by specifying their preferences (e.g., for a parking spot: nearest or nearest trusted parking spot, and for a route: least crowded or shortest) in the application. The application then calls the required REST API of the IoTRec of the parking spot and route recommendation and presents the recommended parking spot and route to the user, as presented in Fig. 6(b).

3) Walking Route to the Parked Car: After a user has parked his car in the recommended parking spot and saved the location of the car in the application, he can later request a walking route from his current location to his parked car. The application then calls the REST API of the IoTRec for the walking route to the parked car and shows the walking route to the parked car on the application screen, as presented in Fig. 6(c).

4) Parking Area Occupancy Statistics: The Rich Parking application shows the parking areas to the user if the user wants to analyze the statistics manually, as presented in Fig. 6(d). When the user clicks on any parking area, the application calls the REST API of the IoTRec for parking area occupancy statistics, obtains the parking area statistics for the current day and current hour for the selected parking area, calculates the occupancy statistics in percentage, and presents the results to the user. For example, in Fig. 6(d), when the user clicks on the area `Sta. Lucia east`, the application shows a popup to the user that indicates there is a 54% chance of finding a free parking spot in this parking area.

B. Evaluation Overview

To evaluate our IoTRec system, we developed a prototype, the Rich Parking android application, which was shared with the citizens of Santander, Spain. For expected availability (occupancy statistics) of parking areas, the parking sensors' data is collected for the duration of nine months, i.e., from October 2017 to June 2018. We approached a higher number of Santander's citizens through various meetups requesting volunteers for the evaluation of our application. We then conducted a meeting with the volunteered citizens willing to participate in the evaluation and explained to them how to use and evaluate the application. A total of 41 citizens of Santander committed to being engaged in our evaluation, and 30 of them installed our application from the Google Play store. To conduct our evaluation, we designed a questionnaire that participants completed at the end of evaluating our application. Out of the 30 participants, 27 evaluated our application by answering our questionnaire. Table V presents the questionnaire with its possible answers to these questions for the evaluation of our prototype.

C. Evaluation Results

The evaluations results are based on the questionnaires completed by the 27 citizen participants. We present the evaluation results for each question in the questionnaire.

1) Quality: Fig. 7 presents the evaluation results of the quality of the recommended routes and parking spots. It shows a good response as 89% and 81% of the involved citizens were satisfied (identified by their positive ratings of 3, 4 and 5-stars) with the quality of the recommended routes and parking spots, respectively. 78% and 63% of the participants found the quality of recommended routes and parking spots, respectively

TABLE V
QUESTIONNAIRE FOR THE EVALUATION OF OUR PROTOTYPE, RICH PARKING APPLICATION.

No.	Question	Possible Answers
1	Quality of the routes and parking information?	1-star (very bad) to 5-stars (very good)
2	Functionality of show parking spots?	1-star (little useful) to 5-stars (very useful)
3	Functionality of route calculation?	1-star (little useful) to 5-stars (very useful)
4	Functionality of walking route to the parked car?	1-star (little useful) to 5-stars (very useful)
5	Functionality of parking statistics?	1-star (little useful) to 5-stars (very useful)
6	Easy to navigate?	1-star (very difficult) to 5-stars (very easy)
7	Easy to calculate a route?	1-star (very difficult) to 5-stars (very easy)
8	Easy to analyze a route?	1-star (very difficult) to 5-stars (very easy)
9	Provided data of parking spots and statistics are reliable?	No, Probably No, I do not know, Probably Yes, Yes

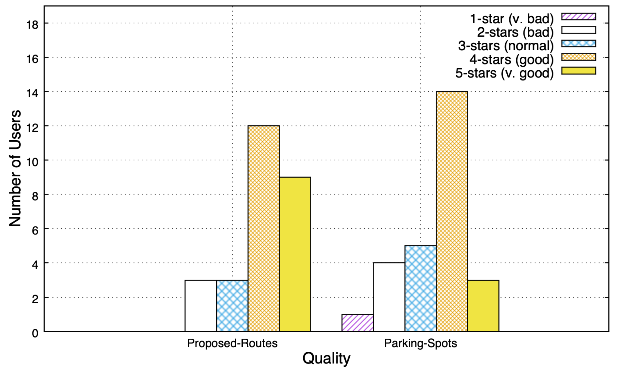


Fig. 7. Evaluation results of the quality of recommended routes and parking spots.

to be good/very good (4 and 5-stars). Overall, these evaluation results give us a good indication of the high quality of the recommended routes and parking spots and the satisfaction of the users.

2) *Functionality:* Fig. 8 shows the evaluation results of the recommendation of parking spots, routes calculation, walking route and parking area occupancy statistics in terms of usefulness, with a scale of 1-star (very useless) to 5-stars (very useful). For the functionalities of the recommendations of parking spots, route calculation, walking routes and parking area occupancy statistics, around 93%, 96%, 93% and 85%, respectively, of the participants found these functionalities to be useful (identified by positive ratings of 3, 4 and 5-stars), and 85%, 81%, 70% and 78% of them found these respective functionalities to be useful/very useful (ratings of 4 and 5-stars).

3) *Ease of Use:* Fig. 9 presents the evaluation results of the ease of use of the navigation, route calculation and route analysis in the application. For the ease of use of the navigation, route calculation and route analysis, around 81%, 67% and 74% of the participants gave positive ratings. More specifically, 67%, 52% and 48% of the participants found it easy or very easy to use the navigation, route calculation and route analysis, respectively. This result indicates that we need to focus more on improving the application interface for route calculation and route analysis.

4) *Reliability:* The results for the reliability question about whether the participants believe that the provided data of parking spot and parking area occupancy statistics are reliable are shown in Fig. 10. Based on the interaction of the involved

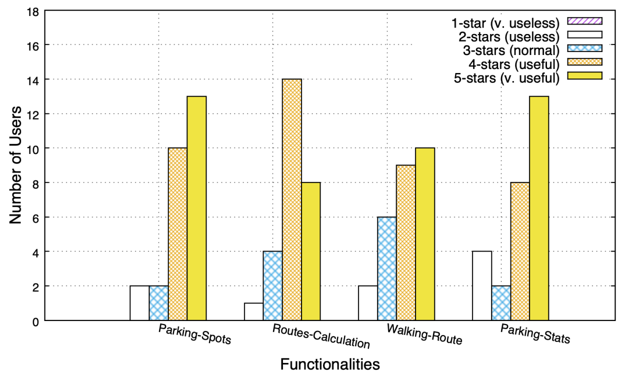


Fig. 8. Evaluation results of the functionalities of the recommendations for parking spot availability, route calculation, walking route to a parked car and parking area statistics.

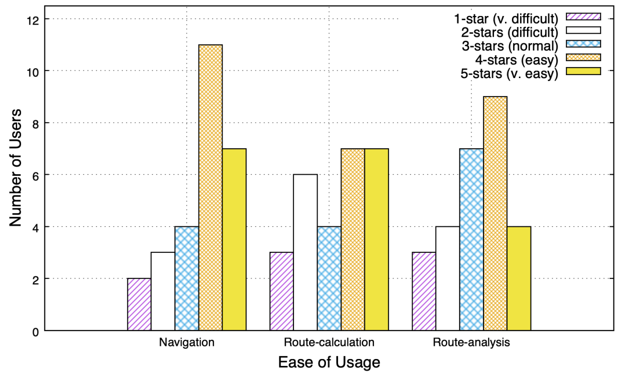


Fig. 9. Evaluation results of the ease of use of the navigation, route calculation and route analysis.

citizens with the application, close to 85% of them believe that the data provided about parking spot and parking area occupancy statistics are reliable, which is a good indication.

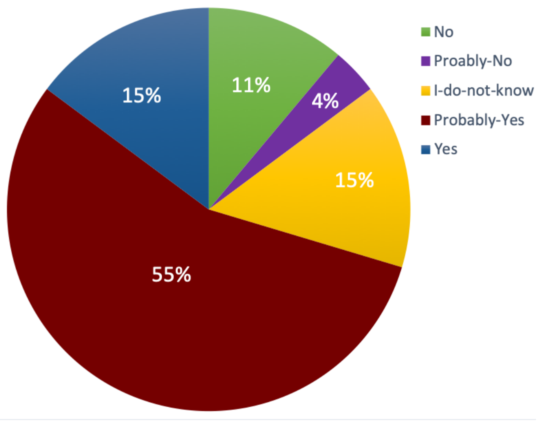


Fig. 10. Evaluation results of the reliability of the provided parking spot data and parking area occupancy statistics.

VIII. Conclusion and Future Work

This paper has presented the development, implementation and evaluation of an IoT Recommender (IoTRec) for a smart parking system. The main purpose of this system is to provide a better experience to both users and managers in terms of vehicular mobility in a city by relying on IoT technologies.

The IoTRec mainly provided the GDPR-compliant recommendations of a parking spot (nearest or nearest trusted parking spot) and a route (least congested or shortest route) leading to the recommended parking spot, as well as the real-time provision of the expected occupancy of parking areas based on the historical IoT data. Finally, the proposed system was evaluated through a prototype, called Rich Parking, which was utilized and tested by the citizens of Santander, Spain.

As future work, we are planning to study the bottleneck of road loads along the calculation of the least congested route and add the ability to select a better route by applying machine learning techniques. We also have a strong plan to extend IoTRec in future projects and enhance performance evaluation in terms of average service/response time with respect to different number of users, as well as adding a qualitative comparison between our proposed IoTRec with some other existing systems. Additionally, we plan to extend the parking area occupancy statistics to more than one year. We also intend to apply machine learning techniques to predict the state of parking spots in the near future and to consider that in the recommendation of parking spots. We will also explore the possibility to integrate Google Analytics with our smart parking smartphone application to further analyse when and how often the users interact with it.

IX. Acknowledgment

This research was supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 723156, the Swiss State Secretariat for Education, Research and Innovation (SERI) and the South-Korean Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MISP) (No. R7115-16-0002). This research was partially funded by the Spanish Government (MINECO) under Grant Agreement No. RTI2018-093475-AI00, project FIERCE (Future Internet Enabled Resilient smart CitiEs)

References

[1] T. N. Pham, M.-F. Tsai, D. B. Nguyen, C.-R. Dow, and D.-J. Deng, "A Cloud-based Smart-parking System based on Internet-of-Things Technologies," *IEEE Access*, vol. 3, pp. 1581–1591, 2015.

[2] L. Mainetti, L. Palano, L. Patrono, M. L. Stefanizzi, and R. Vergallo, "Integration of RFID and WSN technologies in a Smart Parking System," in *22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2014, pp. 104–110.

[3] C. W. Hsu, M. H. Shih, H. Y. Huang, Y. C. Shiue, and S. C. Huang, "Verification of Smart Guiding System to Search for Parking Space via DSRC Communication," in *12th International Conference on ITS Telecommunications*, 2012, pp. 77–81.

[4] S. Poslad, S. E. Middleton, F. Chaves, R. Tao, O. Necmioglu, and U. Bugel, "A Semantic IoT Early Warning System for Natural Environment Crisis Management," *IEEE Transactions on Emerging Topics in Computing*, vol. 3, no. 2, pp. 246–257, 2015.

[5] G. Vojkovic, "Will the gdpr slow down development of smart cities?"

[6] E. Mougiakou and M. Virvou, "Based on GDPR Privacy in UML: Case of e-Learning Program," in *2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 2017, pp. 1–8.

[7] P. Sotres, J. Lanza, L. Sanchez, J. R. Santana, C. Lopez, and L. Munoz, "Breaking Vendors and City Locks through a Semantic-enabled Global Interoperable Internet-of-Things System: A Smart Parking Case," *Sensors*, vol. 19, no. 2, p. 229, 2019.

[8] P. Sotres, J. R. Santana, L. Sanchez, J. Lanza, and L. Munoz, "Practical Lessons from the Deployment and Management of a Smart City Internet-of-Things Infrastructure: The SmartSantander Testbed Case," *IEEE Access*, vol. 5, pp. 14 309–14 322, 2017.

[9] L. Sanchez, L. Munoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis *et al.*, "Smart-Santander: IoT experimentation over a smart city testbed," *Computer Networks*, vol. 61, pp. 217–238, 2014.

[10] Smart Santander Data Sets. [Online]. Available: http://datos.santander.es/data/

[11] H. Baqa, N. B. Truong, N. Crespi, G. M. Lee, and F. Le Gall, "Quality of Information as an Indicator of Trust in the Internet of Things," in *IEEE 17th International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 2018, pp. 204–211.

[12] D. Wuest, F. Fotrousi, and S. Fricker, "Combining Monitoring and Autonomous Feedback Requests to Elicit Actionable Knowledge of System Use," in *International Working Conference on Requirements Engineering: Foundation for Software Quality (RefsQ)*. Springer, 2019, pp. 209–225.

[13] Worldwide interoperability for semantics iot (wise-iot). [Online]. Available: http://wise-iot.eu/en/home/

[14] P. Sotres, C. L. de la Torre, L. Sanchez, S. Jeong, and J. Kim, "Smart City Services Over a Global Interoperable Internet-of-Things System: The Smart Parking Case," in *Global Internet of Things Summit (GIoTS)*, 2018, pp. 1–6.

[15] R. E. Barone, T. Giuffre, S. M. Siniscalchi, M. A. Morgano, and G. Tesoriere, "Architecture for Parking Management in Smart Cities," *IET Intelligent Transport Systems*, vol. 8, no. 5, pp. 445–452, 2013.

[16] C. Shiyao, W. Ming, L. Chen, and R. Na, "The research and implement of the intelligent parking reservation management system based on zigbee technology," in *Sixth International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*. IEEE, 2014, pp. 741–744.

[17] L. Lambrinos and A. Dosis, "DisAssist: An Internet of Things and Mobile Communications Platform for Disabled Parking Space Management," in *IEEE Global Communications Conference (GLOBECOM)*, 2013, pp. 2810–2815.

[18] A. Yavari, P. P. Jayaraman, and D. Georgakopoulos, "Contextualised Service Delivery in the Internet of Things: Parking Recommender for Smart Cities," in *IEEE 3rd World Forum on Internet of Things (WF-IoT)*, 2016, pp. 454–459.

[19] M. M. Rathore, A. Ahmad, and A. Paul, "IoT-based Smart City Development using Big Data Analytical Approach," in *IEEE International Conference on Automatica (ICA-ACCA)*, 2016, pp. 1–8.

[20] P. Sadhukhan, "An IoT-based E-parking System for Smart Cities," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017, pp. 1062–1066.

[21] "A Privacy-preserving Smart Parking System using an IoT Elliptic Curve based security platform, author=Chatzigiannakis, Ioannis and Vitaletti, Andrea and Pyrgelis, Apostolos, journal=Computer Communications, volume=89, pages=165–177, year=2016, publisher=Elsevier."

[22] FIWARE. Last accessed: March 2019. [Online]. Available: https://www.fiware.org

[23] "oneM2M - Standards for M2M and the Internet of Things," accessed: May 2016. [Online]. Available: http://onem2m.org/

[24] FIWARE Data Models. Last accessed: March 2019. [Online]. Available: https://fiware-datamodels.readthedocs.io/en/latest/Parking/doc/introduction/index.html

[25] OMA, "Open Mobile Alliance, NGSI Context Management," 2012, last accessed: March 2019. [Online]. Available: http://www.openmobilealliance.org/release/NGSI/V1_0-20120529-A/OMA-TS-NGSI_Context_Management-V1_0-20120529-A.pdf

[26] FIWARE Data Model for Parking. Last accessed: March 2019. [Online]. Available: https://fiware-datamodels.readthedocs.io/en/latest/Parking/doc/introduction/index.html

[27] FIWARE Data Model for Traffic Flow. Last accessed: March 2019. [Online]. Available: https://fiware-datamodels.readthedocs.io/en/latest/Transportation/TrafficFlowObserved/doc/spec/index.html

[28] D. Barth, "The Bright Side of Sitting in Traffic: Crowdsourcing Road Congestion Data," *Google Official Blog*, 2009.

[29] Orion Context Broker. Last accessed: April 2019. [Online]. Available: https://fiware-orion.readthedocs.io/en/master/

[30] Brouter Offline Routing Engine. Last Accessed: November 2018. [Online]. Available: http://brouter.de/brouter/offline.html

[31] Open street maps. Last Accessed: November 2018. [Online]. Available: https://www.openstreetmap.org

[32] Y. Saleem and N. Crespi, "Mapping of Sensor and Route Coordinates for Smart Cities," in *2018 IEEE 42nd Annual Computer Software and*

Applications Conference (COMPSAC), vol. 1. IEEE, 2018, pp. 570–576.

Yasir Saleem received the B.S. degree in Information Technology from the National University of Sciences and Technology (NUST), Islamabad, Pakistan, in 2012, and the M.Sc. degrees in Computer Science by Research from Sunway University, Malaysia, and Lancaster University, U.K. (under a dual degree program) in 2015. He is currently pursuing the Ph.D. degree with the Service Architecture Laboratory, Institut Mines-Telecom, Telecom SudParis, France. His research interests include Internet of Things, Semantic Web, social Internet of Things, cognitive radio networks, and wireless sensor networks. He served in the TPC for IEEE MELECON 2016 and FMEC 2016 conferences He is also a Reviewer of various journals, such as the IEEE Wireless Communications, the IEEE Communications Magazine, Pervasive and Mobile Computing, Ad Hoc Networks, Computer Networks, the Journal of Network and Computer Applications, Artificial Intelligence Review, the IEEE ACCESS, Wireless Networks, and many others. He also has been a Reviewer for various conferences, such as IEEE ICC 2013, IEEE Globecom 2014, IWCMC 2015, MICC 2015, ICIN 2017, and ISNCC 2017. http://www.yasirsaleem.com/

Pablo Sotres is a senior research fellow at the University of Cantabria, Spain. He has been involved in several different international projects framed under the smart city paradigm, such as SmartSantander; and related to inter-testbed federation, such as Fed4FIRE, Fed4FIRE+, Wise-IoT and FED4SAE. Among his current research interests are IoT-enabled smart cities, M2M communications and network security.

Samuel Fricker is a Professor of Software Engineering at University of Applied Sciences and Arts Northwestern Switzerland (FHNW). He has received his doctoral degree in Informatics from University of Zurich, a Master's degree in Software Engineering from the Royal Institute of Technology in Stockholm, and a Swiss federal diploma in Computer Sciences from École Polytechnique Fédérale de Lausanne in Lausanne. He is a member of the IEEE. His research interests are in the intersection of artificial intelligence and software engineering, automating the alignment of technological systems and their social context in which they create value.

Carmen López de la Torre is a former research fellow at University of Cantabria, Spain. She is currently working as high school teacher with particular interest on the ICT area. During her research career she has been active in different FP7 and H2020 EU international projects where her research interests have been focused in wireless networks optimization, IoT platforms and Smart Cities.

Noel Crespi holds Masters degrees from the Universities of Orsay (Paris 11) and Kent (UK), a diplome d'ingenieur from Telecom ParisTech, a Ph.D and an Habilitation from Paris VI University (Paris-Sorbonne). From 1993 he worked at CLIP, Bouygues Telecom and then at Orange Labs in 1995. He took leading roles in the creation of new services with the successful conception and launch of Orange prepaid service, and in standardisation (from rapporteurship of IN standard to coordination of all mobile standards activities for Orange). In 1999, he joined Nortel Networks as telephony program manager, architecting core network products for EMEA region. He joined Institut Mines-Telecom in 2002 and is currently professor and Program Director, leading the Service Architecture Lab. He coordinates the standardisation activities for Institut Mines-Telecom at ITU-T, ETSI and 3GPP. He is also an adjunct professor at KAIST, an affiliate professor at Concordia University, and gust researcher at the University of Goettingen. He is the scientific director the French-Korean laboratory ILLUMINE. His current research interests are in Service Architectures, Sofwarization, Social Networks, and Internet of Things/Services. http://noelcrespi.wp.tem-tsp.eu/

Gyu Myoung Lee (S'02, M'07, SM'12) received his BS degree from Hong Ik University, Seoul, Korea, in 1999 and his MS and PhD degrees from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2000 and 2007. He is with the Liverpool John Moores University (LJMU), UK, as Reader from 2014 and with KAIST Institute for IT convergence, Korea, as Adjunct Professor from 2012. Prior to joining the LJMU, he has worked with the Institut Mines-Telecom, Telecom SudParis, France, from 2008. Until 2012, he had been invited to work with the Electronics and Telecommunications Research Institute (ETRI), Korea. He also worked as a research professor in KAIST, Korea and as a guest researcher in National Institute of Standards and Technology (NIST), USA, in 2007. His research interests include Internet of things, future networks, multimedia services, and energy saving technologies including smart grids. He has been actively working for standardization in ITU-T, IETF and oneM2M, etc. In ITU-T, he currently serves as a WP chair in SG13, the Rapporteur of Q16/13 and Q4/20 as well as the chair of FG-DPM. He is a Senior Member of IEEE.

Roberto Minerva holds a Ph.D in Computer Science and Telecommunications from Telecom SudParis, France. He was the Chairman of the IEEE IoT Initiative, an effort to nurture a technical community and to foster research in IoT. Roberto has been for several years in TIMLab with responsibilities on service architectures. Currently he is involved in activities on SDN/NFV (technical leader of the SoftFIRE H2020 project), 5G, Big Data, architectures for IoT. Now he is a research engineer in Telecom SudParis working on IoT software architecture and the digitalization of businesses in several industries. He is author of several papers published in international conferences, books and magazines.

Luis Sánchez received the Telecommunications Engineering and Ph.D. degrees from the University of Cantabria, Spain, in 2002 and 2009, respectively. He is currently an Associate Professor at the Department of Communications Engineering, University of Cantabria. He is active on IoT-enabled smart cities, meshed networking on heterogeneous wireless scenarios, and optimization of network performance through cognitive networking techniques. He has a long research record involved in projects belonging to the fifth, sixth, seventh, and H2020 EU Framework Programs. He has authored over 60 papers at international journals and conferences and co-authored several books. He often participates in panels and round tables discussing about innovation supported by IoT in smart cities. He also acts as an expert for French ANR (Agencie National Recherche) and Italian MIUR (Ministero dell'Istruzione, dell'UniversitÃ e della Ricerca) reviewing and evaluating research and development proposals.