PlanetData

Network of Excellence

FP7 – 257641

D13.2 ParkMe Design

Coordinator: Jacek Kopecký

1st Quality Reviewer: Oscar Corcho
2nd Quality Reviewer: Snorri Gudmundsson

Deliverable nature:	Report (R)
Dissemination level: (Confidentiality)	Public (PU)
Contractual delivery date:	M16
Actual delivery date:	M16
Version:	1.0
Total number of pages:	20
Keywords:	linked data, geographical data, crowdsourcing, mobile, design, API

Abstract

ParkMe is a mobile and Web application that combines geographic data and parking space information with user location, social networks and other data sources in order to let its users conveniently find parking, and related added-value services, when coming to work or driving into town. ParkMe has a particular focus on gathering space availability data about car parks through crowdsourcing from the inputs of its users. In this deliverable, we describe the envisioned architecture of the system, including its public API, and we also include a draft of the "Terms & Conditions" for the application (including the Privacy Policy). Note that "ParkMe" may not be the final name of the app when it is published because there are by now several other apps with similar names.

Executive Summary

ParkMe is a mobile and Web application that combines geographic data and parking space information with user location, social networks and other data sources in order to let its users conveniently find parking, and related added-value services, when coming to work or driving into town.

ParkMe has a particular focus on gathering space availability data about car parks from the inputs of its users. The application will let its users add information about car parks and their up-to-date status, effectively crowdsourcing the creation of the parking data. Naturally, the application will publish the aggregate results as linked open data, to enable other third-party mashups and applications.

The primary users of ParkMe are smartphone owners who drive and are looking for parking spaces — ParkMe will encourage them to let the application know about the availability of car parks where they park, or try to find a spot and fail. Secondarily, car park operators may submit their authoritative availability data. The application does not inherently distinguish car park operators from other users, but reliable data sources from car park operators and others who submit data will emerge through a social graph of trust formed by ParkMe users.

Based on the analysis of requirements in D13.1, this present deliverable describes the envisioned architecture of the system, including its public API.

The envisioned architecture of the system is described in two parts: first, we analyze the structure of the ParkMe mobile app, showing its user interfaces, interactions with the outside world, and other internal components. Second, we detail the components of the back-end server, listing the data stores that it needs to include, the external data sources that it needs internally to façade, and the interfaces — both the API for use by the mobile app and possibly also by other third-party applications and mashups, and the Web application which will be the back-end server's direct user interface. We also analyze the data flows within the architecture of the server.

The API of the back-end server is designed to be resource-oriented and HTTP-friendly, especially focused on hyperlinking and support for caching. In short, we aim to make the API *RESTful*. The description of the API in this deliverable is primarily structured by the API's high-level architectural components, and secondarily by the Web resources that comprise the API.

In an appendix, we also include a draft of the "Terms & Conditions" for the application, which includes the Privacy Policy.

Note that "ParkMe" may not be the final name of the app when it is published because there are by now several other apps with similar names.

Document Information

IST Project Number	FP7 – 257641	Acronym	PlanetData
Full Title	PlanetData		
Project URL	http://www.planet-data.eu/		
Document URL	http://wiki.planet-data.eu/web/D13.2		
EU Project Officer	Leonhard Maqua		

Deliverable	Number	D13.2	Title	ParkMe Design
Work Package	Number	WP13	Title	Call1: ParkMe RTD

Date of Delivery	Contractual	M16	Actual	M16
Status	version 1.0		final ☒	
Nature	report ☒ prototype □ demonstrator □ other □			
Dissemination Level	public ☒ restricted to group □ restricted to programme □ consortium □			

Authors (Partner)	Jacek Kopecký (OU)			
Responsible Author	Name	Jacek Kopecký	E-mail	j.kopecky@open.ac.uk
	Partner	OU	Phone	+447946721458

Abstract (for dissemination)	ParkMe is a mobile and Web application that combines geographic data and parking space information with user location, social networks and other data sources in order to let its users conveniently find parking, and related added-value services, when coming to work or driving into town. ParkMe has a particular focus on gathering space availability data about car parks through crowdsourcing from the inputs of its users. In this deliverable, we describe the envisioned architecture of the system, including its public API, and we also include a draft of the "Terms & Conditions" for the application (including the Privacy Policy). Note that "ParkMe" may not be the final name of the app when it is published because there are by now several other apps with similar names.
Keywords	linked data, geographical data, crowdsourcing, mobile, design, API

Version Log			
Issue Date	Rev. No.	Author	Change
2012/1/16	0.1	Jacek Kopecký	First version for review
2012/1/25	1.0	Jacek Kopecký	After review comments

Table of Contents

1 Introduction

ParkMe is a mobile and Web application that combines geographic data and parking space information with user location, social networks and other data sources in order to let its users conveniently find parking, and related added-value services, when coming to work or driving into town. ParkMe has a particular focus on gathering space availability data about car parks from the inputs of its users.

The primary users of ParkMe are smartphone owners who drive and are looking for parking spaces — ParkMe will encourage them to let the application know about the availability of car parks where they park, or try to find a spot and fail. Secondarily, car park operators may submit their authoritative availability data, which they may have in electronic monitoring systems, or, as is the case at the Open University and other large employers, the data may come from security personnel who regularly monitor the car parks.

Because ParkMe does not expect to have explicit partnerships with car park operators (as explicit partnerships are relatively expensive to establish and maintain), the application does not inherently distinguish car park operators from other users. Any ParkMe user may choose to make their particular submissions public as a so-called "User's Data Source", which may be selected by other users as trusted, forming a social graph of trust.

Reliable data sources from car park operators are likely to be trusted by many users; but also ordinary users may make their data sources public, for example for the benefit of colleagues from the same department who happen to drive to work somewhat later, and who will be happy to know from the user what car parks are already full. As the general aggregation algorithm cannot judge a car park to be completely full after just one user says so (partly because that would make the data prone to manipulation), marking a colleague's data source as trusted will allow a user to see the estimate of "full" early, while others may still see it as "nearly full".

In Deliverable D13.1, we analyzed the requirements for the application, and the data sources that can be used in it. In this present deliverable, we include a summary of the requirements (in Section 1.1 below). The main contribution of this deliverable is a description of the envisioned architecture of the system (Section 2), and a specification for the application's public API (Section 3). Section 4 contains some concluding remarks and a brief overview of the development plan for ParkMe.

In effect, this deliverable represents a specification for the development work that needs to be done in order to realise the ParkMe applications. Additionally, Appendix A contains a draft of the "Terms & Conditions" for the application (including the Privacy Policy).

Note that "ParkMe" may not be the final name of the app when it is published because there are by now several other apps with similar names.

1.1 Requirements Summary

Deliverable D13.1 put the requirements on ParkMe in three categories: 1) requirements on the mobile app, 2) requirements on the Web application, and 3) requirements on the back-end services that will support the apps. Further, the requirements are characterized as *must-have requirements* (marked as *R.x.y* below) that describe the core capabilities of the system, *nice-to-have features for incentives (FI.x.y)* that will bolster the adoption of ParkMe, and *nice-to-have features for usability (FU.x.y)* that are will make the system easier to use.

The full list of requirements is summarized in Table 1.1. The result of the ParkMe project must meet all the *must-have requirements*, and it should meet as many of the *nice-to-have features* as possible, without undue bias towards either incentives or usability because both must be well-represented for the app to have a chance of success.

The mobile app is the main thing for ParkMe users, therefore it is the focus of most of the requirements; it is the only component for which we have identified incentive features. The ParkMe Web application should primarily offer data management functionality for registered ParkMe users; the requirements on the Web application are oriented towards this functionality. Optionally, the Web application may also include functionality comparable to the mobile app for map-based navigation and car park information; here, the requirements of the mobile app would apply. Finally, the ParkMe server must be able to process all the known car park location and availability information, relevant data sources, and user accounts. The requirements in the table cover the services provided by the server to the mobile and Web app.

1) Requirements on the Mobile App

Requirements	Incentive features	Usability features
R.1.1 Implementation in Android	**FI.1.1** Where did I park?	**FU.1.1** Highlighting nearest car park
R.1.2 Map-based UI	**FI.1.2** Parking time reminder	**FU.1.2** Ranking of found car parks
R.1.3 Car park information in the map	**FI.1.3** Additional information in the map	**FU.1.3** Voice information
R.1.4 Interpreting various user input as car park availability	**FI.1.4** Rich information about car parks	**FU.1.4** Voice interaction
R.1.5 Users as data sources	**FI.1.5** Ratings for car parks	**FU.1.5** External discoverability of user data sources
	FI.1.6 Related local services	**FU.1.6** Syncing of data
	FI.1.7 Related local data	
	FI.1.8 Unregistered use	

2) Requirements on the Web Application (optionally may include some of the above)

Requirements	Usability features
R.2.1 Registering user accounts	**FU.2.1** User data management
R.2.2 Deleting user accounts	

3) Requirements on the Back-end Services

Requirements	Usability features
R.3.1 Accessing and integrating external data sources	**FU.3.1** Optimizing access to real-time external data sources
R.3.2 Publishing static car park information	**FU.3.2** Provenance and information quality assessment
R.3.3 Publishing car park availability information	**FU.3.3** Processing on server
R.3.4 Publishing user data sources about car park availability	**FU.3.4** Scalability
R.3.5 API for user data management	
R.3.6 API for static car park data submission	
R.3.7 API for car park availability data submission	
R.3.8 Management interface	

Table 1.1: List of Requirements from D13.1

2 ParkMe Architecture

ParkMe consists of three main high-level parts: the mobile app itself, a back-end server that contains all the data and aggregates user input into parking availability estimates, and a Web application that gives a Web interface to the server.

The requirements identified in Deliverable D13.1 translate into a set of expected implementation components and their interactions. In Sections 2.1 and 2.2 respectively, we specify the architecture for the ParkMe mobile app and for the back-end server, which includes the Web application. In the outer page margins, we note the requirements and features directly related to the text. The architecture covers all the identified requirements.

2.1 ParkMe Mobile App Architecture

R.1.1 Figure 2.1 shows the main components of the mobile app, defined in the text below. The app will be developed on the Android system.

Naturally, the bulk of the mobile app deals with the user interface. The figure shows three envisioned interfaces: 1) the **main map-based screen**, 2) the screen with **the details of a car park**, and 3) the screen for invoking parking-related services directly from the app, by using the **OmniVoke** technology [2]. All these user interfaces will be realized as separate event-driven Android *activities*.

R.1.2 The main map-based screen will show a map, starting at the user's current location (using the built-in An-
R.1.3 droid "My location" overlay, which receives device location mainly from the GPS system), any car parks and
FI.1.3 other information (such as the locations of businesses selected by the user). At the top, the screen will show
FU.1.1 the current car park (either depending on the user's location, or another one selected by the user) along with its

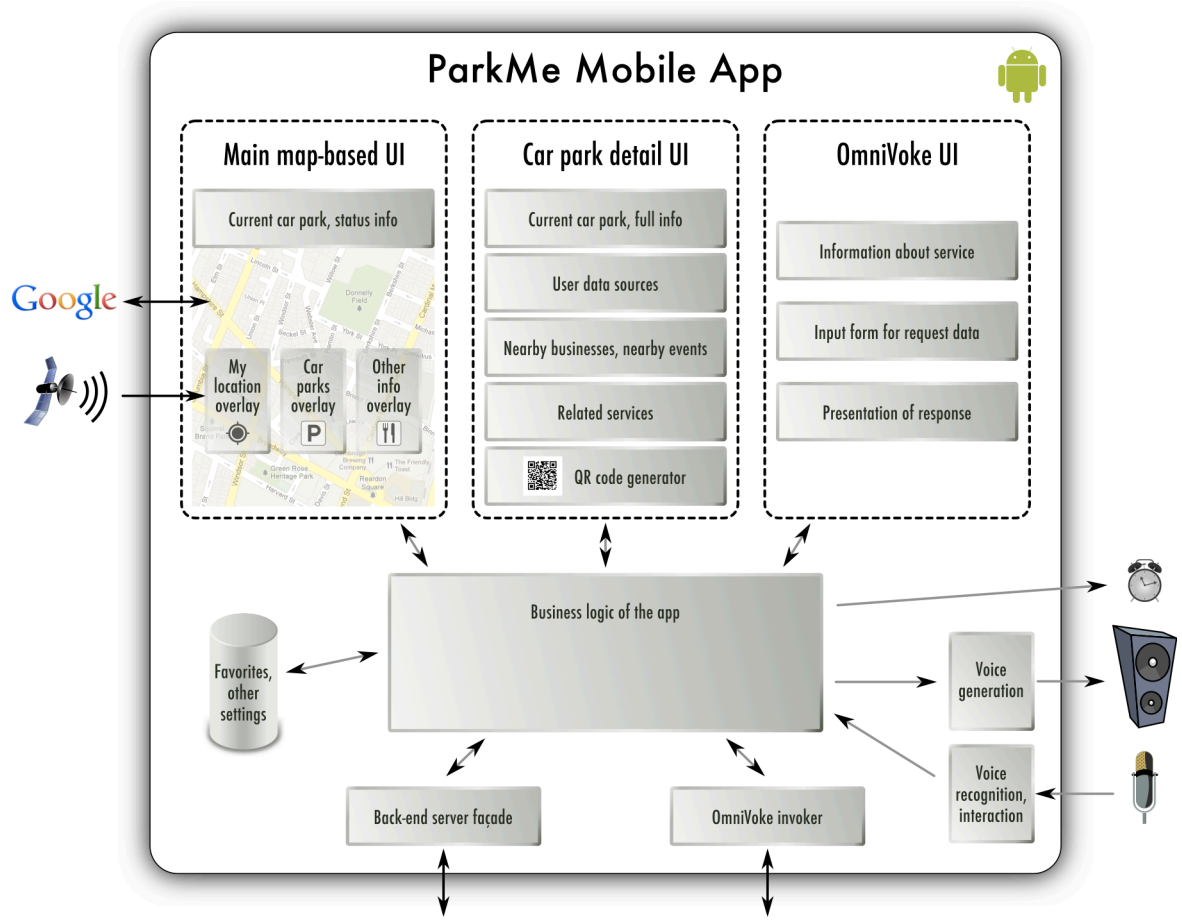

Figure 2.1: Architecture diagram for ParkMe mobile app

up-to-date availability status. The user will be allowed easily to submit an update of the availability status of the current car park. R.1.4

The car park detail screen will show various types of information about the car park. First, it will display all the static information about the car park (its address, opening hours, number of disabled parking spaces etc.), along with its rating (an aggregate of ParkMe users' submissions and external ratings gathered from the Web). Then it will list the user data sources for this car park (i.e. the data sources of users who maintain — or frequent — this car park), from which the user can select authoritative (trusted) sources. The screen will also show nearby businesses and events, and related services (such as car wash booking). Finally, if the user chooses to publish their submissions as a public data source for the car park, the screen will generate a QR code for easy sharing with other mobile users; through standard Android data sharing features, the app will allow the user to share the data source pointer (the QR code image or a textual link) in many ways, including by submitting it to social networks. FI.1.4 FI.1.5 R.1.5 FI.1.7 FI.1.6 R.1.5 FU.1.5

The OmniVoke screen will present a mobile user interface for the semantic Web service invocation platform OmniVoke [2], presenting a generic mechanism for using the services related to a car park directly from the ParkMe app. It will display information about the service, let the user formulate their request and, upon invocation, display the results. Alternatively, for services that advertise a Web interface or a specific Android app, ParkMe can simply forward the user there.

The central processing component of the mobile app is called "Business logic of the app" in the figure. It will coordinate data and user interaction flows. It is complemented by a local data store for the user's configuration (especially a list of favorite car parks, a list of trusted data sources, and the current parking location for the "Where did I park?" feature), a façade for communication with the API of the back-end server, and the OmniVoke semantic service invoker. The local data store in the mobile app will be implemented using readily-available tools of Android, i.e., we will not necessarily use RDF here. FI.1.8 FI.1.1

To implement a parking time reminder, the app will use the standard Android notification mechanism. Optionally, the app may also provide a voice interface: giving the user information (for instance, when the selected car park becomes full before the user reaches it), and interaction through voice commands from the user (for instance to find the nearest available car park). FI.1.2 FU.1.3 FU.1.4

2.2 ParkMe Back-end Server Architecture

Figure 2.2 shows the main components of the back-end server of ParkMe, defined in the text below. In the figure, the grey arrows indicate internal data flows, while the solid black arrows represent communication with external systems. An arrow indicates the direction of the communication: $A \longrightarrow B$ means A submits data to B, $A \longleftrightarrow B$ means data in exchanged in both directions, and $A \hookrightarrow B$ is a pull-style network interaction, where B requests some data from A (in other words, it is important to note that B initiates the communication, but the actual data goes from A).

As the server is mainly a database, the architecture diagram shows a row of data stores and façades for external data sources. While initially, all the data stores will be in a single RDF triple store (we will use OWLIM [1], through its Java Sesame API), scalability optimizations will likely require their splitting along the lines of the logical distinctions made in the diagram. FU.3.4

The core data stores contain information about car parks — any static information (such as location, opening hours, number of disabled parking spaces etc.), car park ratings, and up-to-date aggregate availability. All this data can be gathered from ParkMe users as well as from external sources (which is especially important for *seeding* the application with some initial data). Therefore, there is a crawler component that can access external data sources and look in them for new information. The crawler needs to be optimized for balance between information freshness and network communication which causes load on external servers, especially when accessing dynamic data sources such as SFPark for parking availability, or external sources of car park ratings. As a basis for the crawler, we will use either the crawler4j library[1], or Web-Harvest[2]. R.3.1 FU.3.1

[1] `http://code.google.com/p/crawler4j/`
[2] `http://web-harvest.sourceforge.net/`

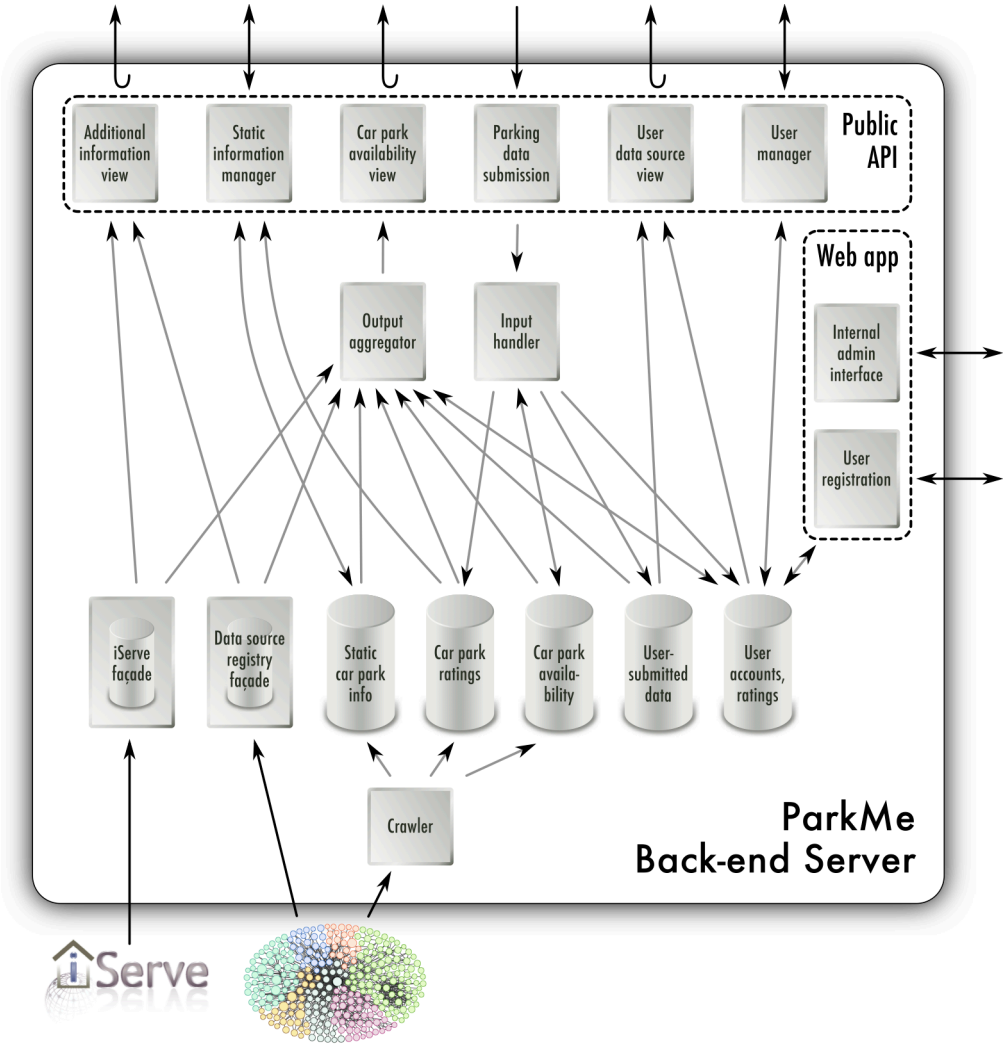

Figure 2.2: Architecture diagram for ParkMe back-end server

Other data stores in the server deal with information from and about ParkMe users: user accounts for storing
FU.1.6 users' settings and favourites (although the app must also be usable if the user hasn't registered), ratings for
FI.1.8 aggregating conflicting input from different users, and the actual individual submissions from users who choose to publish their parking availability data.

R.3.1 The server will also access external data stores, especially the service registry iServe[3] and a registry of public data sources; this access will be mediated by façade components in the server.

The server will provide a network API for accessing and submitting data, implemented in a resource-oriented and HTTP-friendly manner (see Section 3 for details), using Java's standard JAX-RS REST support, and its reference implementation, Jersey.

R.3.2 Mainly, the server will publish all its information about car parks, in three separate parts of the API: 1) ac-
R.3.6 cessing (and submitting updates to) the mostly-static data (location, opening hours, ratings), 2) a view on the
R.3.3 parking availability data, and 3) a view on the user data sources. These interfaces will include query capabilities:
R.3.4 the static data must be able to provide car parks in a given area, the availability data must be able to integrate the aggregate availability data with selected user data sources (for instance for trusted data from a car park operator).

To provide the mobile app with suitable access to the external data sources (esp. iServe), with additional car-park-related information such as the services available in the car parks, businesses or events in the area etc.,

[3] http://iserve.kmi.open.ac.uk

the server itself will façade these data sources to the mobile app through the "Additional information view" part of the API.

As the main point of ParkMe is to crowdsource parking availability data, a crucial part of the API will handle submission of availability information. There will be a number of ways the mobile app can let the server know, sometimes indirectly, about parking availability status, as discussed further in Section 3. R.3.7 R.1.4

The two central processing components of the server are an "Input handler" and an "Output aggregator", serving mirror roles in the application. The input handler will get parking availability submissions and mainly aggregate them in the general car park availability data and save it in the user's public parking availability data source (if the user chooses to publish it). The submitted input can also have bearing on car park ratings (to indicate a measure of the popularity of a car park by ParkMe users) and user ratings (for instance if a user is detected to often submit conflicting information, they may be marked likely malicious). On the other side, the output aggregator will react to queries by the mobile app for availability and further information about selected car parks, with a given list of trusted user data sources to treat preferentially, and a type of businesses and services that the user may be interested in. The aggregator will integrate all the available information and rank the results, but it may also note in user ratings if some user's data source is often taken as trusted. FU.3.2 FU.1.2

To support user registration and data management (especially for syncing between the mobile app and the server), the server will provide a user data management API, and user registration will also be an important functionality of the Web application provided by ParkMe. Optionally, the Web application can also give richer functionality for user settings management, and it may also contain some of the features of the mobile app. Finally, the Web application will also handle the server's administration interface, for tasks including (de)registering interesting external data sources such as geographical data, relevant business directories, event lists, statistical data about local areas etc. FU.1.6 R.3.5 R.2.1 R.2.2 FU.2.1 R.3.8

As already implied, the back-end server will be primarily implemented in the Java programming environment. It will be deployed in a Tomcat Web application container on a virtualized Unix server with redundant network-based storage.

Note that depending on the performance of the first ParkMe prototypes, the architecture may later be affected if some processing needs to move from the mobile app to the server. FU.3.3

3 ParkMe Public API

The back-end server of ParkMe will provide a network API for accessing and submitting data. The API is designed in a resource-oriented and HTTP-friendly manner, especially with focus on hyperlinking and support for caching. In short, we aim to make the API *RESTful*.

In particular to support caching (and the scalability benefits it brings), all data returned by cacheable operations will be complemented by meaningful caching metadata: an expiration time (on the order of minutes for parking availability estimates, longer for static car park data) and last update time. Thus clients and caches will be able to rely on earlier-retrieved data, and subsequent requests for new data will only cause processing and data transfers when the data has, in fact, been updated since the client (or cache) last retrieved it.

In the subsections below, we describe the components of the ParkMe back-end server API, focusing on the resources they provide and the relevant operations on these resources. Table 3.1 contains a summary list of the resource types of the ParkMe API, along with their place in the architecture components, and the number of the section where we discuss the component. This table may serve as a reference when looking for a discussion of some particular resource type. The codes (such as PAVAIL, UDS) are used when referring from resource descriptions to other resources, especially when describing hyperlinks.

Several of these resources are providers of Linked Data: PARK with static information about car parks, UDS containing users' data about parking availability, and PAVAIL representing aggregate parking availability estimates. This aspect is also discussed in the sections below.

Code	Description	API component	Section
PARKS	List of known car parks	Static Information Manager	3.1
PARK	Static information about a car park	Static Information Manager	3.1
PNEAR	Queries for car parks, their availability, further info	Car park availability view	3.2
PAVAIL	Aggregated availability and other information for a single car park	Car park availability view, Parking data submission	3.2, 3.3
UDS	A user's data source for a car park	User data source view, User manager	3.4, 3.5
UBACK	User's settings from the mobile app	User manager	3.5
UPROF	Public profile of a user	User manager	3.5
UA	User account	User manager	3.5
ANEAR	Queries for additional data near a location	Additional information view	3.6
ADATA	A single piece of additional data related to parking	Additional information view	3.6

Table 3.1: Listing of main resources of ParkMe API

3.1 Static information manager

The static information manager provides read and update access to the mostly-static information about car parks, such as location, opening hours, number of disabled parking spaces etc. Effectively, this component publishes car park data known to ParkMe as Linked Data, in resources of the type PARK. For a simple query interface for the static data, and for the functionality of adding car parks, there is the resource PARKS, representing the list of car parks known to ParkMe.

Note that data submissions for car park static data will initially be moderated (in order for us to control the quality of the data); therefore data submission methods (POST, PUT, DELETE) here will return the HTTP status code 202 Accepted. These methods may also be invoked with authentication; registered users may have priority in the submission moderation queue.

The PARK resource type is ParkMe's main provider of Linked Data: car parks submitted by ParkMe users will get URIs that will resolve to their PARK description resources. Each PARK resource will further link to the PAVAIL resource that contains aggregate availability estimates for this car park, enabling follow-your-nose discovery.

Resource PARK: static information about a car park (one per car park known to ParkMe)

GET	retrieve static information about the car park, along with its rating information links to PAVAIL for simple availability information (no parameters)
POST	submit an update of the data (moderated) the submitted triples will be seen as replacements for parts of the existing data
DELETE	submit a report that the car park doesn't exist (moderated)

Resource PARKS: list of car parks (singleton, parametrized)

GET	query for car parks near a location (this will not necessarily be used by the mobile app) parameters: geo point, radius returns list of car parks (only locations and links to PARK resources)
POST	submit a new car park creates a new PARK resource which only becomes listed in availability queries after approval

3.2 Car park availability view

The car park availability view will give simple read access to the aggregate estimates of car park availability (PAVAIL resources), and a rich query interface over all the data known to ParkMe (in the PNEAR resource).

For estimating the availability status of a car park, the system uses the base data aggregated from all user submissions, but the client may provide a list of UDS data sources that should be treated as trusted by the current query. This way, different users may see different availability status depending on the data sources they trust: for example in highly-dynamic situations when a busy car park is nearly full, the general aggregation algorithm cannot pronounce a car park as completely full after just one user says so (partly because that would make the data much too prone to manipulation), but if a trusted source does say the car park is now completely full, the user who trusts that source will see the estimate of "full", while others may see it as "nearly full". Optimally, authoritative data sources will emerge, so there should be few cases where different users see wildly different results.

Further, along with car parks with their locations and availability estimates, the user may request additional information such as the services available in the car parks, user data sources for the car parks, businesses or events in the area etc. This is a network communication optimization, to avoid multiple network requests (and the associated latency); the Car park availability view API component can collect all the needed information and return it in response to a single request.

The PAVAIL resource type provides the dynamic car park availability portion of ParkMe's Linked Data. The PAVAIL resource, as linked from a PARK resource, will be parametrized to also contain a list of user data

sources for this car park (the UDS resources). In this way, coming from PARK resources, a client can dynamically discovery all the relevant UDS resources as well.

Resource PAVAIL: the availability of a car park (one per car park, parametrized, also discussed below)

GET retrieve the car park's detailed availability information
parameters:
- a list of trusted data sources
- types of requested additional information for the car park (services, user data sources)

POST *defined below*

Resource PNEAR: car parks near a location (singleton, parametrized)

GET query for car parks and additional information near a location
parameters:
- geo point, radius,
- a list of trusted data sources
- types of requested additional information for the car park
  (businesses, services or events in the area, possibly with specific categories)

returns:
- car parks with estimated availability, just locations, names and simplified availability info
- list of additional data (businesses, services, events) with locations, types and categories

3.3 Parking data submission

Submitting data about parking availability is the core operation of the API. For usability, the API must be able to accept various types of user submissions: a simple flag that the car park has places or is full, possibly with subtler distinctions such as "getting full" or "many people are leaving"; precise submission with a number of available places (which we can expect from car park owners); or even information that a user is parking in the car park (a sign that there probably still are spaces available, if our user found one), or that the user is leaving — the application may be able to submit these kinds of indicators even if the user does not feel like submitting more concrete information.

ParkMe cannot simply accept all submissions at face value, therefore it has to aggregate them according to various estimates of the quality of the data, for instance if it comes from an anonymous user or from a registered one, if the user is seen as trusted by many other users, and if the user's submissions have in the past been accurate (validated by further submissions from others).

Resource PAVAIL: the availability of a car park (one per car park, also discussed above)

GET *defined above*

POST submit a record that the car park is full or empty, optionally with a number of available places,
or submit a record that the user is parking in or leaving this car park
(the client may choose to provide authentication credentials, for reputation-evaluating purposes, but also because for an authenticated user who publishes a UDS for this car park, the submission will propagate there)

3.4 User data source view

The user data source view provides public access to the parking data published by a user (UDS resources), which may be useful outside of the ParkMe application. Note that since every data submission overrides previous submissions by this user, the data source can only provide a single data point. Also, since parking availability information decays with time (it becomes less and less trustworthy), the user data source may contain no data at all if the last submission is expired.

The UDS resource type can be seen as a provider of Linked Data (the user's submissions), and each UDS resource links to its respective PAVAIL resource parametrized to treat the UDS resource as trusted.

Resource UDS: A user's data source for a car park (one per user per car park, also discussed below)

GET returns the last entry given by this user for this car park, unless it has expired,
includes a profile description of the user, includes hyperlinks:
- PARK — the car park itself,
- parametrized PAVAIL — availability estimate for the car park, trusting this source

DELETE *defined below*

3.5 User manager

The final API component is the User manager, which handles user data (profile, settings and parking availability data sources) and account deletion. Account creation (user registration) and password changes are only available through the Web application, therefore there is no need to model them in the API.

The resources provided by the User manager all require authentication by the client. They are modeled as single resources whose content varies by who accesses them.

Users that want to publish parking availability data sources will be asked to provide a short textual description (profile) of themselves, so that others can evaluate whether they should use this source. We will recommend that the profile description include contact details (such as a phone number).

Resource UA: User account (singleton, varying through authentication)

GET retrieve the user's account data, including the profile description,
includes hyperlinks:
- UBACK — for retrieving or storing the user's mobile app settings (for backup),
- UPROF — for updating the user's profile description,
- UDS — all the user's parking data sources

Resource UDS: A user's data source for a car park (one per user per car park, also discussed above)

GET *defined above*
DELETE delete the data source because the user no longer wants to publish it for this car park

Resource UPROF: Public profile of a user (singleton, varying through authentication)

GET retrieves the profile description of the user
PUT updates the profile description of the user

Resource UBACK: User's settings from the mobile app (singleton, varying through authentication)

GET retrieves the user's previously saved settings
PUT updates the user's settings

3.6 Additional information view

This component provides a view on the additional car-park-related information such as the services available in the car parks, businesses or events in the area etc., which is façaded from external data sources. It contains two types of resources: ADATA for a single piece (an RDF graph) of additional data, for example a description of a service or an event, according to the Linked Data publishing principles; and ANEAR for querying for services, businesses or events near a given location.

Resource ANEAR: additional services/businesses/events near a location (singleton, parametrized)

GET query for services/businesses/events near a location
parameters:
- geo point, radius,
- types of requested data
  (businesses, services or events in the area, possibly with specific categories)

returns: list of descriptions (of businesses, services, events) with locations, types and categories

Resource ADATA: additional service/business/event (one per description)

GET retrieve all the information about this particular service/business/event
(this will façade iServe for services, plus other data sources, and possibly transform the data)

4 Conclusions

In this deliverable, we have described the architecture and the API of the ParkMe application. As noted in the text already, it is possible that both the architecture and the API may still evolve as we implement the system, especially in response to scalability and performance considerations.

At the end of this deliverable, in Appendix A, we include a draft of the "Terms & Conditions" for the application, which includes the Privacy Policy. As a crowdsourcing application, ParkMe must be clear about what data is collected and how it is used, so that we do not detract possible users by worries for their privacy.

REFERENCES

[1] Barry Bishop, Atanas Kiryakov, Damyan Ognyanoff, Ivan Peikov, Zdravko Tashev, and Ruslan Velkov. OWLIM: A family of scalable semantic repositories. *Semantic Web*, 2(1):33–42, 2011.

[2] Ning Li, Carlos Pedrinaci, Maria Maleshkova, Jacek Kopecky, and John Domingue. OmniVoke: a framework for automating the invocation of Web APIs. In *ICSC 2011 Fifth IEEE International Conference on Semantic Computing*, 2011. Available at `http://oro.open.ac.uk/29272/`.

Appendix A ParkMe Terms and Conditions

These are the terms and conditions of use for the mobile app ParkMe, and the associated website.

You may use the mobile app and the website without registering. You can register (providing us only with your email address, which we won't give to anyone else) to give more weight to your parking availability data and to be able to share your data with other users of ParkMe. If you register, you agree NOT to use a name of another person or entity with the intent to impersonate them, and you agree NOT to use a name that is offensive, vulgar, obscene or unlawful. You further agree NOT to use the account of another person or entity without the proper authorization. Finally, you agree NOT to submit knowingly false information about car park availability to ParkMe. ParkMe reserves the right to refuse registration, or to cancel an account found in violation of the above terms.

ParkMe does not guarantee that any data will be available through the application. All data submitted to ParkMe is the sole responsibility of the person who originated such submission. ParkMe cannot guarantee the authenticity of any data provided by its users. You acknowledge that all data submitted and accessed by you using ParkMe is at your own risk and you will be solely responsible and liable for any damage or loss to you or any other party resulting therefrom.

You shall not (directly or indirectly):

- take any action that imposes or may impose (as determined by ParkMe in its sole discretion) an unreasonable or disproportionately large load on its infrastructure;
- interfere or attempt to interfere with the proper working of the application;
- bypass any measures ParkMe may use to prevent or restrict access to the Service;
- run any form of auto-responder or "spam" on the application.

As ParkMe is a mobile app, the users must be aware of applicable law and regulations on using mobile devices, especially when operating and parking vehicles. ParkMe cannot be held liable for the actions that its users take while using the app, or as consequence of using the app.

Privacy Policy

This Privacy Policy covers ParkMe's treatment of personally identifiable information and other data that ParkMe gathers when you are using the app. ParkMe is being developed by researchers at The Open University, Milton Keynes, United Kingdom.

What Personally Identifiable Information Does ParkMe Collect?

What you tell us: for submitting parking data into the system, we require your email address; for sharing your entries with other users we further need your name and you can also supply a photo or a logo. Naturally, ParkMe also stores all the parking data you submit, which means your location, down to the level of the car park you use, and the time when you submitted it. Your email address is used to distinguish your data from that of other users and to send you important updates about the service (such as significant changes to these terms and conditions); your name and photo/logo is displayed to other users if you choose to share your data.

What we collect automatically: When you use ParkMe, our server automatically receives and logs information from your browser or mobile platform, including your location, IP address, cookie information, and the page you requested. We treat this data as non-personal information (we do not tie it to your email address), and we only use this data in aggregate form.

How is My Information Shared?

If you do not enable sharing of your submitted parking data, ParkMe only uses your submissions in aggregate form to establish whether car parks have any spaces available. If you enable sharing, your name, photo/logo,

and your recent submitted parking data (up to 24 hours old) may show up in lists of users who visit a particular car park; it may be available through search, and it will be available directly to whoever sees your sharing link (QR code).

If you disable sharing, our server will no longer provide your data, but the internet and users who have seen your earlier data may store copies of it and disclose it to others, for any period of time. We therefore urge you to think carefully about publishing your parking information if you have concerns about your location privacy.

As ParkMe is a research project, your data is not stored by a commercial entity for the purpose of generating income. If ParkMe ever becomes commercial, you will be duly notified and given the option of removing all your data before it is transfered to the commercial owner.

We may release personal information when we believe in good faith that release is necessary to comply with the law, including laws outside your country of residence; or protect the rights, property, or safety of the University, its employees, ParkMe users, or others.

Email Communications: If you do not want to receive any email from us, please indicate your preference during the registration process or by making a modification on your account settings page, once you're logged into the service. Please note that if you do not want to receive legal notices from us, such as notices regarding this Privacy Policy, those legal notices will still govern your use of the ParkMe, and you are responsible for reviewing such legal notices for changes.

Is Personal Information About Me Secure?

Your ParkMe account is protected by a password for your privacy and security. You need to prevent unauthorized access to your account by selecting and protecting your password appropriately, and limiting access to your computer and browser by signing off after you have finished accessing your account.

We endeavor to safeguard user information to ensure that user account information is kept private. However, we cannot guarantee the security of user account information. Unauthorized entry or use, hardware or software failure, and other factors, may compromise the security of user information at any time.

Overseas Users

If you are located outside of the United Kingdom (UK), please note that the ParkMe site is hosted on our computer servers in the UK. Therefore, your information may be processed and stored in the UK. As a result, UK government, courts, law enforcement or regulatory agencies may be able to obtain disclosure of your information through laws applicable in the UK. Your submission of any Personal Information to ParkMe will constitute your consent to the transfer of your Personal Information over the internet to UK. The data protection rules in the UK and other countries through which your data may pass may be different than in your country.

How Do I Delete My Account?

Should you ever decide to delete your ParkMe account, you may do so by clicking on the "delete account" link on your account settings page. If you terminate your account, its data will be removed from the site and deleted from our servers. Because of the way we maintain the servers, such deletion may not be immediate, and residual copies of your information may remain on backup media a reasonable amount of time.

What If I Have Questions Or Concerns?

If you have any questions or concerns regarding privacy using ParkMe, please send a detailed message to j.kopecky@open.ac.uk. We will make every effort to resolve your concerns.

Last update: 2012/01/15